SYSTEMATIC INTEGRATION OF URBAN FARMING INTO URBAN METABOLISMS

Waste As A Resource For Urban Food Production

Eren Gozde Anil
5263557

Delft University of Technology
Master of Architecture
Building Technology Track
Graduation Project Report (P4)

Delft University of Technology
MSc. Architecture, Urbanism and Building Sciences
Building Technology Graduation Studio AR3B025
Eren Gozde Anil | 5263557
June 2022
Mentors: Andrew Jenkins | Climate Design & Sustainability
Michela Turrin | Design Informatics

TUDelft Delft University of Technology
Faculty of Architecture
and the Built Environment

Abstract

As the global population rise, climate conditions get more and more unpredictable, natural resources deplete; cities need to take action in order to sustain healthy living conditions as well as to ensure food safety. Currently, cities are solely dependent on external sources and suburban areas for natural resources and food as well as waste management. This linear metabolism results in cities consuming 60-80% of natural resources and producing 50% of waste globally. (Tsui et al., 2021) This problem can be overcome by introducing urban farming into cities by utilising waste and underused space as a resource for urban food production. Waste can be circulated in the city in order to generate a network of waste producing functions and farms.

There are urban farming systems which can digest waste and produce supplements for urban food production. However, the quest of choosing an urban farming system based on existing vacant spaces and waste flows is a complicated task. The complexity is a result of variables in the equation which may effect decision making such as different systems, waste types, vacant space characteristics as well as the size of spaces and the quantity of available waste. Moreover, in sites consisting of numerous vacant spaces and waste sources decision making is even more complex and laborious. If human designers were to perform this task then they would need to iterate countless times for each vacant space, each waste source close to it and each potential urban farming systems. However, when it comes iterating and repeating the same steps, computers are explicitly faster, time-efficient and error free. Therefore a decision making tool which can assist designers to choose urban farming systems based on existing conditions can be a practical resource.

This paper investigates how to integrate urban farming into cities by utilising under-used spaces and existing waste sources via using a decision making tool. The design rules and the methodology are formed based on literature review regarding different farming systems, varying waste flows and computational approaches. A prototype of the tool is generated and tested on 2 case studies in order to showcase the potential of such an approach combining food production with waste management.

Keywords: *urban farming, decision making tool, waste as a resource, urban farming systems*

Acknowledgements

This 8 months long adventure on working with this project made me grateful to a lot of people for their guidance and support along the way. I would like to use this opportunity to thank my dear mentors Andrew Jenkins and Michela Turrin who guided and supported me all along the project. I am indebted to my mentors for making these 8 months the greatest I had in TU Delft.
Urban farming and urban waste flows were new to me when I started this graduation topic. And thanks to Andrew Jenkins, my interest in the topic never faded instead it rised even more. I am truly grateful for all the tutoring sessions and the guidance as well as the constant encouragement.

To be completely honest, there were times I was unsure how to start and which way to go with the decision making tool. I would like to thank Michela Turrin to guide me out of the puzzle by questioning and making my ideas surface. I am more than grateful for all the eye-opening and inspiring tutoring sessions.

It is undoubtably true that, I gave never-ending speeches on my graduation project to all my family members and close friends in order to understand and deal with the puzzle I had in my hands. I would like to thank my family for supporting and always believing in me through out my education and especially all along my graduation year. I would like to thank my close friends who were here studying with me and to the ones who were there for me virtually.

Lastly, I would like to thank TU Delft's The Architectural Engineering + Technology department for giving me the opportunity to work on innovative topics and creating the grounds for an eye-opening experience and a great learning environment. There were times I questioned my decision with doing a masters in Building Technology. However looking back I am more than gratified and thankful.

TABLE OF CONTENTS

01

RESEARCH FRAMEWORK

1.0 RESEARCH FRAMEWORK

Introduction

The global population is estimated to rise by 50% by 2050 compared to 2013 (Graamans, 2021), resulting in increased food demand. Extreme weather conditions such as drought and higher temperatures are expected and to overcome these challenges climate change brings, agriculture should be climate-proof in order to adapt to extreme weather conditions and become resistant to plagues (Roggema, 2009). In other sources, it is stated that current food supply networks are robust to random failures however they are vulnerable to targeted cascading disturbances (Graamans, 2021) and climate change due to changes in CO2 levels, precipitation, temperatures, rainfall variability, pests, and diseases (Fresco, 2009). Therefore climate-resilient, flexible, and sustainable agricultural systems should be introduced which are productive, responsive to change, and resource-efficient as advocated by Fresco (2009). Roggema & Keeffe asserts that otherwise, a food crisis may happen if this vulnerable, dependent system fails (2014). Other challenges food production faces are land scarcity and dependency on rural areas for food as stated in the same paper. Urban farming is a promising solution to overcome these challenges regarding land scarcity and food security by providing alternative spaces to grow crops. On the other hand, cities lost their place in the chain to contribute to sustainability as the urban areas are dependent on delta areas for energy supplies and to global food supply chains (Roggema & Keeffe, 2014). Food is imported to the cities from various locations by passing through many stakeholders consuming energy to store and transport and food is wasted on the way due to poor storage conditions. Moreover, the urban metabolism is based on linear processes such as consumption of energy, natural resources, food, and land while producing waste, greenhouse gas emissions. As reported by Tsui et al. cities consume 60-80% of natural resources while producing 50% of waste and emitting 75% greenhouse gas globally (2021). By introducing urban farming, cities can regain their influence on sustainability, and the loops of urban metabolism can be closed.

In this paper, a systematic way of integrating urban farming into urban metabolisms by utilising waste as a resource will be discussed. Existing vacant spaces within or on the buildings are proposed to be used as potential urban farming locations as cities tend to be dense urban areas with little or no available space. There is no one size fits all solution to close the loop, design problems are location specific and multifaceted as there are various types of urban farming systems which require different resources and are capable of utilising different kinds of waste in addition to generating different outputs. Consequently, a decision-making tool is proposed as a solution to this multifaceted problem. The decision-making tool is designed to support designers to choose the most suitable urban farming system depending on existing factors such as vacant spaces, waste flows within an urban development in the early stages of design. Various waste types, including CO2, organic waste, black water, rainwater, residual heat, are covered in the following sections and each is assessed in terms of their potential and feasibility to be used as an input for the urban farm. This urban development can be a neighbourhood, a university campus or a specific region within the urban fabric. During the decision making, the aim of the intervention is considered and the most suitable system is proposed to the designer by the tool.

Problem Statement

Cities consume 60-80% of natural resources while producing 50% of waste and emitting 75% greenhouse gas globally (Tsui et al., 2021). Demands of the city are supplied externally resulting in a dependent system that is not resilient to fluctuations. Demands and waste flows of cities follow a linear process line resulting in high GHG emissions, making the 2050 goals (European Commission, 2020) become unachievable if a change in the design of urban metabolism is not made. Integrating urban farming as part of an urban metabolism has the potential to close the loop of some organic materials and resources within the urban environment however vast amounts of waste types and urban farming systems complicate the decision making process.

Sub-problems:

The primary needs of a city are clean water, energy, materials and food. Food is imported from outside resulting in energy consumption and waste due to transportation, storage conditions, in addition, to complete dependency on external factors and global supply chains which are vulnerable to climate conditions. Secondly, cities are currently dependent on delta areas for energy supplies. By 2050, renewable energy sources should replace conventional ones (Ministry of Economic Affairs, 2017) and to do so waste energy can be utilised and repurposed. Thirdly, global water demand is expected to exceed supply by 40% within 20 years (UNEP, 2014 as cited in Gondhalekar & Ramsauer, 2017) Clean water is a crucial factor for the future city therefore water-efficient systems should be integrated into proposed urban farms. Moreover, in agriculture burning fossil fuels is common practice to increase CO2 levels in the greenhouses (Bao et al., 2018), however alternative circular solutions must be found. Lastly, the design problem of introducing an urban farm utilising waste as an input in urban metabolisms is a complex multifaceted problem that requires prior knowledge.

Objective:

Making a shift to a more resilient, less dependent urban metabolism can be achieved by introducing urban farming systems to urban development by utilising the waste flows as resources for the urban farm. A suitable urban farming system is highly dependent on the location, the situation demands, existing spaces, and existing waste flows. The objective of this thesis is to provide a decision-making strategy, therefore a decision-support tool of urban farming systems utilising various waste streams from the site with the main aim of closing the loop within the urban metabolism.

Sub-objectives

While designing the decision-making tool the following aspects should be paid attention to:

The loop of waste flows including water, energy, food should be closed by using the urban farming system as a hub of resource utilisation and exchange.

Proposed urban farming systems should introduce alternative food supply points in addition to global supply chains.

A decision-making tool guiding designers to choose urban farming systems should be developed.

Research Question

This thesis aims to answer the following questions:

> In which situations can different urban farming systems employ different urban waste flows in order to promote the circularity of food production and resources in urban contexts by augmenting the design process with decision support systems?

Sub Research Question:

Which kind of waste flows are viable to be utilised by the urban farm?

Which kinds of urban farming systems are suitable to repurpose the urban waste flows including water, CO2, heat, organic waste?

How can urban farming systems be combined and have a symbiotic relationship to close the loop within the urban metabolism?

Which computational approaches are feasible to construct the decision-making tool serving the purpose of generating a network of inputs, outputs, and urban farming systems (operators) with given criteria and rules to design?

Background Questions:

What are the commonly practised indoor farming strategies and are there any limits to these systems?
What kind of effects does CO2 have on crops?
What are common practices to increase CO2 levels in the greenhouses?
What kind of measures should be taken when utilising black water as an input for the urban farm for the sake of public safety?
What kind of measures should be taken to ensure the safety of collected rainwater?
Is there any risk of using rainwater for the irrigation of crops without processing it?
What are the potential benefits of utilising rainwater for irrigation?
Which kinds of wastes are suitable as substrates for mushroom production?
What kind of mushrooms can be grown in different kinds of waste?
What are the potential advantages of mushroom production?
Which kinds of wastes are suitable for vermiculture?
What are the potential advantages of vermiculture?
How can residual heat from the urban farm be utilised or stored to heat buildings?
What are the challenges of the present food supply system?

Final Product:

The final product of the thesis is a decision-making tool to support designers to choose fitting urban farming systems to existing conditions such as vacant spaces, waste flows in the vicinity, and project goals like maximum food production, research or holistic food production. The tool will be tested on case studies including TU Delft Campus as the primary case study while the site of Urban Greenhouse Challenge in Washington is the secondary case study. The outcomes will be illustrated by using the decision-making tool on 3d models of the locations.

Research Scope:

The proposed strategy or design method is constructed to be used for an urban setting such as a neighbourhood, a university campus, or a region within the city.

The decision-making tool will be designed to be applicable for various locations and circumstances however these circumstances only include temperate climates as including warmer climates may introduce complications to design.

Load-bearing capacity is one of the bounding conditions when working with existing vacant spaces. Roofs tend to have a lower load-bearing capacity than intermediate floor levels. Consequently, the weight of the system has to be considered; however for the development of the decision-making tool, the load-bearing capacity of the structures will not be calculated. Instead lighter systems will be suggested for spaces with potentially lower load-bearing capacities, and heavier systems for spaces that may have higher load-bearing capacities.

The main purpose of the decision-making tool is not to design the whole system but to make suggestions regarding the urban farming system(s) to use.

Urban farming has the potential to make a social impact by improving the psychological well-being of nearby inhabitants, decreasing food prices, providing job opportunities, improving the health and well being of the community, and increasing community sense. However, only quantifiable benefits are in the scope of the thesis, the rest will be treated as secondary results. These include amount of waste that can be used as resource and amount of food which can be produced in the farms

In this project, vacant spaces are defined as spaces which do not have any designated purpose. Therefore, parking lots, sports fields and already utilised roofs are not treated as vacant.

Waste sources which are in the scope of the research are food waste, spent coffee grounds, paper, sawdust, CO2, rainwater and heat.

It should be noted that the software or tools which are mentioned in the following sections are the ones that are used for this research and they can be replaced by any other tool if needed. However, the logic and methodology behind these steps remain the same.

A prototype of the decision making tool is developed based on the set design rules and methods however data collection is done manually due to time and resource restrictions.

Waste quantities, vacant space sizes, waste demands of systems and yields are calculated based on simplified calculations disregarding environmental factors.

Waste sources included in case studies are not assessed in terms of life cycle. Some of the waste sources are already being used as an input by other industries. However, these are still included in the decision making for research purposes.

Research Approach & Methodology:

The research has three main components: literature review, decision-making criteria & rules, and case studies to test the decision-making tool as illustrated in figure 1.1.

Literature Review: A literature review is conducted in order to construct the database of different urban farming systems, their inputs, and outputs. Secondly, literature is studied to assess the potentials and feasibility of different urban waste flows (organic waste, black water, rainwater, heat, CO2) to be utilised as inputs for urban farms. Thirdly, various computational approaches, their advantages, disadvantages, and suitability to the research are investigated.

Based on the literature review, a database of each urban farming system, its inputs, outputs, and system overview is constructed in order to assist the decision-making process. Suggested systems consisting of waste flows, urban farming systems, and agricultural produce are based on literature review and the applicability of the interventions are validated by reference projects.

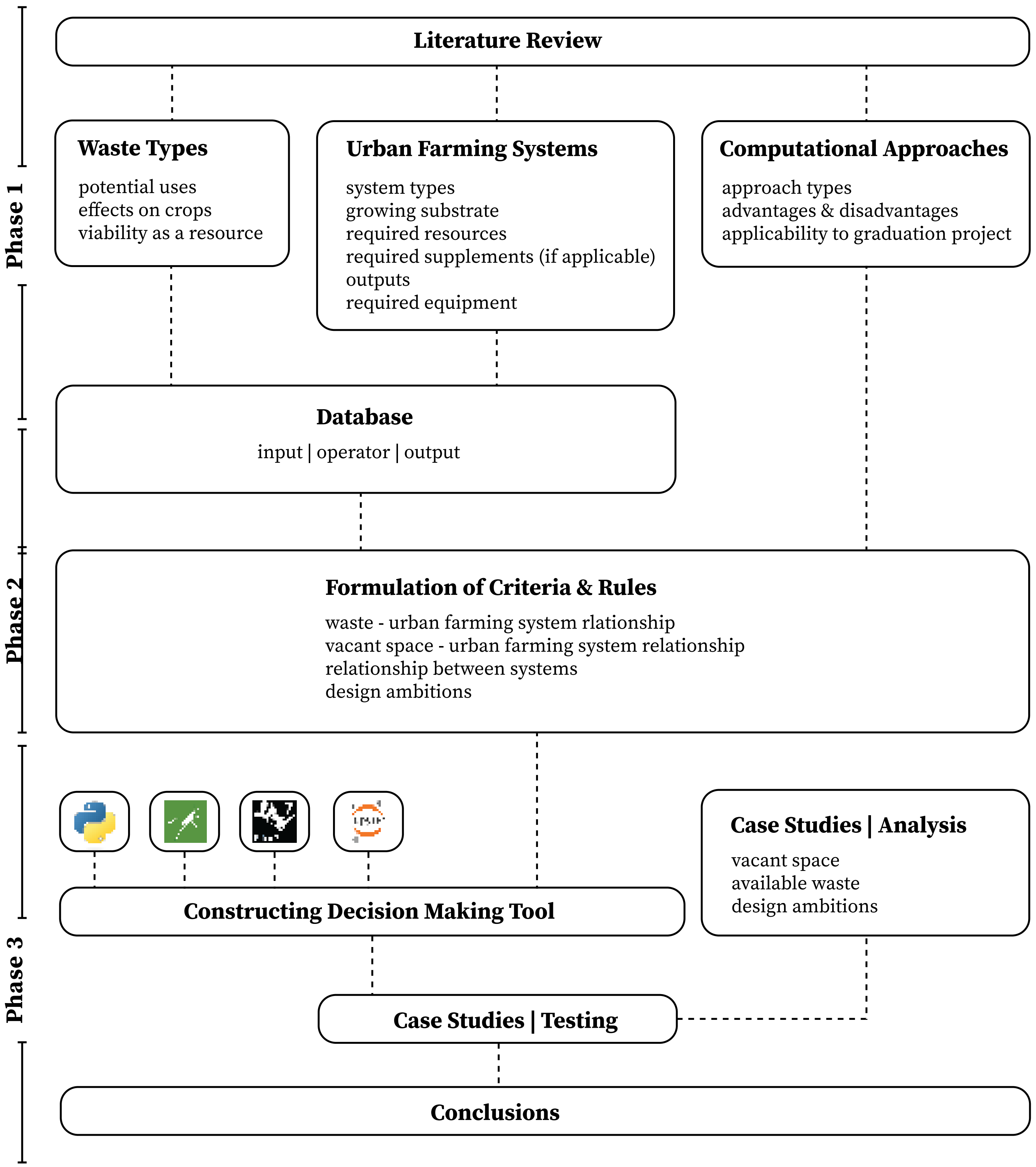
Formulation of Criteria & Rules: Following the literature review, the aforementioned database of urban farming systems, inputs, and outputs is formed to guide the formulation of rules. Criteria and rules are formed in such a way that an urban farming system will be suggested by the decision-making tool based on what kinds of waste flows and vacant spaces are present. These rules also encompass other characteristics of the spaces such as solar exposure and structural capacity. Criteria and rules to decide on a system depending on existing waste flows, available space, and project goals are formulated in order to provide guidelines for decision making

Case Studies: To test and develop the decision-making tool 2 case studies are conducted. Case study locations are selected from temperate climates and each case study has a different scale in order to validate the feasibility of using the tool fro varying sizes of projects. The first and primary case study indicating the performance of the decision-making tool will be TU Delft Campus. The second case study will be the site of the Urban Greenhouse Challenge in Washington DC. Data regarding available space and existing waste potential will be collected. The decision-making tool will be tested on the aforementioned case studies to prove the validity of the tool and to highlight the potential of integrating urban farming into cities.

In order to develop the decision-making tool Rhino for 3D modelling and illustration of results visually, Python, Jupyter Notebook, and Grasshopper for implementing the decision-making criteria & rules are used. However, it should be noted that the logic and methodology behind decision making is not software specific, therefore the same methodology can be used with other softwares.

In Phase 1 literature review results are unveiled. There are 5 main topics investigated in the literature review: (2.1) Current Food System & Challenges, (2.2) Urban Farming, (2.3) Urban Farming Systems, (2.4) Waste Streams In The City, (2.5) Computational Approaches. The literature review will be followed by (2.6) Conclusions of Literature Review and (3) Outline of Design Task.

In Phase 2, consisting of decision-making criteria & rules of the decision-making tool is investigated. This is followed by Phase 3: Case Studies and Phase 4: Conclusions. These chapters are followed by reflection, bibliography and appendix.



Literature Review

Waste Types
potential uses
effects on crops
viability as a resource

Urban Farming Systems
system types
growing substrate
required resources
required supplements (if applicable)
outputs
required equipment

Computational Approaches
approach types
advantages & disadvantages
applicability to graduation project

Database
input | operator | output

Formulation of Criteria & Rules
waste - urban farming system rlationship
vacant space - urban farming system relationship
relationship between systems
design ambitions

Constructing Decision Making Tool

Case Studies | Analysis
vacant space
available waste
design ambitions

Case Studies | Testing

Conclusions

Figure 1.1 Research Methodology

Glossary

Urban Metabolism: Body of systems which consume and/or produce natural resources (including air, food, household waste) either in pure or used form, Collection of systems which control waste flows in the city

City: A portion of the city for example a neighbourhood, a campus

Waste flows: Stream of resources consist of water, air, heat, and organic(household) waste

Existing Conditions: including vacant space, available waste (air, household waste, heat, water)

Food Production Systems: Mushroom Production, NFT, Media Bed, Raised Bed, Water Culture, Plant Factory, Aeroponics
Systems which only produce food including mushrooms, soft fruits and leafy greens.

Food Producing Supplementary Systems: Aquaculture
Systems which produce supplements in addition to food.
Aquaculture produces fish (food) and nutrient-dense water (supplement for some systems)

Supplementary Systems: Vermiculture
Systems which only produce supplementary items but no food items.
Vermiculture produces fertiliser and worms (fish food) which are both supplements for some systems rather than human food sources.

Supplementary Items: These are resources which are produced by some uf systems and can be used by other uf systems for the sake of symbiotic design.

Critical Items: Resources which are a must for a system to function

Non Critical Items: Resources which can be substituted externally or are not a must for that system to function. These differ from supplementary items as these cannot be produced by uf systems.
Rainwater, CO2, Excess Heat

Non Transferable Items: CO2, Heat, Rainwater

These resources are only used if they are available in the same building as the vacant space. Infrastructure needs to be added for these resources to be transferred.

Search Radius: Search radius is the distance between each vacant space and waste sources around it.

Symbiosis Rate: This number defines how symbiotic the design is in terms of use of waste as a resource
Symbiosis Rate = (Found Resources/Needed Resources)

UF: Urban Farming

DM: Decision Making Tool

02

LITERATURE REVIEW

2. LITERATURE REVIEW

2.1 Current Food System & Challenges

In the current setup of urban environments, there is a range of issues with the design of the urban metabolism. Tsui et al. relate urban metabolisms to material and waste flows. (2021). Similarly, urban metabolism refers to a collection of systems that consume and/or produce natural resources either in a pure or used form in this paper. In urban metabolisms, there are critical issues such as public health, healthy food access, green space, air and water quality, economic development, and community engagement (Ackerman, 2012). Some of these issues can relate to the dependency, vulnerability, linearity of the system in addition to food security concerns.

Modern cities are dependent on the global food network for food distribution (Graamans, 2021), and rely on delta areas and rural parts of the countries for energy supplies and resources (Roggema & Keeffe, 2014). The dependency of cities on rural areas and global supply chains has a negative effect on the robustness and resilience of the system. Current food supply networks are resilient to random failures however the supply chains are fragile under cascading disruptions (Graamans, 2021). Climate change, rising temperatures, CO2 levels, precipitation patterns pose potential risks to food production (Fresco, 2009), in addition to weather extremes, epidemics, pandemics, agroterrorism, floods and decreasing air quality (Graamans, 2021). Therefore, fluctuations in food production can take place as a reaction to changing environmental factors as Fresco reports (2021). The risks include lower food quality, disruptions in sourcing, transporting, and distribution in addition to reduced access to produce (Graamans, 2021). Meanwhile, according to Roggema & Keeffe, a food crisis may happen if this vulnerable system faces a failure (2014). Vulnerability of food supply is not only a valid issue for developing countries, but it also appears as even a more urgent issue among lower-income groups who may not have the economic power or savings to withstand the effects of such failure (Fresco, 2009). Consequently, in order to ensure food security, alternative food suppliers can be introduced not to sustain the whole city but to implement additional options of suppliers.

Secondly, urban metabolisms are based on linear processes including consumption of energy, food, space, and production of waste and greenhouse gas emissions. Cities consume 60-80% of natural resources supplied by rural areas, produce 50% of waste which are either sent to landfills or incinerated and emit 75% of greenhouse gasses (Tsui et al., 2021). Moreover, according to Roggema & Keeffe, cities lost their place in sustainability as food is supplied globally from rural areas (2014). The linearity of consumption and production can be mitigated by introducing circular economy strategies including local/urban manufacturing, production of goods from local sources, recycling, repurposing, urban manufacturing to minimise dependency on global supply chains as Tsui asserts (2021). In conclusion, linear urban metabolisms can evolve into closed loops by introducing local suppliers and utilising the waste flows as resources.

Taking into consideration the aforementioned rising concerns and problems of the current food supply systems and urban metabolism there are necessary actions to take and solutions to deliver. Food systems should be redesigned to be adaptable to changing temperatures, water, nutrient conditions, new pests, and diseases in addition to being climate-resilient, flexible (Fresco, 2009). Since produce is imported from various locations and passed through many stakeholders on the way, 40% of losses occur after harvest during processing, marketing, and storage as discussed in the same paper. Possible solutions to previously mentioned problems include bringing agriculture and food production back into the urban environments and cradle to cradle type of approach which supports the utilisation of waste from an operation as a resource for another operation (Fresco, 2009). The same paper also suggests incorporating multifunctional uses of land and food systems including using recirculation and storage of water, energy, CO2, and heat storage in aquifers. Lehmann also advocates that it is crucial to link the function of cities and rural areas together to integrate natural and healthy food systems to the urban environments in which people garden and farm locally, compost agricultural and kitchen waste, and grow community vegetables (2011). To sum up, bringing agriculture back into the city is a possible solution proposed by several authors to battle risks of the food crisis, lower greenhouse gas emissions, ensure circularity within urban metabolism and decrease dependency on external factors and global supply chains. In the following chapter, urban farming, its environmental and social benefits as well as urban framing's potential to close the loop will be discussed.

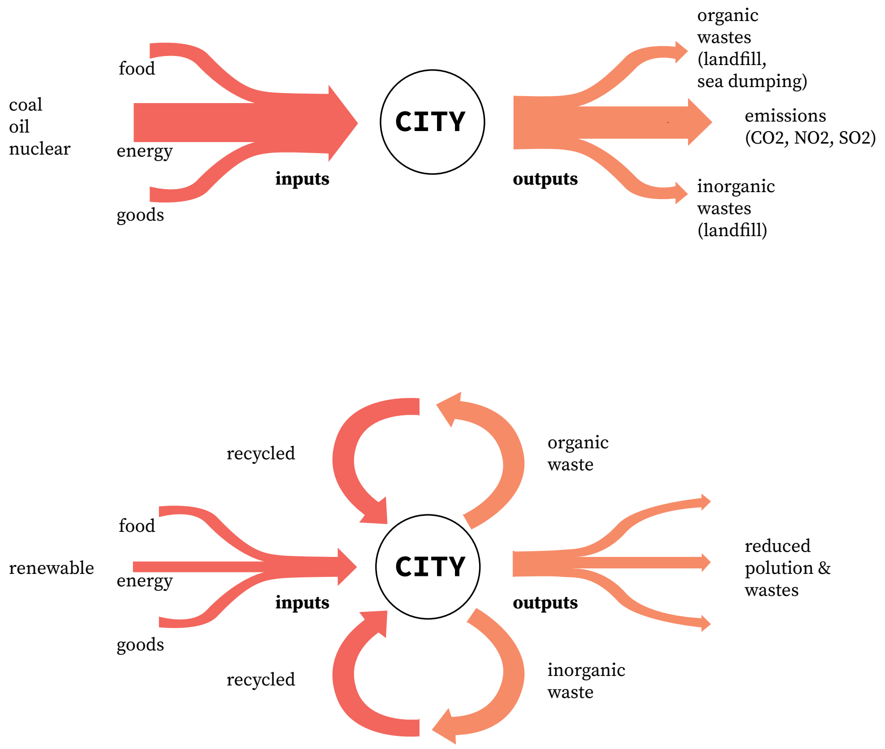


Figure 2.1 Linear Metabolisms of Cities (Top), Circular Metabolisms of Cities (Bottom)
(redrawn from "Cities for A Small Planet" by R.Rogers & P. Gumuchdjian, 1997, p.31)

2.2 Urban Farming

Urban farming refers to growing crops and livestock within urban environments. In this paper, the focus will be on the growing of crops in cities, or in other words the scope of the research is urban agriculture. In this section, urban agriculture will be discussed as a promising solution to the aforementioned challenges inclusive of its potential environmental and social benefits and its ability to close resource loops.

To begin with, urban agriculture has several social benefits including the potential to improve physical health, psychological well-being, financial security and community development in addition to ecological advantages. Integrating agriculture within the cities can improve access to healthy food, promote the consumption of vegetables, introduce green space and consequently improve air quality (Jenkins & Keeffe, 2017). From a psychological well-being point of view, the presence of natural features can assist stress relief, micro restoration of nearby inhabitants simply by being exposed to greenery, potentially reduce depression and anxiety according to the same paper. Jenkins & Keeffe state that urban agriculture has potential economic benefits such as providing new job opportunities, shortening the distribution chain, lowering food prices, and making use of vacant spaces (2017). Even though urban agriculture may not cover all or even the majority of the food demand of major cities, it can make a positive impact on food security (Ackerman, 2012). As asserted by the same author, urban agriculture is an attractive solution for some neighborhoods with low access to produce, low income, high obesity, and diabetes rates, and comparatively more vacant spaces (2012). The same article also discusses that urban agriculture can benefit society by enhancing community development by empowering the local stakeholders, transforming neglected spaces into public resources which provide opportunities for social interaction, promote self-sufficiency and engagement of young people in such neighbourhoods. To sum up, since urban agriculture has a potential to increase food security by introducing an additional supply in neglected urban areas (Ackerman, 2012), it is a promising solution to increase the quality of life in cities and it may overcome the challenges of future.

On the other hand, in addition to societal impact urban agriculture can make a significant environmental impact. Introducing agriculture to cities, reduces the food miles and consequently decreases the carbon footprint of food production (Jenkins & Keeffe, 2017). Another benefit of urban agriculture is making use of vacant sites in cities since more land will be needed to feed the global population with a healthy diet. Most urban areas have a considerable amount of vacant space for food production such as empty lots or rooftops (Ackerman, 2012). Technological advancements in food production can also be utilised in urban settings. For example, higher yields can be achieved by hydroponic systems in comparison to other growing methods (Rorabaugh et al., 2002 as cited in Ackerman, 2012). Yields can be ten times higher than field-grown crops depending on the crop types as mentioned in the same source. Therefore, high yield production which can be offered by advancing greenhouse technology is a possible solution to growing demands (Ackerman, 2012). In addition, with such advancements, food can be produced year-round and be supplied to the customers with freshness as stated in the same article. Furthermore, according to Lehmann, the concept of waste should be diminished by switching to a closed-loop system where waste is recycled, repurposed, composted (2011). Sustainable waste management should be adopted which corresponds to utilising waste as a resource (Lehmann, 2011). In addition to the direct environmental benefits, urban agriculture has the potential to close the loop in urban metabolisms including water, energy, waste flows if additional systems are utilised (Jenkins & Keeffe, 2017). Meanwhile, the energy balance can be optimised by introducing exchange, storage, and exergy principles as explained by Lehmann (2011). For instance, heat accumulated in the urban farms can be used to heat nearby buildings consequently energy sources can be decentralised as advised in the same article. In addition to heat, Jenkins & Keeffe suggest that rainwater can be harvested and wastewater can be utilized as fertilizer (2017). Waste flows in the urban metabolism consist of heat, food, organic waste, black water, rainwater will be discussed in section 2.4 along with each waste flow's potential to be integrated into urban farming to close the loop.

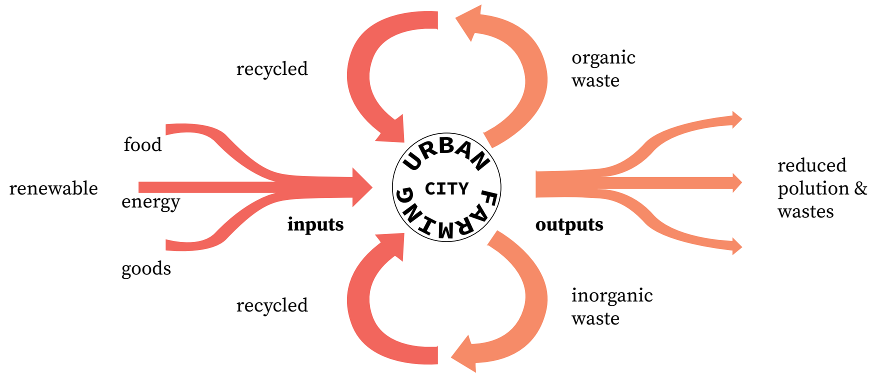


Figure 2.2 Achieving Circular Metabolisms Through Adopting Urban Farming
(adapted from "Cities for A Small Planet" by R.Rogers & P. Gumuchdjian, 1997, p.31)

2.3 Urban Farming Systems

There are various urban agriculture systems including soil-based agriculture, water-based agriculture, mushroom farms, vermiculture and different soilless agriculture techniques including NFT, deep water culture, media beds and plant factories. In this section, each system and technique will be described along with their advantages, disadvantages and some examples.

2.3.1 SOIL-BASED AGRICULTURE

Soil-based agriculture inherently utilises soil as growing a medium. Soil is essentially a mixture consisting of decomposed organic matter, nutrients, minerals, and microorganisms that promote plant growth (Jenkins, 2018). However, nutrients do not remain within the soil and over time soil can be deprived of these nutrients due to excessive agricultural activity (Jenkins, 2018). Therefore it is crucial for agriculture to replenish the composition of soil by supplementing it with nitrogen, phosphorus and potassium as stated in the same paper. Even though soil-based agriculture is a conventional method of growing plants, integrating it into the cityscape is troublesome due to several factors. Firstly, in many cities soil is contaminated due to previous industrial activity, bleaching fields and chemical dumps (Jenkins, 2018). Even though industrial activities usually do not take place within the cities anymore, Jenkins states that soil is exposed to atmospheric transportation of contaminants due to vehicle emissions, industrial discharge and waste incineration (2018). Consequently, if the soil is to be selected as a growing medium in the city, contaminant levels should be studied to see whether they are within the safety threshold provided by the government (Jenkins, 2018). Furthermore, technically soil-based agriculture can be an approach to urban farming if the soil is not contaminated or raised beds are utilised as a growing technique however available space on ground level in cities is limited (Jenkins, 2018). Rooftops or vacant spaces within the buildings can be an alternative approach however soil-based agriculture may result in higher dead loads than what the existing structure is actually capable of supporting safely according to the same author (2018). In conclusion, soil-based agriculture may not be the optimal solution to integrate agriculture into urban environments however raised beds will remain as an option for the sake of developing the decision-making tool as they have the potential to be utilised in some cases.

2.3.2 WATER-BASED AGRICULTURE

An alternative to soil-based agriculture is water-based agriculture which is a soilless growing method. This type of food production is carried out by hybridised food systems which utilise technical products including mechanical pumps, glass, plastic to grow crops in nutrient-rich water (Jenkins, 2018). Water-based agriculture has advantages over soil-based growing methods. These advantages include shorter harvest cycles, more productive food production systems, 4-10 times higher crop yields and maximised production (Ackerman, 2012; Jenkins, 2018). In such systems, roots of the crops are in direct contact with a nutrient-dense solution either by adding the solution into an inert medium such as gravel, sand, vermiculite, perlite, coco coir, by suspending the roots into the solution or by spraying the roots with nutrient-dense solution periodically as explained in the same paper. In this method, the nutrients are delivered to plants' roots through water (Jenkins, 2018). According to Jenkins, crops' roots are directly in contact with nutrient solution therefore crops do not need to exert energy to acquire nutrients and instead they can produce mass resulting in higher productivity of systems (2018). In addition, as asserted in the same paper, with such systems the need to replace the soil in order to maintain the soil structure, fertility and to diminish the negative effects of pests is eliminated. With water-based agriculture systems lettuce, radishes, carrots, potatoes, flowers can be grown however some specific systems are more suitable for growing certain crops due to varying root structures (Jenkins, 2018). Moreover, another advantage of water-based agriculture is that it can be utilised in non-arable regions (Jenkins, 2018) which makes it a preferable approach to growing crops in urban environments.

Hydroponics

Hydroponics is a system consisting of a water reservoir, water pump and plants in a tray which are supplied with mineral salts added to water automatically or manually (Jenkins, 2018). Nutrient-rich water is pumped to the crops from the reservoir, then water falls back to the reservoir due to gravity and this cycle is repeated until the nutrients in the water are depleted as described in the same paper. In order to replenish the nutrient levels in the reservoir, nutrients are supplemented regularly as stated by Jenkins (2018). These systems are lightweight in comparison to soil-based agriculture therefore implementation of water-based agriculture into the urban fabric is comparatively more realistic and achievable as discussed in the same source. Even though water within the system needs to be discarded regularly along with unutilised nutrients, these systems use less water than soil-based growing techniques as explained by the same author. Moreover, since water-based agriculture takes place in a protective environment, it is resilient to climate conditions consequently these systems impact food security positively (Jenkins, 2018). On the other hand, even though hydroponics have superior benefits compared to traditional soil-based agriculture, some downsides are also present. Firstly, it is dependent on depleting fossil fuels for nutrients' sourcing while nutrient and water use is inefficient as water is discarded along with unutilised nutrients (Jenkins, 2018). Besides, human intervention is necessary for a stable system growing crops continuously as remarked in the same paper. Even though there are disadvantages to such a system, its advantages over soil-based agriculture make hydroponics a viable approach to grow food in cities.

Aquaponics

Another water-based food production method is aquaponics. In aquaponic systems, similar types of equipment to hydroponic systems are used however instead of a reservoir a fish tank is introduced to the system in addition to a filter (Jenkins, 2018). According to Jenkins, such a system benefits from the symbiosis between fish, bacteria and plants consequently it is a holistic approach to food production that is dependent on the naturally occurring nitrogen cycle (2018). The same paper notes that this symbiosis is initiated by waste ammonia that is produced by fish. This filtration system enables waste produced by fish to be utilised by crops as nutrients since nitrate is a form of nitrogen that can be absorbed by plants as explained in the same source. Conversion of fish solid wastes to nutrients is a slow process therefore these solids should be separated from the water supply (Jenkins, 2018). After the removal of solids, they can be repurposed as fertilisers for soil-based agriculture purposes as stated by the same author (2018). Furthermore, inputs to the system include fish food and calcium (Jenkins, 2018). Similar to any other system, aquaponics has advantages and disadvantages which will be explained next.

Aquaponics has many benefits environmentally, financially and from a food production perspective. Thanks to the symbiosis between fish and bacteria, well being of fish increases, bacteria populations thrive, excellent water quality is achieved and crops grow with high yields (Jenkins, 2018). Moreover, since the nutrient demands of crops to grow are supplied by the fish, such a system is not dependent on fossil fuels or manmade expensive fertilisers resulting in lower operational costs according to the same source. Aquaponic systems are 10-15% more productive than hydroponic systems and with the same costs, they can produce 17 kg tilapia and 8 tomato plants while hydroponics can produce only a few tomato plants as reported in the same article. In addition to higher yields, these systems are less susceptible to developing diseases (Jenkins, 2018). From an environmental point of view, water is used more efficiently as it does not have to be discarded typically, less water is used in total even though initial water demand is high to supply the fish tank and water is only lost due to evaporation and transpiration which can be overcome by minimising the water-air contact or by condensing transpiration as discussed in the same article. Another advantage of aquaponics is that human interaction is minimised as a result of benefiting from natural cycles (Jenkins, 2018). Aquaponics can take place in indoor, closed environments just like hydroponics which eliminates the potential harm climate conditions can have on food production as discussed in the same source. In terms of viability to introduce into urban environments, it is an applicable method to grow food in cities since it is considered as a lightweight system (Jenkins, 2018). Nevertheless, aquaponics has some disadvantages, some of which can be overcome. Firstly, it is a heavier system compared to hydroponics however as Jenkins demonstrated the fish tank and crop growth can take place in different locations within the building (2018). Secondly, water becomes acidic over time and to overcome this issue it can be supplemented with naturally occurring pH buffers such as eggshells and chalk (Jenkins, 2018). Thirdly, fish should be supplied with fish food which is usually produced industrially and dependent on fossil fuels as mentioned previously, however it can be also supplied by vermiculture according to Jenkins (2018) which will be explained further in section 2.3.5. Lastly, aquaponics is a more complex system than hydroponics (Jenkins, 2018) however its benefits once it is set up is inevitable as previously discussed. Therefore aquaponics remains as a preferable urban farming method for the development of this research.

2.3.3.Soilless Agriculture Techniques

Hydroponics and aquaponics are soilless agriculture methods however their nutrient delivery systems can be designed in 4 different ways for both systems including NFT, water culture, media beds and aeroponics. Each growing technique will be explained in this section with its advantages and disadvantages.

NFT

With Nutrient Film Technique (NFT), crops are placed in a channel that supplies a thin layer of nutrient-rich water to the roots of the crops (Jenkins, 2018). The design of the system can be done in different ways. One approach is a cascading system in which slightly inclined channels are stacked to grow low leaf crops like lettuce, radish and peas (Jenkins, 2018). In this system, nutrient-dense water is pumped to the highest end of the top channel and it exits at the lowest point of the top channel with gravity to enter the channel below and circulate until the lowest point of the system to reach the reservoir as described in the same paper. Another design is an A-frame which is oriented in the east-west direction to make better use of space and minimise overshadowing (Jenkins, 2018). Both the cascading system and A-frames are suitable for growing lettuce, strawberry, spinach and herbs as noted in the same source. These systems are scalable however there are dimensioning limitations to adhere to (see appendix A.8 for details). Another limitation of such a system is that only quick-growing short term crops can be grown as long term crops with larger root systems can block the nutrient flow within the channels (Jenkins, 2018). Advantages of NFT include flexibility to grow food horizontally or vertically either on rooftops or on facades according to the same paper. It is considerably lightweight compared to soil-based systems therefore it is potentially suitable for structures with lower load-bearing capacities (Jenkins, 2018). Lastly, according to Jenkins, the channels can be covered with plates with holes in them to minimise water loss due to evaporation (2018). To sum up, due to the lightweight of the system and potential efficient use of space NFT is a viable growing technique for smaller crops in urban environments

Water Culture

Water culture is another growing technique which utilises a larger body of water than the nutrient film technique to grow plants. In such a system, the roots of the crops are suspended in nutrient-rich water about 300mm, while the plants are supported with a floating raft above water (Jenkins, 2018). Nutrient-rich water is pumped from one end of the raceway, moves along the raceway to deliver necessary nutrients for crop growth, and exits the system at the end of the raceway to the reservoir as described in the same paper.

Similar to the other growing techniques, water culture has advantages, disadvantages and limitations. To begin with, advantages of water culture include easy maintenance and handling since when crops are ready to be harvested, they can be removed from one end and new crops can be placed from the other end of the raceway creating an effective conveyor belt-like production system (Jenkins, 2018). Secondly, even if one raceway fails due to any kind of complication the rest would keep functioning optimally therefore the system’s resilience is high as stated in the same source. In addition, according to the same author the water can be heated or kept cool to improve the productivity of food production depending on the climate (2018). Similar to NFT, water culture is suitable for growing crops with smaller root structures like lettuce, spinach, herbs as explained by the same author. On the other hand, such a system consisting of a large body of water is comparatively heavier than NFT therefore it is most suitable to be used on ground level (Jenkins, 2018). However, depending on the context, water culture may have the potential to be integrated into urban fabric therefore this technique will remain as an option while developing the decision-making tool.

Media Beds

Another alternative soilless agriculture technique is media beds. This system utilises a type of substrate to grow crops and allow for growing plants with bigger root structures (Jenkins, 2018). There are a variety of media that can be utilised (See appendix A.10 for details). These systems are scalable therefore the bed can be any size but a minimum of 300mm depth is needed (Jenkins, 2018). An advantage of media beds is that utilisation of media allows for root anchorage and growing bigger plants without having a heavy soil-based growing system according to the same author. However, media beds are heavier than systems utilising NFT (Jenkins, 2018), therefore consideration should be given when or where to use this system.

There are two types of delivery systems to supply nutrients: ebb & flow and gravity-based trickle systems. Within a media bed with an ebb & flow system, media is periodically flooded and drained, consequently, each piece of media is submerged and supplied with nutrient solution (Jenkins, 2018). On the other hand, in a gravity-based trickle system, as Jenkins describes, there are multiple water outlets at the surface, and nutrient solution is continuously dripped (2018). This is a simpler method than ebb & flow however it does not ensure supplying each particle with nutrient-rich solution (Jenkins, 2018). Even though there are disadvantages to the system and to both nutrient delivery techniques, bigger crops like tomatoes can be grown with media beds. Since growing bigger crops with NFT or water culture is not feasible, media beds will be included in the decision-making tool to cover demands for growing bigger crops.

Aeroponics

The last system to discuss within the soilless growing techniques chapter is aeroponics. This system consists of a watertight box which plants are placed in and supported, equipped with a high-pressure nozzle under each plant to spray nutrient-rich mist to the roots directly (Jenkins, 2018). Aeroponics can be set up both horizontally and vertically as stated in the same article. Water is lost only through transpiration as a result of a watertight outer shell supporting the crown of plants according to the same author (2018). Such a system is not commercially used and most common for labs even though it has low water and energy use as stated in the same source. Even though commercial examples of an aeroponic system are limited, this technique will be included in the further steps of the thesis for the sake of constructing a universal decision-making tool.

Plant Factories

Another common approach to growing crops indoors is plant factories. Plant factories are not techniques or methods of food production. They are indoor production setups with controlled, sterile environments (Jenkins, 2018). The environmental controls include air composition, humidity, ventilation, air temperature, nutrient solution, light intensity as reported in the same paper. The use of artificial lighting allows the growing trays to be stacked with high-density production in mind (Jenkins, 2018). In plant factories using water culture hydroponics with shallow trays is common according to the same author.

There are significant advantages of a plant factory. The nutritional value of crops can be increased as a result of controlled environment (Graamans, 2021) Water use efficiency is significantly higher in closed systems due to two main reasons: Firstly, productivity is increased due to climate control with higher CO2 concentrations as Graamans stated (2021). Secondly, it is possible to collect and reuse transpired water condensing at the cooling element as asserted in the same source. For instance, in a glass greenhouse with moderate climate control, 3.2 MJ kg-1 of energy and 9.3 l kg-1 of water are needed for the production of lettuce meanwhile in an open field in Italy the numbers are 2.9 MJ kg-1 and 24.0 l kg-1 respectively as reported in the same source. Jenkins adds that plant factories are not dependent on climate or external factors as they are indoors with high climate control and produce high-quality crops with high hygiene levels, low bacterial growth which extends the shelf life (2018). In the same source, it is mentioned that crops smaller than 300 mm are preferable in such settings to maximise vertical production. As an advantage of utilising artificial lighting, there is no location limitation for plant factories since natural light is not a determining factor (Jenkins, 2018). Although plant factories have many advantages over less controlled food production environments; their dependency on equipment such as air conditioning units, air, circulation fans, CO2 supply units, nutrient solutions supply units, environmental control units and artificial lighting is considered as a disadvantage by Jenkins since all these systems consume energy in addition to high capital investments due to air handling units and artificial lighting (2018). In addition, plant factories are referred to as a less holistic approach for food production compared to other systems in the same paper. Even though there is no location limitation for building plant factories, stacked shelves cause higher loads to be carried by the structure (Jenkins, 2018), consequently, whether it is feasible to integrate plant factories into or on existing buildings is questionable. However, for the sake of research purposes, plant factories will be included in the decision-making tool as an alternative in case the conditions require such a system.

2.3.4 Mushroom Farm

Mushrooms are another product urban farms can produce and they can be cultivated in waste products. Many Fungi are naturally capable of cycling organic matter and nutrients by decomposing waste, and as a result, they produce edible mushrooms (Dorr et al., 2021). This organic waste is converted into a nutrient-rich matter known as spent mushroom substrate which can be used as fertilizer for soil-based agriculture. (Dorr et al., 2021). There are 4 main components of mushroom cultivation such as preparing the substrate, incubation, fruiting and harvesting (see appendix A.3 for details).

Firstly, preparing the substrate includes mixing the substrate and sterilising it (Dorr et al., 2021; GroCycle, 2021). Mushrooms can be grown in various waste-based substrates. The substrate options include manure, spent coffee grounds, wood chips (Dorr et al., 2021), straw, pellets, cardboard, sugar cane, paper, coconut husk (GroCycle, 2021) depending on the variety of mushrooms. For instance, oyster mushrooms (Pleurotus ostreatus), can be grown on some waste materials such as grape marc from wineries, waste from olive oil mills and coffee ground waste from brewing (Dorr et al., 2021). Some of the mentioned waste substrates are easily accessible in urban conditions such as cardboard, paper, coffee grounds and wood chips, therefore these substrates will be included as potential resources in the later stages of research. The mycelium and substrate mixture is pasteurised and placed into plastic bags (Dorr et al.) or in reusable buckets (GroCycle, 2021) for incubation. Incubation should be done in a dark environment since light triggers mushrooms to fruit before it is required as explained in the same source (see appendix A.3 for details). One way of fruiting mushrooms is cutting holes in the bag or bucket and spraying them daily for high humidity levels as described in the same source as shown in figures 2.3 and 2.4. The last step is harvesting the mushrooms once they are ready to be picked. Mushrooms can be harvested several times from the same mixture (Dorr et al., 2021). Due to mushrooms' capability to grow in waste substrates and due to the fact that there are examples of such practices mushroom production is a potential urban farming system.

One real-life example of using vacant spaces is mushroom farms started by Cycloponics. They specialise in building farms in unused spaces in cities including an abandoned underground parking lot in Paris called Cave. The same company also built urban farms in cellars of old buildings and a former bunker built in 1878 by the Germans. Another example is Smallhold which aims to grow everywhere possible by providing micro-farming solutions in addition to large scale urban farms. Project locations include warehouses, restaurants, supermarkets where the mushroom farm is placed above the aisles or the bar.

To sum up, mushrooms can be cultivated in several waste-based substrates in dark environments which can be found in the existing urban fabric. Besides, there is a range of potential waste sources present in cities which can be used as substrates. Therefore for this research, integrating mushroom farms by utilising waste as a resource is a viable option and will be further investigated in section 2.4.3 with potential wastes and their applications.

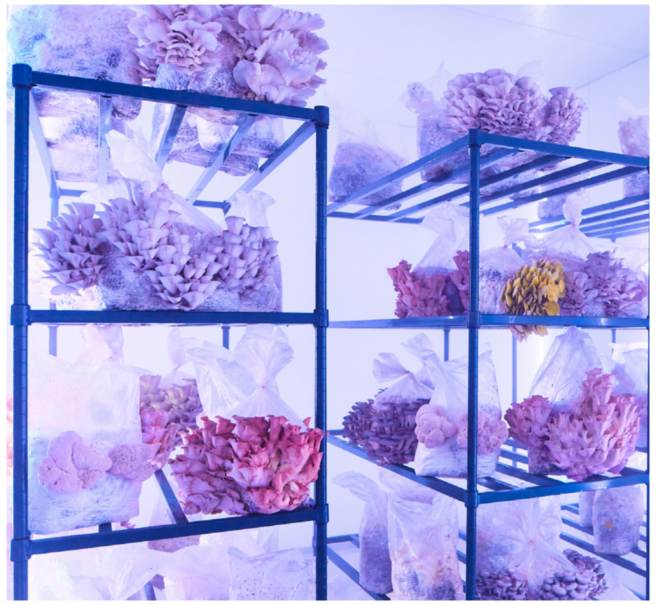


Figure 2.3 Mushroom Cultivation in Bags by Smallhold
Smallhold. (n.d.)from https://www.smallhold.com/about

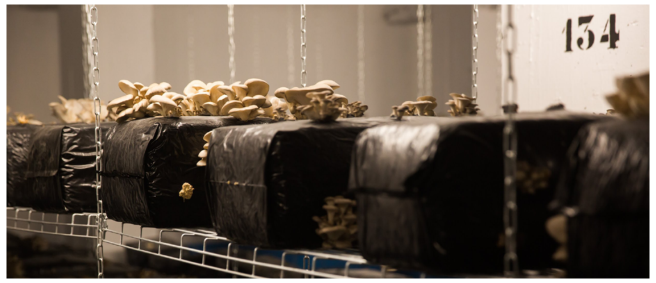


Figure 2.4 Mushroom Cultivation in Bags by Cycloponics
The Cave in Paris. (n.d.). Cycloponics. from http://cycloponics.co/galerie/

2.3.5 Vermicomposting

Vermicomposting is a method that naturally decomposes waste as earthworms and microorganisms mineralise organic waste and convert them into nutrient-rich organic matter (Sharma & Garg, 2019). The outputs of the system are vermicompost and earthworms according to the same paper. Vermicompost can be used as a soil conditioner with many advantages over chemical fertilisers as it will be discussed in the next paragraph. The second product of vermicompost, earthworms, can be used for medicine, or as fish food (Sharma & Garg, 2019; Jenkins, 2018). Sharma & Garg notes that vermicomposting takes place in an aerobic environment and is dependent on the symbiotic relationship between earthworms and organisms (2019). Metabolic activities that earthworms perform, turn the waste mixture into vermicast and then to vermicompost which is a nutrient-rich product as stated in the same article. Important environmental factors for vermicomposting include moisture, temperature, pH range and Carbon/Nitrogen ratio (See appendix A.2 for other factors). According to Sharma & Garg, feeding rate and growing density are also influential factors for earthworm growth and consequently for vermicomposting (2019).

Moreover, vermicomposting has many advantages and benefits including producing fertilizers as a byproduct, simply benefiting from natural cycles between earthworms and microorganisms and utilising waste. According to Sharma & Garg, it is an eco-friendly and zero waste management method, which consumes less energy, produces less GHG at lower costs compared to traditional compost, and reduces pathogen levels in waste (2019). In addition, Abbasi et al. report that vermicomposting is a clean process by nature that requires less energy and material than other biotreatment methods and most importantly all the nutrients in the waste mixture returns to soil (2015). Vermicomposting can take place with a range of organic waste types however for this research kitchen waste, paper, food waste (Sharma & Garg, 2019) will be taken into account as resources. Another benefit of vermicomposting is the vermicompost it produces which has more available nutrients than the initial waste mixture (Garg et al., 2006) and when used as fertilizer influences plant growth positively (Sharma & Garg, 2019). When compared to chemical fertilisers and compost, plants gave the highest yields when aided with vermicompost (Yang et al.,2015 as cited in Sharma & Garg, 2019). Vermicompost has the potential to be used in nearby soil-based urban farms or to be sold local gardens and small scale horticulture operations.

As discussed previously, vermicomposting is an effective, low-cost method for waste management that has secondary benefits such as producing high-quality fertilizer and fish food. In addition waste types which can be a substrate to vermicomposting such as paper, kitchen waste can be easily accessed in urban areas while agriculture waste can be provided by nearby urban farms potentially. Moreover, extra earthworms can be used as fish food for a nearby aquaculture system. In conclusion, vermicompost will be included in the later stages of research and design as a viable supplementary method to be integrated into urban metabolisms.

As a conclusion of the literature review on different urban farming systems, vermicomposting, mushroom production, NFTs, water culture, media beds, aeroponics, raised beds and plant factories are included as urban farming systems in the later stages of this research.
Waste Streams In the City

2.4 Waste Flows in Cities

Cities consume food and natural resources supplied by rural areas and after consumption transfer their waste out of the city for waste management purposes including incineration, landfills and waste management facilities. Therefore city metabolisms are based on linear processes. As Tsui et al. report linearity of consumption and production can be replaced by circular economy strategies by repurposing waste (2021). Waste produced within the city can be repurposed to feed urban farms in order to contribute to the circular economy. Some of the waste types produced in the city include CO2, black water, organic waste, residual heat if not stored or repurposed and rainwater if not collected. In this chapter different waste types and their potentials to be used as resources for urban farming will be discussed and conclusions will be derived regarding the suitability of each waste type to integrate into urban farming.

2.4.1 CO2

Carbon dioxide is a greenhouse gas that is emitted by people, burning fossil fuels and industrial activities. More than 80% of global greenhouse gas emissions need to be cut to keep the temperature rise under 2C by 2050 while lands should be used efficiently to compensate for the growing food demand (Bao et al., 2018). Integrating urban farming into urban metabolisms by utilising vacant spaces, has the potential to tackle both of these problems since it increases the land-use efficiency, shortens the transportation path food takes. As Bao et al. report CO2 is the most important carbon source for plant growth via photosynthesis and is a limiting factor to plant growth in indoor farms with sufficient light, water and nutrients (2018). Every square meter of an indoor farm can sequester 15 times more CO2 from the atmosphere than in an open field and 50 times more than a forest according to the same source. Therefore urban farms can be designed to be exchange hubs of CO2 and can lower carbon emissions.

Higher CO2 concentrations in urban farms have advantages in terms of productivity. Increasing CO2 concentration from 400 to 1000 ppm influences plant growth rate and increases the yield of flowers and vegetables by 21-61% (Bao et al., 2018). According to the same authors, Carbon Dioxide enrichment is a common practice in agricultural activities to increase yields and productivity, shortening growth time (2018). Carbon enrichment can be supplied by a range of sources. Most commonly CO2 concentrations are increased by burning fossil fuels such as natural gas, directly from CO2 tanks, flue gas from heating systems and liquid CO2 (Bao et al., 2018; Blom et al., 2012). During summertime plants grow robustly therefore CO2 demands are high while heating demands are low (Bao et al., 2018) resulting in inefficient use of natural gas.

Moreover, indoor CO2 concentrations should be monitored carefully for growth productivity and human health as CO2 concentrations above 5000 ppm cause dizziness (Blom et al., 2012). Concentrations ranging between 400-1000 ppm stimulates plant growth (Bao et al., 2018), after exceeding 1000 ppm photosynthesis plateaus (Blom et al., 2012) while in a tightly clad greenhouse with little or no ventilation, concentration can drop below 200 ppm which significantly hinders plant growth (Blom et al., 2012). In another article, it is reported that supplemented CO2 concentrations should be 600-1000 ppm for leafy vegetables and 1000-1500 ppm for fruit vegetables under sufficient light, humidity and temperature conditions (Xin et al., 2015). The same authors report that carbon enrichment has other benefits such as increased vitamin and sugar content in fruit crops; better quality and appearance of crops; higher resistance to diseases and pests; and longer harvesting periods (2015). The research results from 166 articles revealed that yield increase of 93.37%, enhanced resistance to disease and pests by 41.57% and advanced maturity period by 48.80% with fruit vegetables such as cucumber, tomato, chili, zucchini, eggplant and strawberry (Xin et al., 2015). CO2 enrichment has many advantages and a potential to be integrated into urban farming however CO2 sources used in conventional farms do not augment circularity and due to fossil fuel use, it is not sustainable in the long term. Nevertheless, there is an alternative source for CO2 to supplement urban farms which is readily available within urban environments due to respiration. In public assembly rooms, CO2 concentrations can reach 2500 ppm (4500 mg/m3) (Persily & Polidoro, 2020). According to ASHREA standards, indoor CO2 levels should not exceed outdoor levels by more than 600 ppm, while outdoor concentration is 380 ppm in most cases (Prill, 2000). In addition, CO2 levels should be kept below 1000 ppm (1800 mg/m3) in schools and 800 ppm (1440 mg/m3) in offices (Prill, 2000). Therefore such places with high CO2 concentrations should be ventilated for indoor air quality, usually, the ventilation output is exhausted to outside. However, it can be used for carbon enrichment of the urban farm as exemplified by ICTA RTG Lab. In this example, air from the laboratories which contain heat and high CO2 concentration is fed into the Rooftop Garden (RTG) (Sanyé-Mengual et al., 2014). RTG Lab is designed to use CO2 concentration in the residual air from offices and laboratories as carbon enrichment and the flows will be monitored by sensors as explained in the same paper. To conclude, as increasing CO2 levels in urban farms has positive effects on the crops CO2 will be treated as potential waste which can be sourced from crowded indoor spaces within the site to be used as a nutrient source for the crops.

2.4.2 Water

Water is another urban waste flow that will be investigated to be used as a resource for urban farms. In this section, rainwater, blackwater and their potentials to be integrated into farming will be discussed. As pathogens can be ingested by consumption of raw crops irrigated with contaminated water such as rainwater and reclaimed water (Ortells, 2015), the risks and strategies will be illustrated.

Rainwater

To begin with, rainwater provides an essentially clean water source and has environmental benefits over utilising water supplied by municipalities. In this section, its benefits, risks and strategies to overcome potential risks. Water is essential for agriculture however considering growing water scarcity alternative solutions should be developed. Water scarcity is a growing global problem that affects the world including the North Western Hemisphere as well as arid North African and Middle Eastern countries (Lundy et al., 2018). The same author reports that rainwater use is widely considered as a viable and sustainable practice to conserve water resources (2018). In another source, it is added that rainwater harvesting can contribute to sustainability in agricultural practices and increase resilience against drought (Dile et al. 2013 as cited in Macias-Corral & Sanchez-Cohen, 2019). Rainwater has been used for irrigation for a long time and it has advantages such as being nearly sodium-free, soft water, containing certain crop nutrients thus reducing the need for fertilisers, being decentralised and off-grid water supply, better stormwater management, reducing water runoff (Deng, 2021). On the contrary, pollutants in the water challenge the practice as they may pose risks to crops, farmers' and consumers' well-being as asserted in the same paper. The quality of water collected from roofs is worse than the quality of rainwater itself as Hofman-Caris et al. states (2018). Rainwater quality is affected by the material, age and design of the catchment surface, geographic location, precipitation patterns, pollutant loadings and storage conditions (Deng, 2021). According to Campisano et al., several studies illustrated that with the first flush majority of contaminants accumulated on the catchment surface are washed off during the beginning of rainfall (2017). Therefore first flushing can be practised to improve the quality of harvested water. It can also be improved in the storage tank by increasing pH, sedimentation of particles and precipitation of heavy metals (Campisano et al., 2017). In the same article, it is stated that these treatment processes can improve the quality significantly and lead to compliance with potable water standards. Opaque tanks which are periodically maintained should be used for storage to battle algal growth (Deng, 2021). Moreover, debris capture with filters and post storage treatments can be utilised to improve the quality (Campisano et al., 2017). According to the same authors, regular maintenance such as cleaning the catchment surface, gutters, storage tank, filters, first flush diverters, debris screens significantly improve water quality (2017). Since there are possible strategies to ensure the safety of utilising rainwater for irrigation, it remains as a viable method.

Blackwater

Another waste flow within urban environments is blackwater however there are potential risks of irrigating crops with reclaimed water. Human exposure can take place through the consumption of crops or occupational exposure (Ortells, 2015). In developing countries wastewater is used for irrigation of crops due to water scarcity and economic reasons while in developed countries it is seen as an environmentally sustainable, economical practice in addition to being a way to battle water scarcity as explained in the same paper. According to Ortells, even though wastewater is treated by secondary or tertiary wastewater treatment, it can contain pathogens (2015). Ortells investigated the risks of consuming lettuce irrigated with secondary and tertiary treated wastewater containing norovirus in Spain (2015). Results showed that lettuce internalises norovirus and the virus can reach the edible parts of the crop under laboratory conditions (Ortells, 2015). However, as reported by the same author risks are not known for using reclaimed water for irrigation in open field conditions consequently further research is needed (2018). Meanwhile, The German Federal Institute for Risk Assessment (BFR) states that only drinking quality water should be used for the irrigation of crops which are consumed raw (2020). In conclusion, black water is decided to be left outside of this project's scope as it may pose risks to human health by consumption or occupational exposure.

2.4.3 Organic Waste

Food and Agricultural Waste

Food is supplied to cities from outside as discussed before and leaves the city to be discarded in the form of food waste if not consumed. Food waste consists of any food and inedible parts of food which are removed from the supply chain to be recovered or disposed of (Ostergren et al., 2014 as cited in Pharino, 2021). In the European Union, food service generated almost 11 million tons of food waste including both edible and inedible parts during 2021 (FUSIONS, 2016 as cited in Pharino, 2021). According to a FAO estimate one third of food produced is lost or wasted globally (Muneer & Narula, 2021). This waste can be repurposed by utilising it as a resource for some urban farming systems to achieve circularity as opposed to linear processes.

Different kinds of food waste can be suitable for different purposes including vermicomposting and mushroom production. Kitchen and food waste are nutrient-rich, readily non-toxic and biodegradable waste types suitable for vermicomposting (Sharma & Garg, 2019). In addition, agricultural waste such as crop residues, leaf litter, sawdust can be vermicomposted according to the same paper. Sharma & Grag states that fruit and vegetable waste, crop residues and agricultural waste can be used as bulking substrates to balance the carbon content (2019). As discussed in Section 2.2, vermicomposting has many advantages including fish food and fertiliser production therefore utilizing food waste as a resource has the potential to feed fish, provide nutrients for soil-based agriculture as well as waste management.

Spent Coffee Grounds

Coffee waste is another waste flow present in urban metabolisms. As a result of coffee consumption 6 million tons of spent coffee grounds are generated globally in a year (Machado, Rodriguez-Jasso, Teixeira, & Mussatto, 2012 as cited in Mirón-Mérida et al., 2021). According to several sources, using spent coffee grounds is a quick and inexpensive way of growing mushrooms (Mirón-Mérida et al., 2021; GroCycle, 2021). With spent coffee grounds oyster mushrooms can be grown (Dorr et al.). As exemplified with a mushroom farm in Paris, 30 tonnes of spent coffee grounds are collected from nearby supply points such as cafes in 2018 and spent mushroom substrate is sold to farmers as fertilisers (Dorr et al.). The same authors report that, on the farm, the substrate is made from spent coffee grounds with wood chips, agricultural lime and mycelium (Dorr et al., 2021). According to the same source, 8728 kg of mushrooms were harvested only in 2018. Spent coffee grounds can be collected from nearby cafes, coffee vending machines, restaurants to be used as a substrate in urban mushroom farms and this approach will be incorporated into urban farming systems in later stages of the project.

Other Waste (Paper, Cardboard, Sawdust)

There are other waste flows in an urban setting including paper waste, cardboard, and potentially sawdust. These waste products can be utilized for mushroom cultivation and vermicomposting. Paper waste can be used as a bulking substrate for vermicompost to balance carbon content (Sharma & Garg, 2019), in addition, can be utilised in mushroom cultivation substrate (Sayner, 2020). For mushroom substrates, even though there are a wide range of waste options, readily available fibrous materials rich in carbon, lignin and cellulose with low nitrogen content should be favoured (Sayner, 2020). The material selection includes paper, cardboard and sawdust (Sayner, 2020; Dorr et al., 2021). In the previously mentioned mushroom farm in Paris, sawdust is mixed into the substrate (Dorr et al., 2021) In conclusion, other types of waste such as paper, cardboard, sawdust which can be found in urban environments will be introduced as potential waste flows for growing mushrooms and vermicomposting in this project.

2.4.4 Residual Heat

Heat accumulates in greenhouses as a result of solar exposure, this residual energy is usually discarded by ventilation in order to keep the indoor temperatures suitable for crops. However, it can be extracted and used as a heat source for surrounding spaces. Rooftop greenhouses can serve as renewable energy sources instead of relying on gas for heating (ten Caat et al., 2021). Ten Catt et al. aimed to unite the simultaneous differences between supply and demand by utilizing synergetic systems, direct heat exchange and cascading in addition to storage of energy (2021). In the same article, it is illustrated that thermal energy from an 850 square meter greenhouse can heat 47 dwellings. In the synergistic energy system, a rooftop greenhouse is incorporated in order to act as a solar collector in the summertime and collect thermal energy while heating itself during winter time to provide a high-temperature energy source for the dwellings (ten Caat et al., 2021). In the same system, aquifer thermal energy storage (ATES) is suggested to tackle mismatches between supply and demand. The system works by extracting excess thermal energy through the floor cooling system therefore heat is carried in a water medium to heat a city block which operates on low temperatures as explained in the same source. In this system, a supermarket exchanges energy by utilising an air-water heat exchanger to heat the greenhouse by increasing the temperature of the water coming from ATES during the winter months (ten Caat et al., 2021). On the contrary, in the summertime, cold water stored in winter is used to cool the greenhouse (ten Caat et al., 2021). In the same paper, it is noted that if the main goal is to grow food then the carbon footprint of such a system integrated into the heat grid would increase carbon footprint due to electricity use for artificial lighting and heat pumps. However, according to ten Caat et al., if the production of food is rather considered as a byproduct, urban rooftop greenhouses can be a potential solution to heating (2021) There are other examples of utilising residual heat from greenhouses to heat dwellings including the RTG-Lab. In this farm, warmer air from the building heats the greenhouse when the temperature is lower than 15C during the night and winter months while lower temperature air from offices and laboratories cools the farm down when the temperature is above 30C in summertime (Sanyé-Mengual et al., 2014). Another example is built in Naaldwijk, Netherlands, where 800 houses are heated by the greenhouses nearby (Urban Blue Grids, n.d.). Surplus heat from greenhouses is stored underground in summer, and stored heat is used for heating dwellings and the greenhouses in the wintertime as explained in the same source. In conclusion, due to the existence of excess heat in urban conditions, it will be used as a potential energy source for the urban farms and for dwellings depending on the circumstances and season during the development of this project.

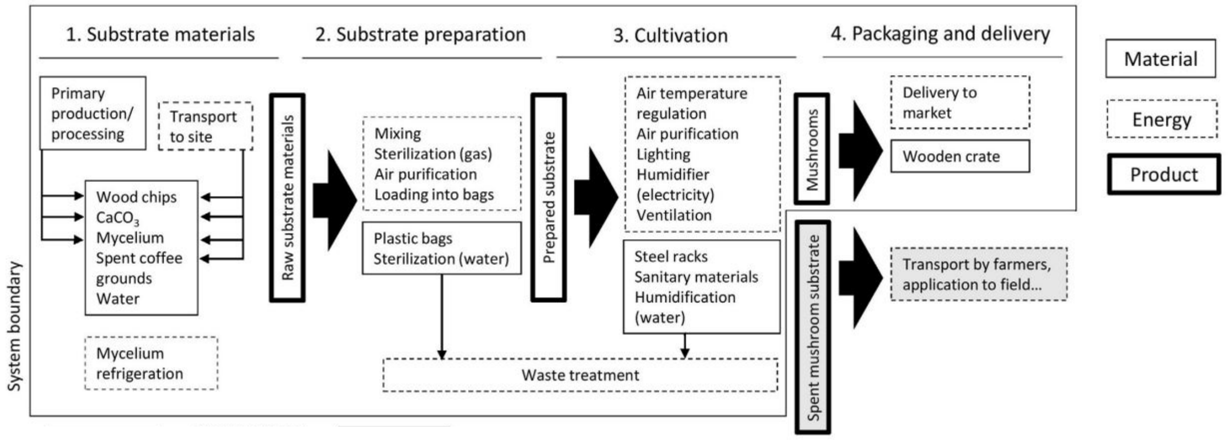


Figure 2.5 Mushroom Production Flow by Dorr et al., 2021

After assessing the potentials of various types of urban waste in order to integrate waste flows into urban farming it is concluded that some waste types have more potential than others. For this research and the development of the decision-making tool CO2, rainwater, excess heat, organic waste including kitchen waste, agricultural waste, paper, sawdust and spent coffee grounds will be incorporated into related urban farming systems.

2.5 Decision-Making Tool & Approaches

The last facet of the project to discuss is the computational aspects. The need for a decision-making tool or a support system in addition to computational approaches will be discussed. There are various computational approaches, each of them is suitable for different purposes while they have various advantages and disadvantages to them. In this chapter, different computational approaches, their advantages, disadvantages and suitability to the project will be described.

2.5.1 Demand for Decision Making Tools

This project aims to develop a decision making support tool for designers who aspire to integrate urban farming with existing urban waste flows in order to close the loop and to make a shift towards symbiotic urban metabolisms. The decision-making tool is needed because there is a range of diverse requirements including soft requirements and hard requirements as Chatzikonstantinou states (2021). In the same article, it is mentioned that cities are one of the most complex human-made arrangements and this complex nature of design calls for tools to support decision-making processes. According to Chatzikonstantinou, the complexity of design challenges human cognition as the problem's complexity increases (2021). Nouran advocates that there is a logical leap in design processes (Kroes, Peter and Meijers, Anthonie, 2006 as cited in Nourian, 2017) and adds that the relation between forms and function are not one dimensional (Nourian, 2017). Consequently, decision making is dependent on intuition and reasoning leaps as discussed in the same article. Nouran states that design is about analysis, synthesis and evaluation in addition to advocating that basing design decisions on scientific knowledge and environmental consequences of design decisions (2017). In another source, it is stated that a systematic approach to design space exploration should be utilized in order to decide on optimal solutions (Chatzikonstantinou, 2021). In conclusion, a decision-making tool supports designers to make systematic and science-based design decisions while eliminating the logical leap.

2.5.2 Computational Design Categories

In computational design, there are main categories of approaches including parametric design, performance-based generative design, generative design and algorithmic design. Even though there are discussions about what these terms exactly correspond to, Caetano et al. describe them as follows: In parametric design, there is a numerical or quantifiable factor forming one of a set that defines a system while "parametric" relates to parameter or parameters (Caetano et al., 2020). Secondly, generative design is described as a design paradigm that employs algorithmic descriptions that are less manual than parametric design as described in the same source. Caetano et al. assert that in such design systems, the system executes encoded instructions until the criteria are satisfied (2020). Moreover, performance-based generative design refers to systems where designers set a performance target and an algorithm finds the best fitting solutions for achieving the design goal (Caetano et al., 2020). The same authors consider algorithmic design generative as well since it employs algorithms to generate models (2020). In this project, a generative design approach will be employed in addition to a rule-based decision-making strategy which refers to a system executing operations based on design rules through "if-then" statements (Cubukcuoglu et al., 2019).

Grammars

Several types of grammars are explored since they can provide ways or approaches of applying rules to make decisions. A grammar has a set of rules that apply repeatedly to an initial object to produce a final object (Duarte, 2005). According to Stouffs et al., it can also be seen as a language where each generation starts with an object and rules to achieve new objects which only contain from a terminal vocabulary (2001). For instance, a rewriting rule is based on "lhs --> rhs", where lhs refers to the similar object to be recognised while rhs specifies the manipulation that object will undergo as exemplified in the same source. In the same article, it is explained that a rule applies to an object's part if the lhs matches that object and the matching part is replaced according to rhs rule.

There are several kinds of grammars including colour, graph, shape, sortal, parallel, discursive, parametric grammars. To begin with, a colour grammar by definition uses colours as descriptive elements which can represent materials or different features. (Knight 1989 as cited in Gu & Behbahani, 2018). Such a grammar can be used for the project to represent different urban farming systems on a 3D model to represent the outcomes visually. Secondly, Gu & Behbahani asserts that graph grammars represent shapes or forms topologically which reflect the interrelations within the design both structurally and functionally (2018). As explained in the same article, graph grammars follow two approaches: geometric and semantic. The geometric approach corresponds to a correlation between graph-topological and shape-geometrical vertices and edges (Gu & Behbahani, 2018). In the semantic approaches, graph's nodes refer to the semantics of shapes as explained by Gu & Behbahani (2018). For instance, in a building, the nodes represent spaces, while edges can represent adjacency or opening between them as explained in the same article. The simplicity and abstractness of graphs is an advantage (Gu & Behbahani, 2018). In this project, graph grammars can be used to define the relationships between different urban farming systems and supply points as graphs can be used to specify interrelations and hold data regarding the vertices. Furthermore, shape grammars are based on shape rules which define a set of spatial transformations of shapes and they are especially powerful with modularity and mass customization of design (Gu & Behbahani, 2018). However, in this project shapes do not have any significance therefore shape grammars are not suitable. Moreover, Stouffs & Krishnamurti states that the main problem with grammars is the matching problem and determining when the lhs match the requirements (2001). The same authors introduce "sorts" as a set of similar models and mention that abstract shapes or descriptions can be given as sorts to optimize the manageability and flexibility of the grammar (2001). Sortal grammars relate to sorts and matching the sort descriptions (Stouffs & Krishnamurti, 2001). Sortal grammars can be used to define which location has which attributes to it to match the design rules for this project. Gu & Behbahani describe parallel grammars as algebraic approaches to implement shape grammars where each rule is applied to determine a different aspect of design simultaneously (2018). Discursive grammars are parallel parametric grammars with heuristics to manage rule selection and heuristics are semantic descriptions such as design goals or criteria, which can act as parameters for applying the rules according to the same paper. Discursive grammar consists of programming and design grammar where the programming grammar forms the design briefs based on data given by users and designing grammar gives the rules for design generation while heuristic leads the designs towards a solution matching the design brief (Duarte, 2005). Such a parallel approach might be useful to apply rules to generate designs in this project due to the parallel nature of the grammar. Lastly, with parametric grammars a set of shapes and their transformations are generalized as one generic shape with parameters to adjust the shape according to design rules (Gu & Behbahani, 2018). Such grammars offer flexibility to design as mentioned in the same article however it does not correspond to this project. Last but not least, with parametric grammars a set of shapes and their transformations are generalised as one generic shape with parameters to adjust the shape according to design rules (Gu & Behbahani, 2018). Such grammars offer flexibility to design as mentioned in the same article however it does not correspond to this project. In conclusion, colour grammars, sortal grammars, graph grammars and potentially discursive grammars can be utilised to develop the decision-making tool for this project.

Generative Design Algorithms

In order to select the most fitting range of approaches to work with, a variety of practices are researched including generative design algorithms. These include replacement, evolution and agent interaction mechanisms.

Replacement mechanisms are defined as where a part of a design is replaced by another to generate variations based on rules (Gu & Behbahani, 2018). Shape grammars are examples of these in addition to L-system algorithms which utilize symbols to generate a design with repeating patterns such as plant modelling and street design as explained in the same article. Another replacement algorithm is parametric combination which refers to replacements affecting only the dimensions and proportions of the objects instead of the entire entity of shapes (Gu & Behbahani, 2018).

The second generative design algorithm is evolution mechanisms which match the design alternatives to a set of fitness functions where after each step the fittest alternative is given (Gu & Behbahani, 2018). Genetic algorithms are examples of such a mechanism (Gu & Behbahani, 2018) and evolutionary algorithms mimic natural selection to evolve and adapt (Lian et al., 2010). As stated by Lian et al. evolutionary algorithms are suitable for multi-objective optimization problems which have Pareto-optimal solutions where the solution is a set of compromised solutions (2010). Even though evolutionary algorithms are powerful tools for multiobjective optimization problems, the necessary population size and generation size usually demand a tremendous amount of computing resources according to the same article. The demand for large computing resources makes such an approach unattainable for this project's scope.

Lastly, another generative design algorithm is agent interactions which are based on how the agents in the design space interact with the context and other related elements (Gu & Behbahani, 2018). Cellular automata, swarm intelligence and space colonizations are examples of agent interaction mechanisms. However, these approaches do not correspond to this project by nature. In addition to previously discussed approaches, some other approaches are also explored to see the best fitting approaches to combine and utilise for the development of the decision-making tool. (See appendix B.1 for others and details)

2.6 Conclusions

In this research, due to the previously given reasoning 9 urban farming systems will be used including different hydroponic systems, aquaponics, mushroom cultivation, vermicomposting and raised beds. And these systems will be fed by existing waste flows including CO2, rainwater, heat and organic waste when possible. In order to compile the data from the literature review regarding the potential integration of urban waste into urban farming systems, a database is created during the research. In this database inputs, outputs and different types of systems are shown. Each different-coloured dot represents a different kind of system configuration with varying inputs and outputs. Afterwards, this table is used to convert the data into an input-operator-output format. Input refers to waste that can be utilised, necessary supplements, growing media while output refers to the main product of the system and byproducts. Operator refers to the system in terms of growing method, technique, type of the system whether it is for food production or a supplementary system and system characteristics (see appendix A.2 - A.10). This data is used to develop the design criteria and rules in later stages.

As discussed previously, some computational approaches are more suitable to employ for the development of the decision-making tool due to their inherent capabilities and the limited time and computational resources of the project. These approaches include colour, graph, sortal grammar and rule-based design in addition to potential environmental simulations as shown in figure 2.6. To begin with, solar analysis can be used to determine the vacant spaces solar exposure if an assumption is not made. Secondly, graph grammars are utilized to create the relations between vacant spaces, potential urban farming systems to occupy those spaces and supply points while each node holds information about the vacant space and supply points such as waste output, structural capacity, solar exposure. Edges between nodes define the relation between the nodes whether there is waste or supply exchange with directions of the exchange if a directional graph is employed. Sortal grammars are useful to match the design rules to existing conditions. For instance, each node can hold an attribute key and its value and if that value matches the design criteria an urban farming system can be assigned to that node. Lastly, colour grammars are used to represent different urban farming systems suggested by the decision-making tool on the site model.To sum up, a combination of different approaches are employed to develop the decision-making tool as the design problem requires various steps with different characteristics to be followed to make a decision.

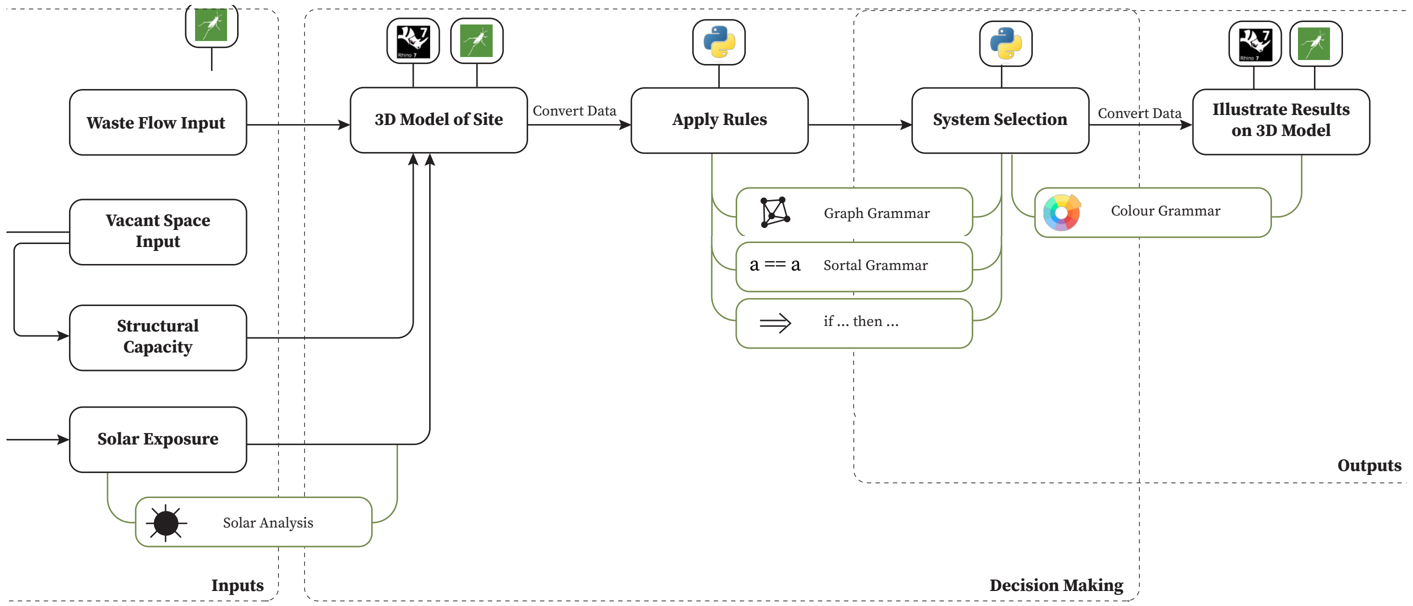


Figure 2.6 Decision Making Process & Computational Approaches (see appendix B.2)

03

DESIGN TASK

3.0 OUTLINE OF DESIGN TASK

3.1 Description of Design Problem

The objective of this thesis is to integrate waste flows into urban farming systems systematically in order to holistically integrate waste streams with food production in urban contexts. The main output of this thesis will be a decision-making tool that will assist designers to select and combine the most fitting urban farming systems to the existing conditions such as available waste flows, vacant spaces and to the requirements given by the stakeholders. The tool aims to maximise the symbiosis within urban environments by perceiving farms as exhange hubs for waste flows and food. When selecting an urban farming system to achieve symbiosis in urban environments, there are a set of questions to ask in order to make viable solutions:

- Where are possible locations to build urban farms? (Vacant Spaces)
    - How high or low is the solar exposure of these spaces?
    - Do these vacant spaces have high load-bearing capacities or should lightweight systems be chosen?
- What are the waste flows and their sources within the site? (Waste Types & Sources)
- What is the main aim of building an urban farm on the given site? (Project Goals)
- What kind of urban farming systems correspond to the existing conditions and requirements? (Production Method)
    - What are the inputs of this system?
    - Can the inputs be sourced from waste flows or by another urban farm?
    - What are the outputs of this system including byproducts?
    - Does the selected system correspond to existing waste flows, characteristics of vacant space and to project aims?

The decision-making tool is designed to guide the designers towards the optimal options depending on a set of variables and the relationship between these variables. These variables are product types, byproducts, production method, production characteristics, inputs to the system, vacant spaces, their locations, load-bearing capacity of the supporting structures, solar exposure of the spaces, their distance to existing waste flows, their distance to vacant spaces and demands given by stakeholders. As illustrated in figure 3.1, the selection of urban farming systems is dependent on these variables and their interrelations. The selection may seem rather simple for one vacant space and a set of existing conditions, however, there might be multiple vacant spaces. The multitude of available spaces complicates the design problem as it introduces the possibility to build a network of urban farming systems serving each other as illustrated in figure 3.1 (see appendix A.1-A.10). Consequently, combinations of urban farming systems depend on a range of circumstances including the distance between waste output and vacant space which can utilize the waste. To conclude, in order to guide designers who may not have the theoretical or practical knowledge regarding urban farming systems through a multi-layered, complex design task a decision-making tool is a practical design equipment.

The decision-making tool is based on inputs, operators and outputs. Inputs are the needs of the urban farming system to function which corresponds to waste flows in the site as resources. Operators refer to urban farming systems including their production method, nutrient delivery technique, mechanical equipment and design characteristics. Outputs are products of urban farms including byproducts. Table 3.2, illustrates these inputs, operators and outputs. After filling out the table with data acquired from the literature review, each system's inputs, system characteristics and outputs are gathered. Each different coloured dot represents a different system with different inputs, and systems' designs and outputs. One system's output can be another system's input therefore the relationship and distance between two systems will be taken into account for design. The decision-making rules and criteria are developed according to this database prior to programming the decision-making tool. Decision making has mainly 4 steps: Data Collection, Data Preparation & Processing, Applying Design Rules and Illustrating Design Decisions. These steps are followed in order to reach results. Lastly, in order to ensure a wide range of applicability of the tool, it is tested on several case studies.

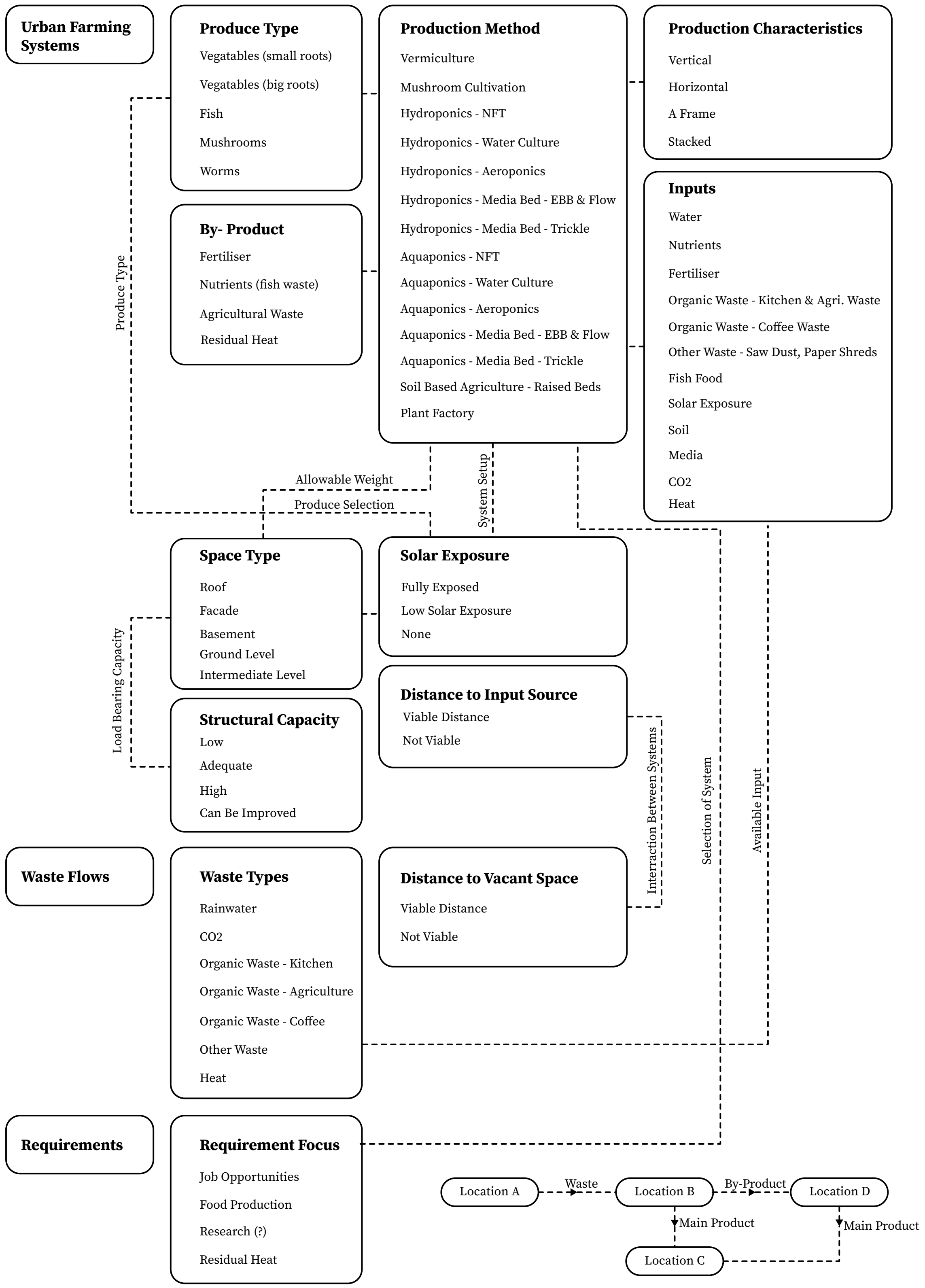


Figure 3.1 Interrelation of Variables

Space	Waste	Supplement	Medium	Growing Technique	Design Characteristic	System Type	Main Product	Bi-Product
Rooftop 🟡🟢	Food Waste ⚪	Fertilser 🟠	Soil 🟠	Compost ⚪	Fish Tank 🔵	Food Production 🟣🔴🟠🟡🟢⚫🟤	Small Crops 🟠🟡🟢⚫🟤	Heat 🟠🟡🟢⚫🟤
Facade 🟢🟤	Coffee Waste 🟣	Nutrient Solution 🔴🟡🟢⚫🟤	Water 🔴🔵🟢⚫	Spawning 🟣	Tank ⚪	Supplementary ⚪	Large Crops 🟠🟤	Food Waste 🔴🟠🟡🟢⚫🟤
Intermediate Floor 🟣🟡🟢🟤	Other Waste ⚪🟣	Calcium 🔵	Fish Tank Water 🔴🟢⚫	Aquaculture 🔵	Stacked System 🔴	Food Producing Supplementary 🔵	Mushrooms 🟣	Spent Mushroom Substrate 🟣
Ground Floor ⚪🟣🔴🔵🟠🟡⚫🟤	CO2 🔴🟠🟡🟢⚫🟤	Lime Bath* 🟣	Air 🟡	Raised Beds 🟠	Horizontal 🟣🟠🟡🟢⚫🟤		Worms ⚪	Fertiliser ⚪
Basement ⚪🟣🔴🔵🟡🟢⚫🟤	Rainwater ⚪🟣🟠🟡🟢⚫🟤		Food Waste ⚪	NFT 🔴🟢	Vertical 🟣🟡🟢🟤		Fish 🔵	Fish Tank Water 🔵
	Heat 🔴		Coffee Waste 🟣	Aeroponics 🟡	Modular Frame 🟡🟢🟤			
			Other Waste ⚪🟣	EBB & Flow 🟤				
			Clay Balls 🟤	Gravity Trickle 🟤				

⚪ Vermiculture
🟣 Mushroom
🔴 Plant Factory
🔵 Aquaculture
🟠 Raised Bed
🟡 Aeroponics
🟢 Hydroponic - NFT
⚫ Hydroponic - Water Culture
🟤 Hydroponic - Media Bed

Table 3.2 Urban Farming Systems (see appendix A.1)

04

DATA COLLECTION

4.0 DATA COLLECTION

In order to make a decision, 3 main categories of information need to be collected such as urban farming systems, vacant spaces and waste outputs. Each category has sub-categories within itself. For vacant spaces, their location, size, structural capacity, and solar exposure should be known. On the other hand, for waste outputs, the sourcing location, waste type and quantity are enough. Lastly, even though data regarding inputs and outputs of systems and the suitability of the system for existing conditions are gathered previously, in order to quantify the decision making waste demands and yields of each system should be calculated.

4.1 Prototype

4.1.1 Simplified Calculations For Waste Demands And Yields

The amount of waste each system needs to use for production is calculated in a simplified manner. These calculations are done for each system and its inputs per square meter. (See appendix D.1 for calculations) It should be noted that these calculations are done only to provide a basis for further steps and they do not take environmental conditions or phenomena into account.

Secondly, yields of different systems per square meter are calculated in order to quantify the food production potentials. These calculations are based on simple geometry since the crop yields are highly dependent on environmental factors including light, CO2 levels and temperature. The minimum distance between different crops and stacking of production trays are prominent factors in these calculations as illustrated in table 4.1. For instance, tomato plants need to be planted with 60 cm distance between each plant horizontally while this distance for lettuce is 30 cm according to agriculture websites. (RHS, n.d.) For stacked systems, the vertical distance between trays is 30 cm based on a vertical farm example. (Viscon Group, 2021) It should be noted that these distances differ from crop to crop and from species to species while such standards are not established formally and are commonly based on experience.

System Name	Horizontal/Vertical	Crop Type	Fruit Yield Annual	Annual Fruit Yield (kg)
Plant Factory	Vertical	Lettuce	624.5555556	87.43777778
Raised Bed	Horizontal	Lettuce	89.22222222	12.49111111
Raised Bed	Horizontal	Dwarf Bush Cherry Tomato	-	2.7
Aeroponics	Vertical	Lettuce	89.22222222	12.49111111
NFT	Vertical	Lettuce	713.7777778	99.92888889
Water Culture	Horizontal	Lettuce	89.22222222	12.49111111
Media Bed	Horizontal	Lettuce	89.22222222	12.49111111
Media Bed	Horizontal	Beefsteak Tomato	256.1403509	89.64912281

Table 4.1 Yields of Different Systems (see appendix D.3 for details)

4.1.2 Analysis Framework

In order to provide guidelines and methodology for the site analysis, a data collection framework is formed. In this framework, what kind of data to collect regarding vacant spaces, and waste outputs are highlighted in addition to interpretations of collected data. Firstly, while collecting data about vacant spaces, their orientation and location in the building (if applicable) should be noted down in order to make interpretations about their structural capacity and solar exposure as given in table 4.2. The structural capacity of spaces is assumed to be low, mediocre or high based on the location in the building or outside. It is assumed that rooftops have low structural capacity, while intermediate floor

Location	Data	Interpretation 1	Interpretation 2
Location A	Basement	stuctural capacity : high (3)	solar exposure : low (1)
Location B	Ground Floor	stuctural capacity : high (3)	
Location C	Ground Floor Outdoor	stuctural capacity : high (3)	
Location D	Intermediate	stuctural capacity : medium (2)	
Location E	Rooftop	stuctural capacity : low (1)	solar exposure : high (3)
Location F	Facade	stuctural capacity : low (1)	
	North	solar exposure : low (1)	
	North East	solar exposure : low (1)	
	East	solar exposure : adequate (2)	
	South East	solar exposure : high (3)	
	South	solar exposure : high (3)	
	South West	solar exposure : high (3)	
	West	solar exposure : adequate (2)	
	North West	solar exposure : low (1)	

Table 4.2 Analysis Framework for Vacant Spaces (see appendix E.0 for details)

Location	Data	Interpretation 1	Quantity Information
Location G	Cafeteria	Organic Waste : Food	serving ... people
Location H	Cafe/Restaurant	Organic Waste : Food	serving ... people
Location I	Agricultural Activity	Organic Waste : Food	
Location J	Wood Workshop	Other Waste : Wood Chips	
Location K	School & Offices	Other Waste : Paper	
Location L	Paper Waste Bins	Other Waste : Paper	
Location M	Espresso Bar	Organic Waste : Coffee	serving ... people
Location N	Conference Hall	CO2	serving ... people
Location O	Classroom	CO2	serving ... people
Location P	Meeting Room	CO2	serving ... people
Location R	Metal/ Sloped Roof	Rainwater	surface area
Location S	Sloped Roof	Rainwater	surface area
Location T	Supermarket	Excess Heat Source	
Location U	Datacenter	Excess Heat Source	

Table 4.3 Analysis Framework for Waste Sources (see appendix E.0 for details)

levels have medium load-bearing capacity. Lastly, spaces on the ground floor or in basements are considered spaces with high load-bearing capacity.

Solar exposure is another factor that effects decision making. Data regarding solar exposure of spaces can be acquired by running a solar analysis however due to time restrictions solar exposure of spaces is determined based on their orientation and location within the building if applicable. Therefore, basements have low solar exposure while rooftops have high exposure. In intermediate floor levels' and outdoor spaces solar exposure is determined by their orientation. North, North-East, and North-West facing spaces have the least exposure while South, South-East, and South-West facing spaces have the most. East or West oriented spaces are considered to have medium levels of sunlight.

Secondly, waste output sources are based on existing functions on-site as shown in table 4.3. For example, if there is a restaurant it will be regarded as a food waste source while the existence of conference halls indicates residual CO2.

Lastly, the project aims such as research, holistic food production, and producing the maximum amount of crops can be determined through communication with the stakeholders and some systems can be prioritised over others based on the project aim. (See appendix A.14) For instance, if the project aim is research then systems such as plant factories and aeroponics should be prioritised. If the driving force of the project is producing maximum amount of food, systems with higher productivity should be prioritised during decision making. These systems include stacked systems such as plant factories, NFTs, aeroponics and potentially media beds. Lastly, if the most important factor behind urban farming is holistic food production, system decisions should be based on the symbiosis rate in order to ensure use of existing waste resources. For research reasons in this project, the project aim is assumed to be holistic food production while developing and testing the prototype.

4.1.3 Representation of Data

There are 3 main data categories to be collected: vacant spaces, waste outputs and urban farming systems. Each category holds information related to the corresponding category. Vacant spaces and waste outputs are represented by nodes and these nodes deliver data about sizes, structural capacity, solar exposure and waste quantity, waste type respectively in addition to the locations of these nodes. For the sake of simplicity, vacant space sizes and waste quantities are simplified to be represented ranging from 1 to 3 as shown in figure 4.1. The ranges for vacant space sizes and waste quantities are calculated in such a way that simplified values for both data sets correspond to each other. Both waste quantities and vacant spaces' sizes are represented by a range between 1 and 3. Firstly, waste demands per square meter of each urban farming system for each required waste type are calculated. Then the vacant space sizes are divided into ranges of minimum and maximum values based on the distribution of surface areas. These minimum and maximum values defining each range are multiplied with waste demands per square meter for each waste type required by each urban farming system. This multiplication establishes the ranges for waste quantities as shown in table 4.4. Lastly, real waste quantities are simplified by checking whic range the quantity corresponds to. If a waste type can be used by different systems and if the quantities needed for these systems vary, then the maximum demand is taken into account since the rest will fall under the maximum quantity.

for each waste type and each system:
Minimum Waste Demand = (waste demand per m2) x (range min)
Maximum Waste Demand = (waste demand per m2) x (range max)

Rainwater (W6)	L/m2	Range 1 (kg) (0 - 200 m^2)		Range 2 (kg) (200 - 1000 m^2)		Range 3 (kg) (1000 m^2- above)	
Plant Factory	600	0	120000	120000	600000	600000	above
Raised Bed	2600	0	520000	520000	2600000	2600000	above
Aeroponics	1306.902544	0	261381	261380.5088	1306902.544	1306902.544	above
NFT	1633.62818	0	326726	326725.636	1633628.18	1633628.18	above
Water Culture	7800	0	1560000	1560000	7800000	7800000	above
Media Bed	3120	0	624000	624000	3120000	3120000	above
Vermicompost	0.8125	0	163	162.5	812.5	812.5	above
Mushrooms	548.4375	0	109688	109687.5	548437.5	548437.5	above

Table 4.4 Rainwater Demand of Different Systems (see appendix D.1 for other waste types)

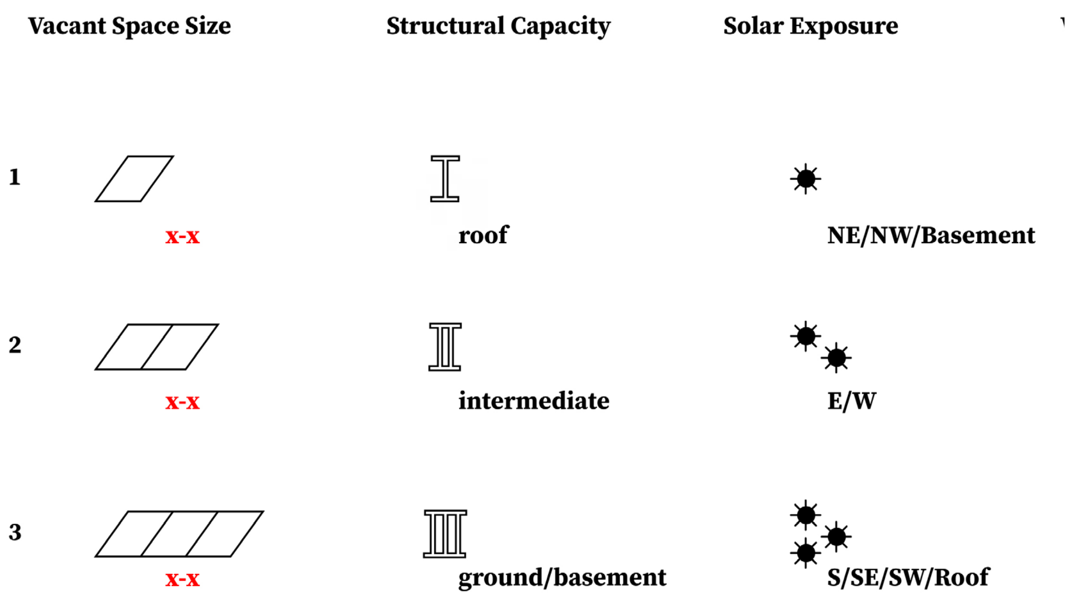


Figure 4.1 Vacant Space Characteristics Represented by Ranges (1-2-3)

4.2 Automated Data Collection (Waste Output Points, Vacant Spaces)

In this research, data collection is done manually however in theory there are ways to collect data regarding waste sources and vacant spaces. These alternative methods are using drone footage&machine learning and ArcGIS.

4.2.1 Drone Footage & Machine Learning

Drones can be used to detect vacant spaces or occupied spaces. After conducting a drone led site survey, the footage can be analysed by a deep learning platform in order to provide information regarding vacant spaces, their location, and size. Similar technology is provided by a company called Nanonets to detect defects in wind turbines, people walking in an aerial view, diseases on agricultural fields, predicting yields of the field and for many other detection purposes. Another technological advancement which could relate to the mapping of vacant spaces is AMP (Automated Mapping Platform) by Woven Planet Holdings. Currently, it is used to collect data on streets, traffic lights or curbs in order to train AI Trucks. To conclude from these examples, machine learning models can be trained in order to detect and map vacant spaces.

4.2.2 ArcGIS Platform

Another platform which holds information regarding the built environment is the ArcGIS Platform. There are a few possible uses for ArcGIS in this research. The first one is to build a 3D Model by using building footprints and building heights data. This set of data can be exported from ArcGIS in .xls format. The shapefile can be used to draw the building footprints while the building heights data can be used to extrude the buildings by using Grasshopper as shown in figure 4.2. Another use for ArcGIS is to determine vacant spaces by excluding layers of data such as parking lots, roads, and buildings. This way the location of potential vacant spaces can be mapped. The last potential use of the same tool is to map the location of waste sources. For example data regarding the location of restaurants can be gathered to map the food waste sources on site. The functions to look for can be based on the site analysis framework. (See appendix E.0)

4.2.3 Waste Audits

The last data set the decision-making tool needs to run is the quantities of waste. This information is used to determine whether a waste source is enough to be used for an urban farming purpose taking place in a vacant space with the current size. How the vacant space size relates to waste quantities will be explained further in the following section. As per UN Sustainable Development Goal 12, reducing food waste by half is advised (United Nations Environment Programme, 2021). And to do so, food waste produced by restaurants should be audited at the retail and consumer levels as advised in the same source. The methodology of auditing is given in the Global Chemicals and Waste Indicator Review Document (United Nations Environment Programme, 2021). There are various ways of waste auditing suggested in the same document. One of them is directly weighing the waste after separating it. By using a smart scale, the measurements can be logged into a spreadsheet to be used in the coming stages of decision making. Such measurement tools are widely available in the market for household and industrial uses.

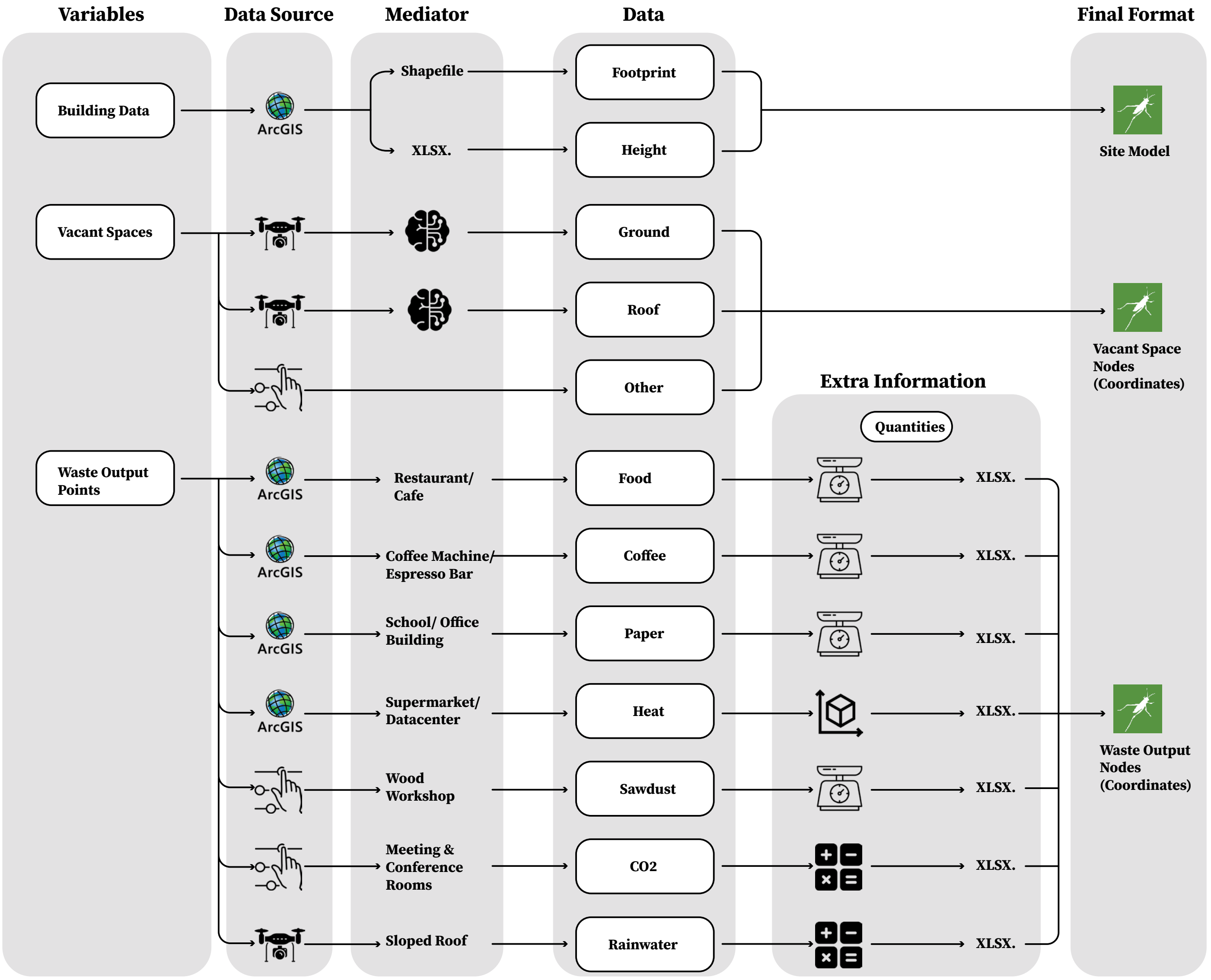


Figure 4.2 Automated Data Collection

4.2.4 Manual Data Collection And Input

Another method to collect the data is rather a low-tech data collection technique. Data regarding vacant spaces, waste sources and quantities can be collected and inputted manually. In this research, data collection regarding waste sources and vacant spaces is conducted this way due to time and resource limitations.

To begin with, manual detection of waste sources is done by mapping the restaurants, espresso bars, sloped roofs, data centres, supermarkets, conference halls, model making spaces, and carpentry ateliers for different waste types based on the analysis framework. Data regarding waste quantities needs to be available or be assumed based on the available data. Secondly, the location of vacant spaces are mapped by checking a satellite view and identifying different uses of space such as greenery, parking lot, building for ground floor level; and PV panels, mechanical equipment, and skylights on the roof level. If there are vacant spaces below ground such as basements or if there are underused spaces in buildings like attics, these need to be detected by simply viewing the floor plans of buildings and during a site survey.

05

STEP BY STEP DECISION MAKING

5.0 STEP BY STEP DECISION MAKING

5.1 Data Preparation

After collecting the data, the next step is to prepare it into a format which can be used by the decision-making tool. In this step, information regarding vacant spaces, waste sources, waste sources within a certain distance from vacant spaces and vacant spaces within a certain distance from other vacant spaces is assembled and stored.

Firstly, after mapping the waste sources in Rhino or any other CAD software, an identifier should be given to the waste output point. In this research waste output points are named with "WO+(integer)". These identifiers and the coordinates of waste sources are exported to an excel sheet using a grasshopper component. Afterwards, the building names of these sources, waste types and quantities are filled into the spreadsheet.

Secondly, in order to fill in the data regarding vacant spaces boundaries of these spaces should be drawn and be recognised by a Grasshopper component. The vacant spaces are sorted from largest to smaller vacant spaces. Sizes should be sorted here in order to prioritise the largest space first instead of coming to a point where resources are used for small pieces of land rather than more efficient larger spaces. Then, similar to waste sources identifiers are given to vacant spaces in the form of "V+(integer)".These identifiers, sizes of spaces and their coordinates are exported to a spreadsheet. In the spreadsheet, the building names of these spaces, their locations in the building and their orientations are filled in. In the table below, the aforementioned spreadsheets and data types to be filled are illustrated. (Table 5.1 & 5.2)

Identifier	Coordinates	Size	Building	Location	Orientation	Tag	Node Type
V0	{2020.580643, 387.391529, 0}	31174.53666	outside	outside	S	V0	vacant space
V1	{935.001269, 534.594454, 0}	23824.09908	outside	outside	N	V1	vacant space
V2	{766.997201, 400.341737, 12.0}	9665.369642	3me	roof	S	V2	vacant space

Table 5.1 Example of Vacant Space Data

Identifier	Coordinates	Source	Waste Type	Quantity	Tag	Node Type	Waste
WO0	{940.159971, 709.345779, 0}	V27	W7	1732320	WO0	waste	heat
WO1	{274.847735, 547.422372, 0}	BK	W2	592000	WO1	waste	sawdust
WO2	{304.212084, 575.947739, 0}	BK	W1	938	WO2	waste	food

Table 5.2 Example of Waste Data

Another dataset that needs to be exported from the geometrical relations of nodes is the nearby waste nodes for each vacant space under a set radius. This dataset is needed in order to exchange waste within given limits and to decrease the distance waste needs to travel from the source to the farm. First, all the connections between vacant space nodes and waste nodes are drawn. Afterwards, the lengths of connections are sorted and the ones shorter than a certain length are filtered. The reason behind sorting is to use the waste source closest to the space first. Lastly, the start points' coordinates are matched to vacant spaces' coordinates while end points' coordinates are matched to waste outputs' coordinates in order to have pairs of vacant spaces and waste source's identifiers for each connection. Similarly, connections between vacant spaces themselves are drawn, sorted and filtered. However, to establish nearby vacant spaces for each vacant space since there will be double lines, duplicate lines need to be removed. Finally, these datasets are exported to a text file in the form of a list. In this project, for prototyping reasons the datasets are exported from Grasshopper to Python and further steps are taken in the Python environment.

5.2 Decision Making Stages

Overall, there are 4 stages of decision making which mainly correspond to varying design rules. As illustrated in figure 5.1, the first 3 stages are quite similar except for the gradually increasing search radius. The search radius is defined by the maximum horizontal distance waste needs to travel from source to farm. The search radius is kept as short as possible in order to decrease the distance between farm and waste. The 2nd and 3rd stages are optional and can be user-defined depending on the project. The 4th stage is also optional and the main aim of this stage is to assign a system to each vacant space regardless of the found items. The rules for these stages will be explained in the next sections

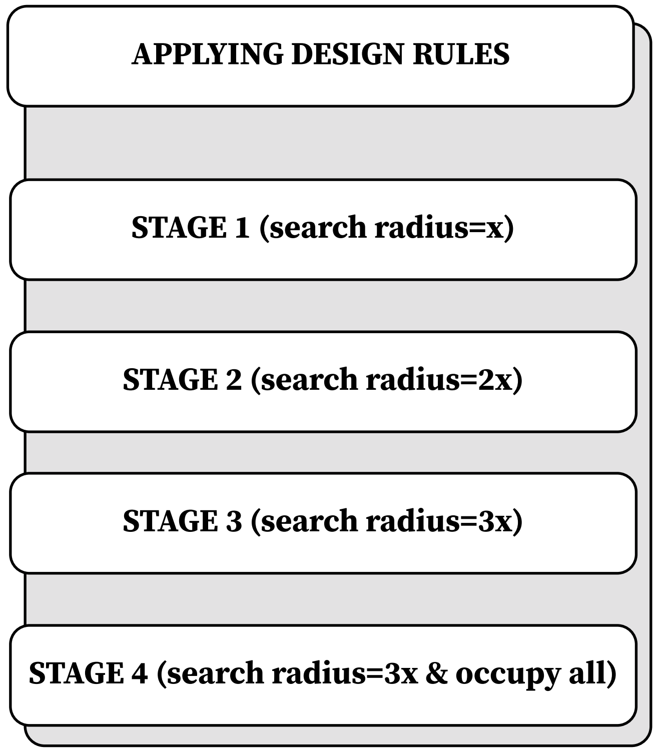

APPLYING DESIGN RULES
STAGE 1 (search radius=x)
STAGE 2 (search radius=2x)
STAGE 3 (search radius=3x)
STAGE 4 (search radius=3x & occupy all)

Figure 5.1 Decision Making Stages (See appendix C.2-C.7 for details)

5.2.1 Stage 1 Assign Systems

Data Processing

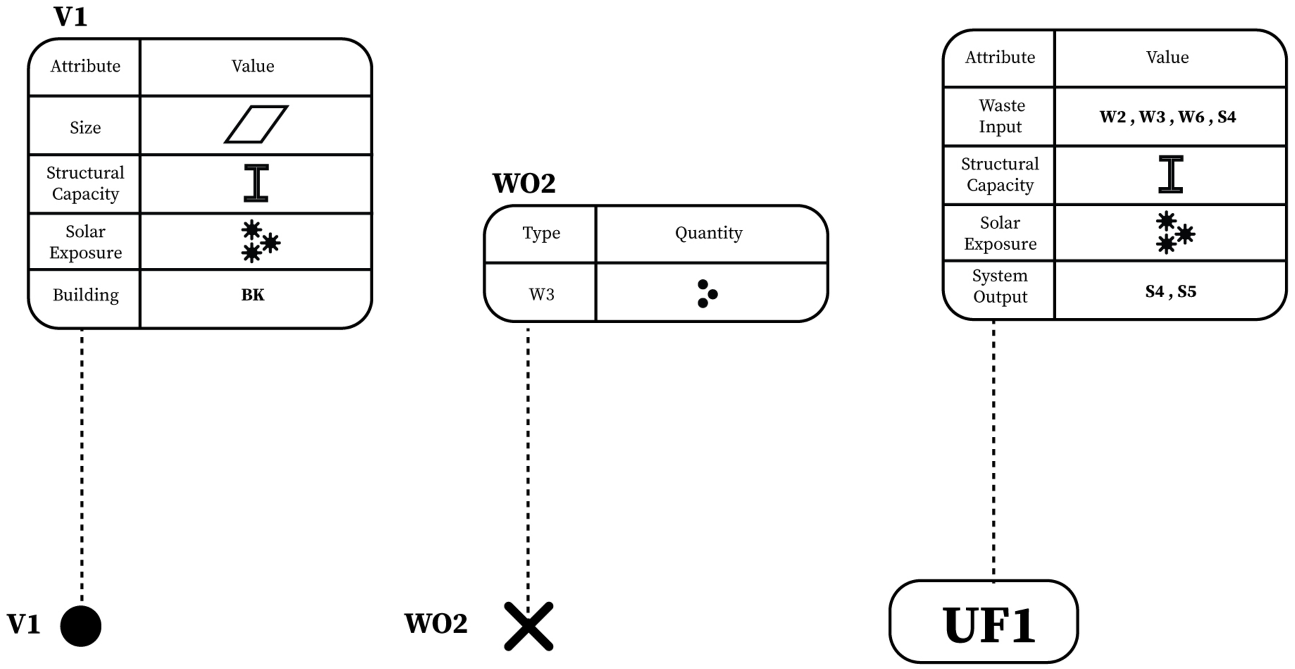
After having all the necessary datasets of vacant spaces, waste outputs, nearby waste sources, and nearby vacant spaces ready; the next step is to process the datasets. In this step, there are 4 main steps: simplifying space's sizes and waste quantities, determining potential food production systems for each space, making a list of nearby waste flows for each space and making a list of found and missing waste sources for every potential system of each vacant space. (See appendix C.4 for details)

Necessary Datasets

Urban Farming Systems

This dataset represents different urban farming systems in the scope of research such as Vermicompost, Agriculture Mushroom Production, NFT, Media Bed, Raised Bed, Water Culture, Plant Factory, and Aeroponics. The systems' characteristics like solar demands and weight are included after simplifying them. This simplification relates to the structural capacity and solar exposure of spaces. Therefore, heavier systems need spaces with higher structural capacity or systems with high solar exposure demands should be placed in locations receiving sufficient sunlight. Systems' inputs, outputs and supplement demands are listed. Lastly, the nature of the system is indicated relating to whether that system is a food production system, food-producing supplementary system or a supplementary system. Food production systems refer to systems which only produce food including mushrooms, soft fruits and leafy greens. Food producing supplementary systems refer to systems which produce supplements in addition to food. Aquaculture is an example of food-producing supplementary system since it produces fish and nutrient-dense water which is a supplement for hydroponic systems. Lastly, supplementary systems refer to systems that only produce supplements but not edible produce. For instance, vermicompost systems produce fertiliser and worms, none of which are advised to be consumed by humans.

{ UF1 :{ type: Supplementary, weight: 3, solar demand: 1, inputs: [food waste, rainwater, sawdust, paper], supplement: None, output: [fish food, fertiliser]}}


V1

Attribute	Value
Size	▱
Structural Capacity	I
Solar Exposure	✱✱✱
Building	BK

WO2

Type	Quantity
W3	∴

Attribute	Value
Waste Input	W2 , W3 , W6 , S4
Structural Capacity	I
Solar Exposure	✱✱✱
System Output	S4 , S5

Figure 5.2 Necessary Datasets of Vacant Spaces, Waste and Urban Farming Systems (left to right)

Vacant Spaces

Vacant spaces data set includes identifiers, coordinates, building, size, location in the building and orientation. During the data processing step, vacant spaces' sizes should be simplified by using the predefined ranges. These ranges can be defined by assessing the distribution of surface areas. Spaces' location in the building and orientation are used to assign integers ranging from 1 to 3 to structural capacity and solar exposure as further explained in section 4.1.2.

{ V1 :{ location: (15,20,0), tag: V1, building: BK, size: 3, structure: 3, solar:1, node: vacant}}

Waste Sources

The waste sources dataset holds information regarding the coordinates, source building, waste type and quantity. Waste quantities are simplified from real data to integers ranging from 1 to 3 for research purposes. These ranges and calculations were previously described in section 4.1.3.

{ WO1 :{ location: (17,28,0), tag: WO1, building: BK, size: 3000, type: sawdust, node: waste}}

Nearby Waste:

This list of space-waste pairs represents nearby waste sources for each vacant space. The first identifier is for vacant spaces while the second identifier represents waste sources. A number of nearby waste pairs can be imported with varying search radii and be used to increasing the search radius step by step.

Nearby Vacant Spaces:

This list of vacant space pairs represents neighbouring vacant spaces around each vacant space. This dataset is used during decision making for food-producing supplementary and supplementary systems.

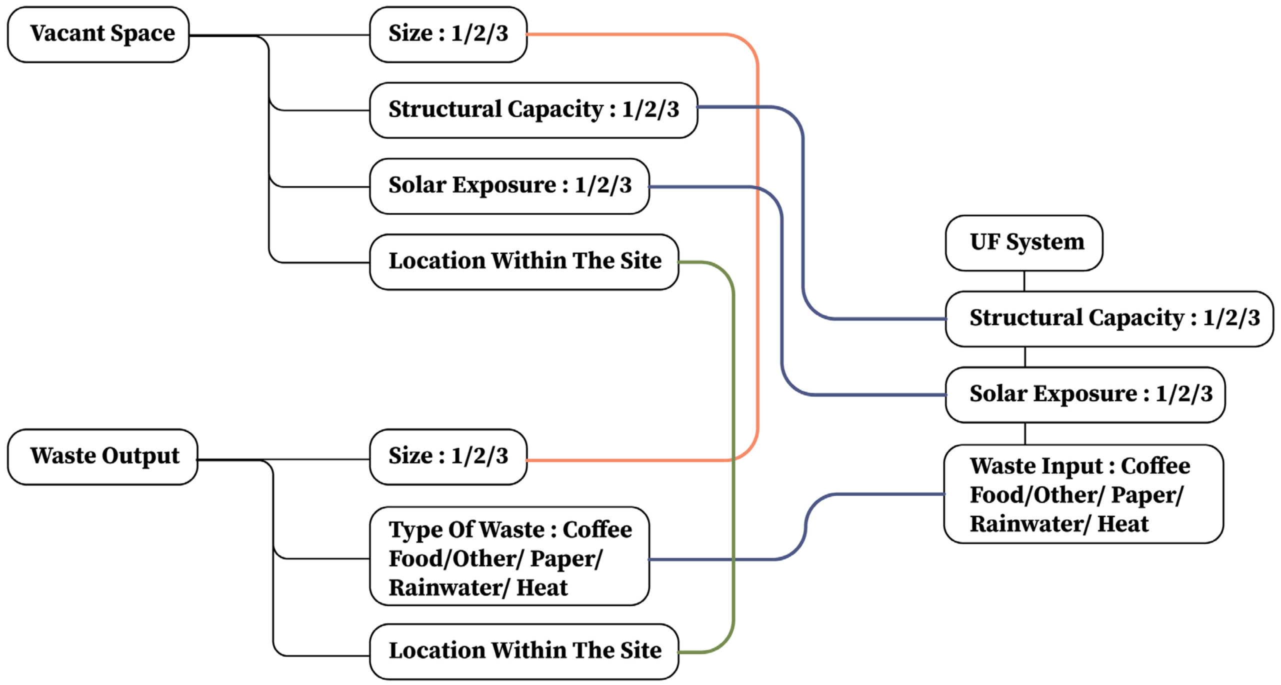

Vacant Space
Size : 1/2/3
Structural Capacity : 1/2/3
Solar Exposure : 1/2/3
Location Within The Site
UF System
Structural Capacity : 1/2/3
Solar Exposure : 1/2/3
Waste Input : Coffee Food/Other/ Paper/ Rainwater/ Heat
Waste Output
Size : 1/2/3
Type Of Waste : Coffee Food/Other/ Paper/ Rainwater/ Heat
Location Within The Site

Figure 5.3 Overview of Design Rules

Data Processing

Step 1. Determining Potential Food Production Systems

Whether a food production system is suitable for a space depends on structural capacity and solar exposure of space. In this step, potential food production systems are added to a list corresponding to respective vacant space if solar exposure of space is equal to solar demands of the system and if the structural capacity of space is bigger than or equal to the weight of the system as illustrated in figure 5.3. (See appendix C.2-C.7 for flow charts)

Step 2. Nearby Waste Sources

At this stage, the dataset of space-waste source pairs is available however the format of data needs to be converted to a format which is easier to access and follow. Therefore a list of nearby waste sources for each vacant space is made based on the aforementioned data set. In addition, since transferring some waste types such as CO2, residual heat and rainwater would require a vast amount of infrastructure, these waste types are used only if their sources are in the same building as the vacant space. Otherwise, the connection should be removed from the pairs list.

Step 3. Found & Missing Resources

This is the last dataset which needs to be created based on potential food production systems and nearby waste sources. In this data set, the type of found and missing waste inputs for each potential system for each vacant space is given along with the sources of found waste types. Missing and found items for each system are based on their inputs given in the urban farming systems list. In the next step, whether the found items are enough in quantity for vacant spaces is assessed. To do so, this task is divided into two steps. First, whether the space's size and found item's size correspond to each other is checked. If the quantity and size are equal to each other the waste type and source are added to the list of enough waste sources, and the same is done for the rest of the list. Then, whether different sources of the same waste type can be combined is checked. If the sum of waste quantities is equal to vacant space size, these waste sources and types are added to enough waste list. If none of these conditions can be satisfied, since there is not enough waste for the space these waste types are removed from the found list and added to the list of the missing items.

Waste source size = Vacant Space Size
or
sum(Waste source size) = Vacant Space Size

The last step is to sort the potential urban farming systems for each space based on the symbiosis rate. Symbiosis rate is found items over total demand. By doing so, it is ensured that the system with the most found items will be prioritised.

Symbiosis Rate = (Enough Waste) / ((Enough Waste) + (Missing items))

Applying Design Rules

In order to, assign urban farming systems to vacant spaces a layered approach is followed. Food production systems are assigned first, followed by food-producing supplementary systems and supplementary systems respectively. This approach is adopted since it corresponds to the inputs and outputs of different system categories. To explain it further, supplementary systems provide supplements for food-producing supplementary systems while food-producing supplementary systems provide supplements to food production systems. By starting from the food production systems it is ensured that a supplementary or food-producing supplementary system will not get assigned if there is no demand for the corresponding supplement.

Assigning Food Production Systems

The system selection starts with food production systems and a previously created list of spaces, potential systems, found and missing items are used in this step. The tool iterates over vacant spaces and potential systems to apply the design rules. There are two restrictions in this step. The first one is the maximum number of missing resources which is currently set to 2. This is a preference which can be decided by the designer and will be illustrated in section 5.3. The second rule is regarding critical items. Critical items are defined by resources which are a must for a system to function. For instance, food waste is essential for vermicomposting while spent coffee grounds, paper waste and sawdust are necessary for mushroom production. Other waste types such as CO2, rainwater, and excess heat can be either compensated by the grid or chosen not to be supplied. While applying the rules hardest criteria to satisfy come first then the rules are eased gradually. For example, first, the tool looks for a potential system which does not have any missing items, then potential systems with 1 missing item and lastly 2 missing items. If there are missing items then a system can only be assigned if none of those missing items are critical items. After assigning a system, the used waste sources should be stored in order to not use them for another system in the coming system selections. In addition, a new list of waste source-space pairs is created which represents the waste exchange between respective nodes. The design rules for assigning food production systems are given below.
(See appendix C.2-C.7 for flow charts)

```
1. Assign a system if there are no missing items
    a) Add used waste outputs to the “used source” list
    b) Create new connections between space and used waste sources
2. Assign a system if there is one missing item
    2.1 If the missing item is not a critical item
    2.2 If found items are not already used by another space
        a) Add used waste outputs to the “used source” list
        b) Create new connections between space and used waste sources
        c) Store missing items
3. Assign a system if there are two missing items
    3.1. If none of the missing items is a critical item
    3.2. If found items are not already used by another space
        a) Add used waste outputs to the “used source” list
        b) Create new connections between space and used waste sources
        c) Store missing items
```

After running through all these rules, assigned systems and corresponding spaces are carried to the next step to assign food-producing supplementary systems. Systems which got assigned a farming system will be referred to as occupied spaces in the coming steps.

Assigning Food-Producing Supplementary Systems

In this step, whether there is a demand for nutrient-dense water is assessed for each occupied space. The necessary inputs for the system that can produce nutrient-dense water are determined. Neighbours of the space are evaluated in terms of size. If the size of the neighbour is equal to the size of the space and if the neighbour can accommodate the needed system in terms of structural capacity and solar exposure, the neighbour is added to the list of potential spaces to provide nutrient-dense water to the initial space. All of the needed resources should be found nearby for the food-producing supplementary system to get assigned to the neighbour space therefore the nearby waste sources for the potential space are checked. Used waste sources should be stored and new space-space pairs should be added to the list of connections to represent the resource exchange. (See appendix C.2-C.7 for flow charts)

```
1. If assigned UF systems need “nutrient-dense water”, make a list of occupied spaces, supplement
demand, potential UF system to produce that supplement and inputs of potential UF system
2.If vacant spaces near supplement-needing UF can accommodate potential UF system add them
to the potential source list
3. For each potential source check if all the inputs of the potential UF system can be found nearby
    3.1 If yes and if missing items are not critical
        a) Assign a potential UFf system to that vacant space
        b) Create new connections between UF and potential source
        c) Create new connections between potential sources and waste sources
        d) Add used waste outputs to the “used source” list
    3.2 If no
        a) Continue with the next potential source
```

After this step, occupied spaces and assigned systems are used in the next step to assign supplementary systems.

Assigning Supplementary Systems

In this step, whether there is a demand for fertiliser or fish food is assessed for each occupied space. The necessary inputs for the system that can produce these outputs are stored. Neighbours of the space are evaluated in terms of size. If the size of the neighbour is equal to the size of the space and if the neighbour can accommodate the needed system in terms of structural capacity and solar exposure, the neighbour is added to the list of potential spaces to provide supplements to the initial space. All of the necessary items need to be available nearby for the supplementary system to get assigned to the neighbour space therefore the nearby waste sources for the potential space are checked. Used waste sources should be stored and new space-space pairs should be added to the list of connections to represent the resource exchange. (See appendix C.2-C.7 for flow charts)

```
1. If assigned uf systems need “fertiliser” or “fish food”, make a list of occupied spaces, supplement
demand, potential uf system to produce that supplement and inputs of potential uf system
2. If vacant spaces near supplement-needing uf can accommodate potential uf system add them to
the potential source list
3. For each potential source check if input of the potential uf system can be found nearby
    3.1 If there is no missing item
        a) Assign potential uf system to that vacant space
        b) Create new connections between uf and potential source
        c) Create new connections between potential source and waste sources
        d) Add used waste outputs to “used source” list
    3.2 If no
        a) Continue with the next potential source
```

5.2.2 Stage 2 & 3 Increase Radius

After going through the first stage of decision making, unoccupied nodes can be carried to stage 2 to run the same decision-making rules however with an increased radius. Stage 2 can be followed by stage 3 to increase the search radius even more. The search radius can be increased as many times as needed. And for each stage with an increased radius, nearby waste and nearby vacant space datasets should be altered according to the determined search radius. This alteration can be done only by changing the maximum length. (See appendix C.2-C.7 for flow charts)

1. *Exclude connections between spaces & waste sources from the list if they are not in the same building*
2. *Exclude connections if the waste source or the vacant space is already used*
3. *List waste sources within the desired radius for each system*

5.2.3 Stage 4 Occupy All

Stage 4 is an optional stage dependent on the user and project. If the goal is to occupy all the vacant spaces then this stage can be run if not Stage 5 Analysis can be started. The previously described layered approach is still valid in this stage however the design rules are eased. However, theoretically, the rules for supplementary systems can be tailored to assign a system only if all the critical items are found regardless of missing items. This feature is currently missing in the prototype, but it could be a good addition to the decision making strategy as it adheres to the principle of assigning systems only if the critical items can be found. The overview of rules for stage 4 is listed below. (See appendix C.2-C.7 for flow charts)

Step 0: Preparation

1. *Exclude connections between spaces & waste sources from the list if they are not in the same building*
2. *Exclude connections if the waste source or the vacant space is already used*
3. *List waste sources within the maximum distance for each system*
4. *Identify potential systems for each space based on structural capacity & solar exposure*
5. *Make a list of found waste sources for each space and for each potential system for the respective space*
6. *Make a list of missing waste sources for each space and for each potential system for the respective space*
7. *Sort potential systems for each space based on how symbiotic that system would be within the context*

(Found items) / ((Found items) + (Missing items))

Step 1: Assign Food Production System

1. *Assign a system if there is a found item*
    - *a) Add used waste outputs to the "used source" list*
    - *b) Create new connections between space and used waste sources*
2. *Assign a system if there is only 1 non-critical missing item*
    - *a) Add used waste outputs to the "used source" list*
    - *b) Create new connections between space and used waste sources*

Step 2: Assign Food-Producing Supplementary System

1. *Assign a system if the demanded items match the found items*
    - *a) Add used waste outputs to the "used source" list*
    - *b) Create new connections between space and used waste sources*
2. *Assign a system and supply demanded items externally*

Step 3: Assign Supplementary System

1. *Assign a system if all the inputs are found*
    - *a) Add used waste outputs to the "used source" list*
    - *b) Create new connections between space and used waste sources*
2. *Assign a system if there is one missing item and the missing item is not critical (In Theory)*
    - *a) Add used waste outputs to the "used source" list*
    - *b) Create new connections between space and used waste sources*
3. *Assign a system and supply demanded items externally (Prototype)*

5.2.4 Stage 5 Illustrating The Results

After making design decisions, the last step is to illustrate the results both numerically and visually. Numerical analysis is done to quantify the yields, and amount of waste used and to give a breakdown of assigned systems. Yields can be calculated by utilising the previously mentioned yield of each system per square meter and multiplying the value by the space size to give an idea of the food production potential in a simplified way. The total amount of used waste can be easily calculated by using the waste quantities inputted initially and the used waste list.

Visualisation is done in order to visually express which system is assigned to which space and to show different waste exchanges. To do so separate lists need to be formed. A list of vacant spaces along with their identifiers, coordinates and assigned systems is formed. This data is exported to the same file which was previously used to map the vacant spaces and waste outputs in the preferred visualisation platform. In this case, Grasshopper is used to visualise the data. The newly imported coordinates are matched with the initial ones to locate the vacant spaces to the corresponding surface. Since urban farming systems are represented by "UF+(integer)", the integer is used to differentiate between different systems. Afterwards, different urban farming systems are represented by different colours. In order to, visualise the waste exchange, pairs of waste and vacant space along with the waste type being exchanged should be put in a list and be exported in a text file. This file is read with a custom made component by using Python in Grasshopper to identify start and endpoints and the waste type. After defining the start and end points a line is drawn between them. The waste types are represented in different colours.

5.3 User Interaction (Representative)

Even though the decision-making tool is composed of a set of technical reasonings, the interaction between the designer and the tool has a significant value. The tool should be easy to operate and be flexible for various project parameters. The methodology and the prototype created within the scope of this research can run in the background to make decisions. However, since the prototype is currently a combination of Grasshopper and Python script, it is difficult for designers to operate without any coding experience. To make the decision making clear and less intimidating a representative user interface, called Foodcyle, is designed. The user interface is simply used to translate designer demands to design parameters. There are mainly 5 panels in the interface: Data Input, Design, Analysis, Customisation and Adaptability.

Data Input:

The data regarding the site, vacant spaces and waste outputs are defined and quantified in this panel. The data input starts with defining the site and its boundaries. The second step is to define what can be considered a vacant space including different floor levels, and limitations regarding the distance from buildings. In the same panel, the data collection method is picked. There are 3 different ways of collecting data as previously described in section 4.2. ArcGIS, drone footage and manual input are these data collection methods and they can be combined to work together. If manual input is selected, then the footprints of spaces are selected on the satellite view by clicking on them. Therefore, collaborations between satellite view providers and GIS Data providers are needed to have access to an aerial view and to draw footprints respectively. After defining and locating the vacant spaces waste outputs are defined. In this panel, which waste types to include and how to collect that data are chosen by the designer. ArcGIS data can be used to identify waste sources based on the previously mentioned analysis framework in section 4.1.2. If the manual data input method is selected than the designer needs to drop pins to the location of waste outputs and fill in necessary information including waste type, building name and waste quantity. This means that the data regarding waste needs to be collected or estimated beforehand. The estimation can be done based on the estimation guidelines mentioned in section 4.1.1. (See appendix D.1).

Design:

The design stage starts with a questionnaire asking about the design rules and the preferences of the designer. On the right-hand side of the screen, a brief explanation is given regarding the terminology. The question and reasoning behind each question are given below. (See appendix H.6 for ther relation between script and the questionnaire)

What is the aim of the project?

Holistic Food Production
Research
Maximum Productivity

The driving force behind this question is defining the prioritisation of urban farming systems. For instance, for maximum productivity, stacked systems should be prioritised over other indoor food production systems. Raised beds can be one of the last systems to use since they are usually used outdoors and all year round food production is challenged by environmental factors. For research purposes, plant factories and aeroponics can be the first systems in the list and have priority over other systems. Finally, in this research holistic food production is the main purpose therefore systems that can be easily integrated into urban contexts are prioritised over others. (See appendix E.0 for prioritisation)

Should all the waste sources be found for food production systems?

Only Critical Items
Both Critical and Non-Critical Items

This question is asks the user to determine the criteria to assign a system. Critical items are necessary resources for a system to work properly. If a critical item is missing then that system cannot be assigned to that space. Non-critical resources can be supplied externally. Looking for both critical and non-critical items limits the decision making however it can be a useful option if the aim is 100% off-the-grid urban farms in terms of providing resources.

How many missing resources are acceptable?

Insert number

If in the previous question, the "only critical items" option is chosen, then systems can get assigned even with missing items. This question is defining the number of items that can be missing.

Should all the waste sources be found for food-producing supplementary systems?

Not Important
Yes

This question refers what happens if the input of aquaculture is missing. There is only one item needed for aquaculture and that item is non-critical. Therefore the answer to this question determines whether a system can be assigned if that one input is missing.

Should all the waste sources be found for supplementary systems?

Only Critical Items
Both Critical and Non-Critical Items

The response to this question determines whether a vermicompost system can be assigned if a non-critical item is missing. If only the critical items option is selected, the response to this question is 1 by default. If only the critical items option is selected, the number of items that can be missing is 1 by default. Therefore, there is no need to ask how many items can be missing.

How far can the waste sources be from vacant spaces?

Insert Number

The response to this question determines the initial maximum distance between waste sources and vacant spaces as well as vacant space to vacant space connections.

Can this distance be increased if there are vacant spaces left?

Yes
No

This question relates to the stages after the first stage and questions whether the initial search radius can be increased if there are unoccupied spaces. According to the response, the maximum length filter in the data preparation stage should be adjusted.

What is the maximum distance waste sources can travel?

Insert Number

This number is the maximum search radius.

How many steps should there be until it reaches the maximum value?

Insert Number

After defining the minimum and maximum, the radius can be increased gradually based on the step count determined by the designer.

Is there a possibility to add infrastructure to transfer CO2, heat and rainwater?

Yes
No

This question relates to the data processing stage where CO2, heat, and rainwater connections are excluded from the pairs list if the waste sources and vacant spaces are not in the same building.

Should all the vacant spaces be occupied even if there are not any found items?

Yes
No

The last question adds another stage to decision making, previously described as stage 4 where all the vacant spaces are occupied by an urban farming system regardless of found items but based on the number of missing items. If this feature is activated then the average symbiotic rate goes down however full potential of the site is utilised for higher yields.

Design Panels

After the questionnaire, there are multiple pages to visualise the design decisions. Assigned systems are visualised in plan view in addition to the waste exchange between nodes. The plans can be viewed closely by zooming in. Each consequent page relates to the search radius and whether all the vacant spaces are occupied. The symbiosis rate and yields for each assigned system can be viewed by clicking on the vacant space. Similarly, the type and quantity of used waste sources can be viewed. On these panels, design rules can be viewed and modified. After modifying the rules, Foodcyle needs to run again to redesign. (See appendix C.8 for user interface)

Customisation

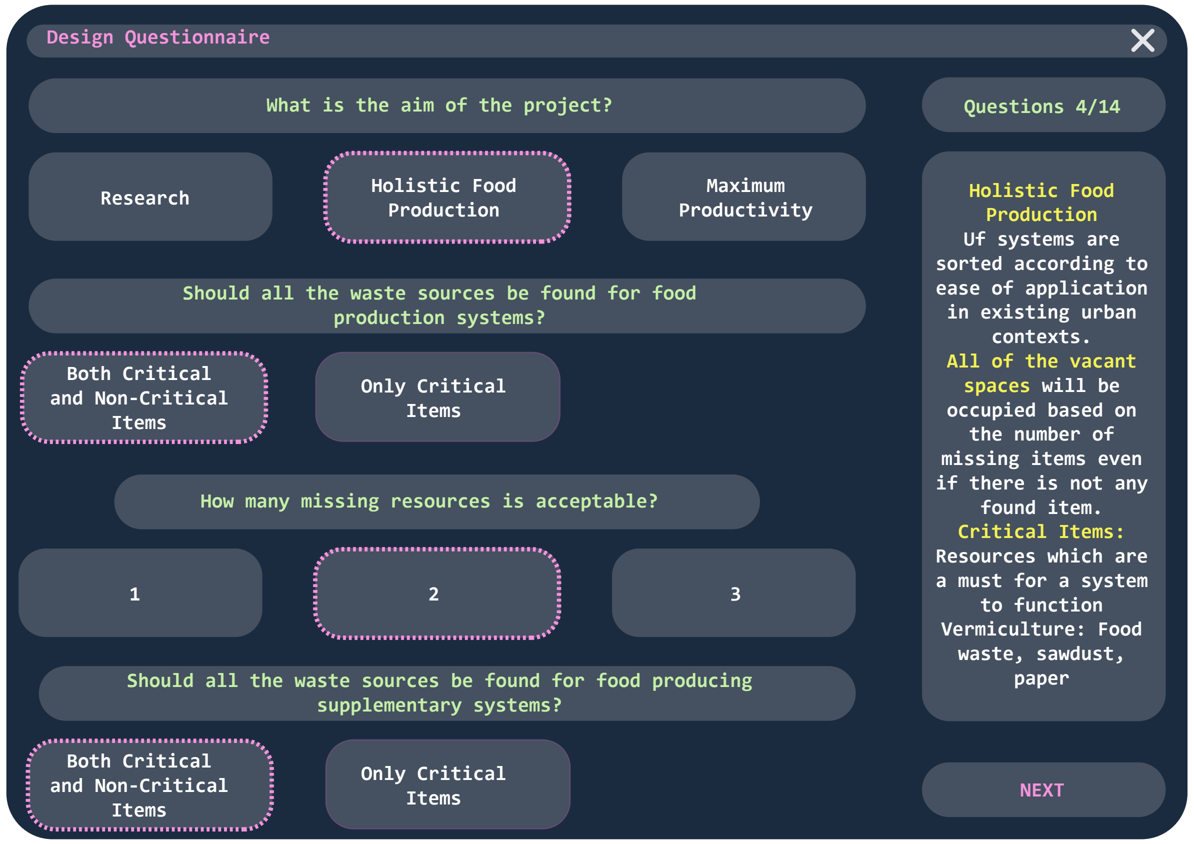
There might be scenarios where more than one system is suitable for the same space. In that case by default, the system highest on the list is assigned to vacant spaces. However, in this panel, those possible options can be provided along with advantages and disadvantages to let the designer decide.

Instead of giving a definitive urban farming solution to the designer, the decision-making tool gives a range of potential systems so that the designer can make informed decisions regarding the system selection based on the information given such as symbiosis percentage, food production rates of urban farming systems per m2 and simplicity of systems. The symbiosis percentage is based on how many of the needed inputs are found while the food production rates are based on previously shown table 4.1. By default, the decision-making tool will suggest the option with the highest symbiosis percentage if the project aim is holistic food production in urban contexts.

The designer might prefer to change the assigned system to another system for aesthetical or practical purposes. In order to entertain such scenarios, in this panel, the user can change one system to another and the tool responds to that change by redesigning the site based on the change with the default setting. The tool can give a warning prior to redesigning if that system is challenging structurally or due to solar exposure of the space. If the site is redesigned, the tool can give an overview of what would change if the initially suggested system is swapped in terms of average symbiotic rate, the total amount of used waste, number of exchanges and yields.

Breakdown of Results:

After the decisions are set, an analysis of the system selections and waste use is provided which can be used as an indicator of design performance. Each waste type and how much of it is used is visualised with pie charts. How many of which urban farming system is assigned is visualised in addition to the yields of different products such as mushrooms, fish, leafy greens, soft fruits and earthworms. Finally, a comparison of what percentage of the residents' needs can be compensated by these farms is roughly calculated in order to quantify the impact in an easily understandable way.



Design Questionnaire

What is the aim of the project?

Research [✓ Holistic Food Production] Maximum Productivity

Should all the waste sources be found for food production systems?

[✓ Both Critical and Non-Critical Items] Only Critical Items

How many missing resources is acceptable?

1 [✓ 2] 3

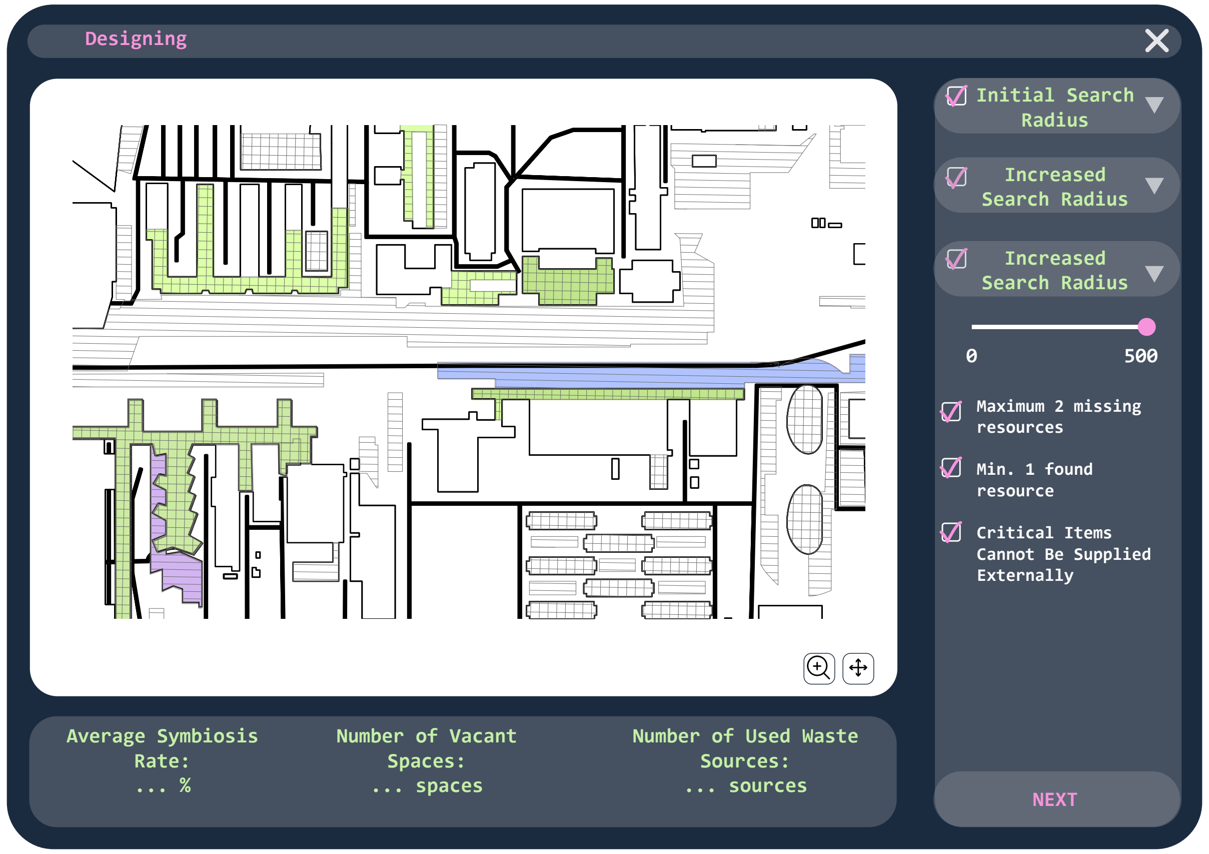
Should all the waste sources be found for food producing supplementary systems?

[✓ Both Critical and Non-Critical Items] Only Critical Items

Questions 4/14

Holistic Food Production
Uf systems are sorted according to ease of application in existing urban contexts.
All of the vacant spaces will be occupied based on the number of missing items even if there is not any found item.
Critical Items: Resources which are a must for a system to function
Vermiculture: Food waste, sawdust, paper

NEXT



Designing

☑ Initial Search Radius ▼

☑ Increased Search Radius ▼

☑ Increased Search Radius ▼

0 500

☑ Maximum 2 missing resources

☑ Min. 1 found resource

☑ Critical Items Cannot Be Supplied Externally

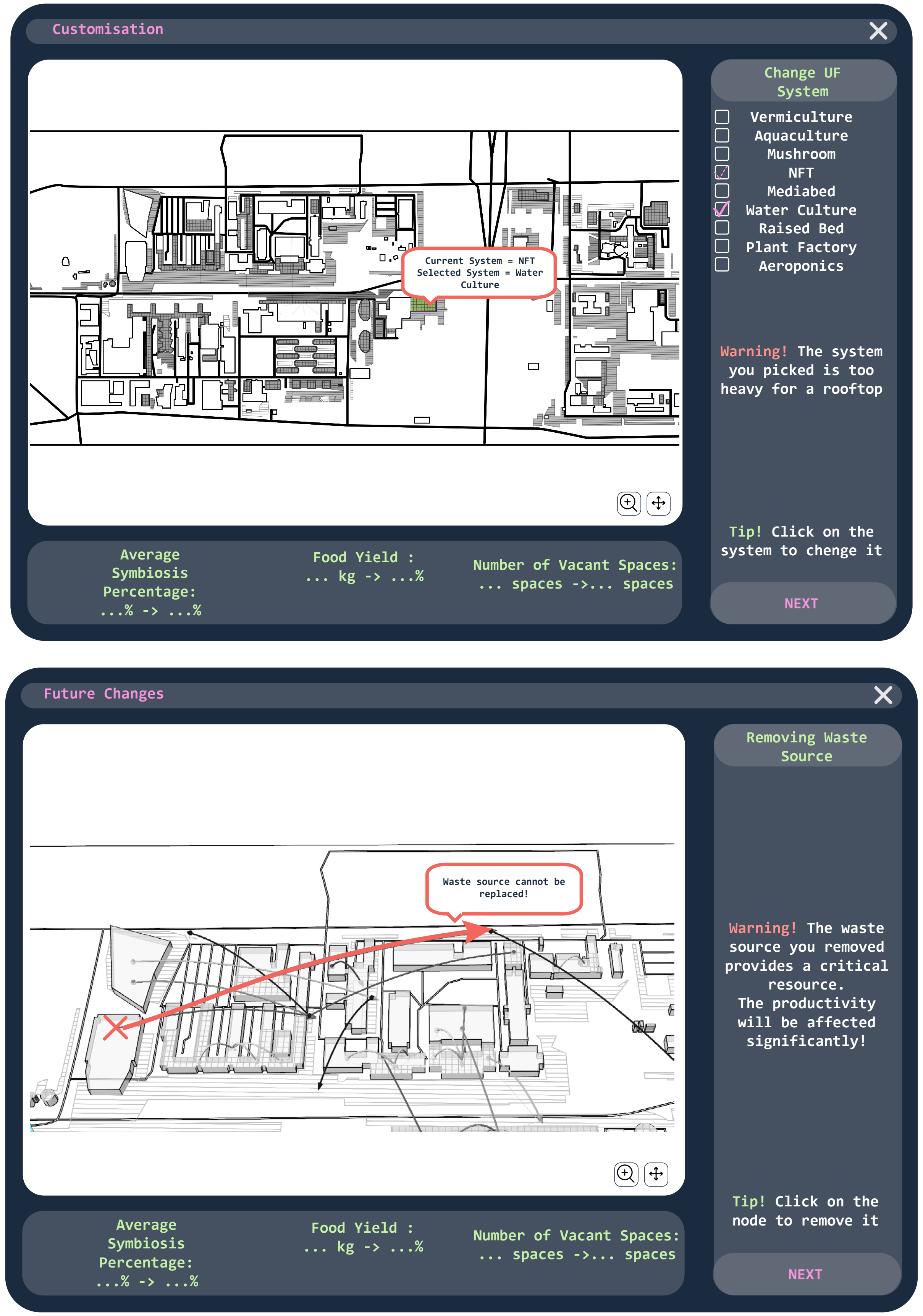
Average Symbiosis Rate: ... %

Number of Vacant Spaces: ... spaces

Number of Used Waste Sources: ... sources

NEXT

Figure 5.4 User Interface: Questionnaire (top), Design Panels (bottom) (see Appendix C.1 for other pages)



Customisation

Current System = NFT
Selected System = Water Culture

Change UF System

☐ Vermiculture
☐ Aquaculture
☐ Mushroom
☑ NFT
☐ Mediabed
☑ Water Culture
☐ Raised Bed
☐ Plant Factory
☐ Aeroponics

Warning! The system you picked is too heavy for a rooftop

Tip! Click on the system to chenge it

NEXT

Average Symbiosis Percentage: ...% -> ...%

Food Yield : ... kg -> ...%

Number of Vacant Spaces: ... spaces ->... spaces

Future Changes

Waste source cannot be replaced!

Removing Waste Source

Warning! The waste source you removed provides a critical resource.
The productivity will be affected significantly!

Tip! Click on the node to remove it

NEXT

Average Symbiosis Percentage: ...% -> ...%

Food Yield : ... kg -> ...%

Number of Vacant Spaces: ... spaces ->... spaces

Figure 5.5 User Interface: Customisation (top), Response To Future Changes (bottom)

Responsive to Future Changes (In theory)

The built environment is not static and subject to change therefore the decision-making tool can be used to evaluate the effect of removing one or more waste sources or vacant spaces. In this panel, the designer removes the waste sources. The tool can give a warning if the resources cannot be compensated for. If it can, it recommends which waste output to use instead.

Software as a Service (In theory)

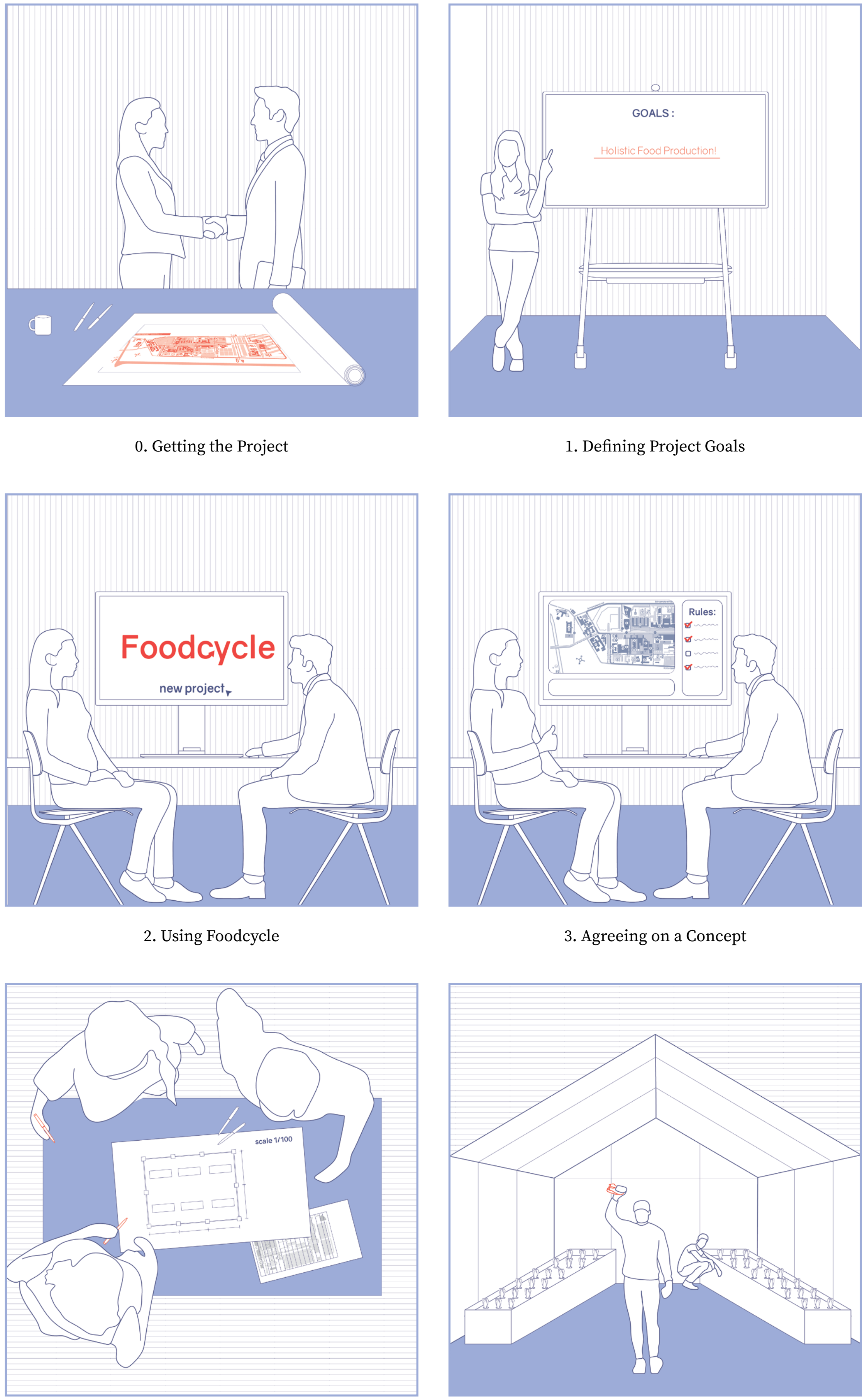
As previously mentioned, data can be collected by using drone footage with deep learning or using ArcGIS. These features relate to the data collection method preference in the interface. Deep learning can potentially be a feature of the decision-making tool however access to drones might be challenging for a designer. If drone footage is selected it could be included in a plan where employees operate the drone and data collection can be provided by Foodcycle upon request. If GIS data is selected and if the data is readily available then GIS maps can be used within the interface without requiring any former GIS experience. Apart from these methods, the decision-making tool can be open source. In conclusion, Foodcycle can be more than a decision-making tool by providing service.

Foodcycle’s Role in Design Process

Foodcycle is designed to give a set of options regarding urban farming systems based on existing conditions. Therefore, it cannot be used on its own as a design tool but a strong collaboration between the tool and designers is needed. The tool is aimed to be used for otherwise laborious tasks such as iterating numerously over the vacant spaces, potential systems and nearby waste sources. As illustrated in figure 5.6, the design process starts with getting the project and followed by defining project goals. These can be holistic food production, maximum food production or research. These goals change the prioritisation of systems as described in section 4.1.2. Afterwards, Foodcycle is used, the design rules are decided on and these rules are customised until the parties are satisfied with the results. Since the tool does not design the farm itself, specialists need to design the buildings, the systems and prepare the business plan. These tasks can be performed by using other decision making tools such as Agritecture Designer or Delphy QMS. A comparison between these and Foodcycle will be made in chapter 7. During the design of the buildings, a participatory design approach can be taken which encourages stakeholders to engage in the design. Last step is to build the farms and operate them.

Transportation of Waste and Supplements From Source To Farm

Even though, how the farms operate is not within the scope of this research the journey of waste is foreseen. Transportation of water, CO2 and heat should be done using pipes and mechanical systems. However, some waste types like food waste, spent coffee grounds, paper and sawdust can be transported rather simply between locations. The journeyof waste starts with producing the waste either by having a sandwich or a cup of coffee from the cafe, or being handed a piece of flyer or making a wooden model in the maquette hall. All these activities simply produce waste at some point along the duration of the activity. These different types of waste can be separated and collected as illustrated in figure 5.7. Since the corresponding farms to feed the waste into are designed to be in a limited radius, these waste containers can be transported easily with a bike or an electric bike. Using containers instead of plastic bags can be beneficial by reducing the plastic waste. Depending on the nature of the farm, waste may need to be mixed, sterilised or precomposted. These processes can take place in the farm. Next step is to use the waste as a resource for the farms whether it is mushroom production or vermicompost. The journey of waste ends when it is utilised but a new cycle starts when the products of the farms are used.



0. Getting the Project

1. Defining Project Goals

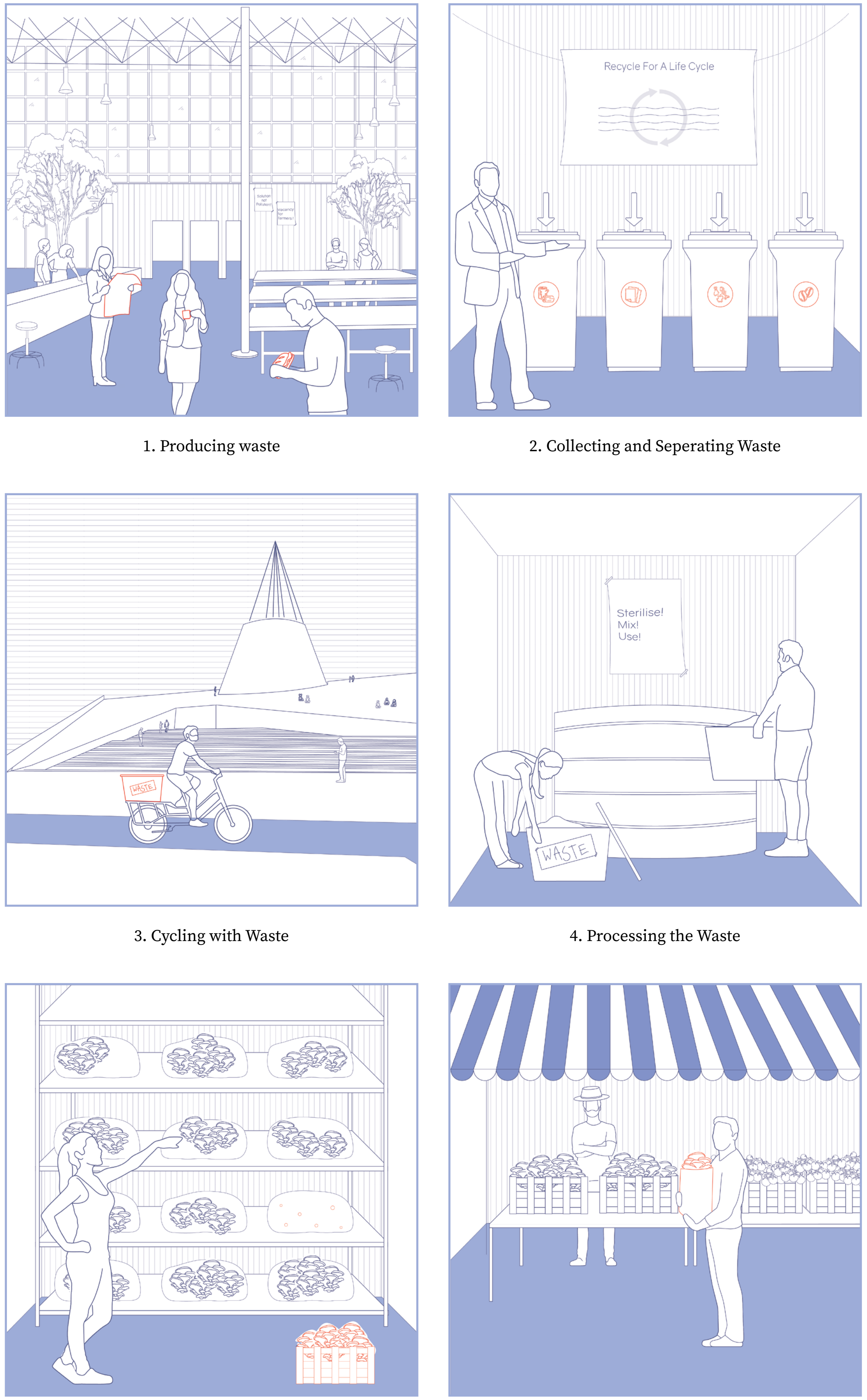
2. Using Foodcycle

3. Agreeing on a Concept

4. Detailed Design

5. Operating & Cultivating

Figure 5.6 Design Process Using Foodcycle



1. Producing waste

2. Collecting and Seperating Waste

3. Cycling with Waste

4. Processing the Waste

5. Growing and Picking Produce

6. Selling and Consuming

Figure 5.7 Journey of Waste

06

CASE STUDIES

6.0 CASE STUDIES

The decision-making tool is designed to work for a wide range of locations with temperate climates. Therefore, the tool can be used universally in order to make urban farming decisions as long as the necessary data is available. Since this research aims to use waste sources on-site instead of sending them away to landfills, incineration or repurposing facilities, the effects of using waste for farming need to be illustrated. Consequently, one primary case study and 1 secondary case study are conducted. The primary case study is TU Delft Campus while the secondary case study is Urban Greenhouse Challenge's site in Washington.

6.1 TU Delft Case Study

TU Delft Campus is selected as the primary case study to test and assess the results parallel to the Climate Action Programme of the university. TU Delft aims to be carbon neutral, circular and contributing to the quality of life by 2030. (Sustainability at TU Delft, n.d.) Integrating urban farming into the campus while using campus waste flows as resources for the farms has the potential to serve as a waste management strategy and food production strategy in addition to increasing the biodiversity on campus and potentially providing social impact. These farms can provide job opportunities, and increase food safety. However, to what extent these farms can provide for Delft and reduce waste needs to be assessed.

6.1.1 Site Analysis

The research regarding the campus started with the site analysis in order to determine the vacant spaces and waste outputs. The data collected in this part is used for decision making afterwards.

Vacant Spaces

The analysis regarding vacant spaces is done by using a manual data collection method. A satellite view of the campus is utilised in order to determine vacant rooftops and ground floor areas. Vacant spaces are defined as spaces which do not have any other designated purpose. In order to determine these on the map, non-vacant spaces are mapped first such as buildings, sports fields, parking lots, bike storage, roads, canals, and other site-specific functions.

Criteria For Roof Tops:

To begin with, rooftops are mapped and assessed based on their availability. For a rooftop to be considered vacant, there are a few criteria:

- It should be a flat roof
  - It should not be used for any other purpose including:
    - Solar energy collection
    - Mechanical equipment
    - Skylights

Criteria For Ground Floor:

After mapping the non-vacant spaces with designated functions, it is concluded that there are a lot of green vacant spaces on the campus. The question "Should all greenery be productive greenery?" is asked. Therefore a new criteria needed to be introduced which defines in what conditions greenery can be considered vacant. In order to limit the green areas included as vacant spaces, only green areas which are a maximum of 30 meters away from the buildings are defined as vacant. The reason behind this criteria is to place urban farms where the users can see and interact with them rather than placing them away from users.

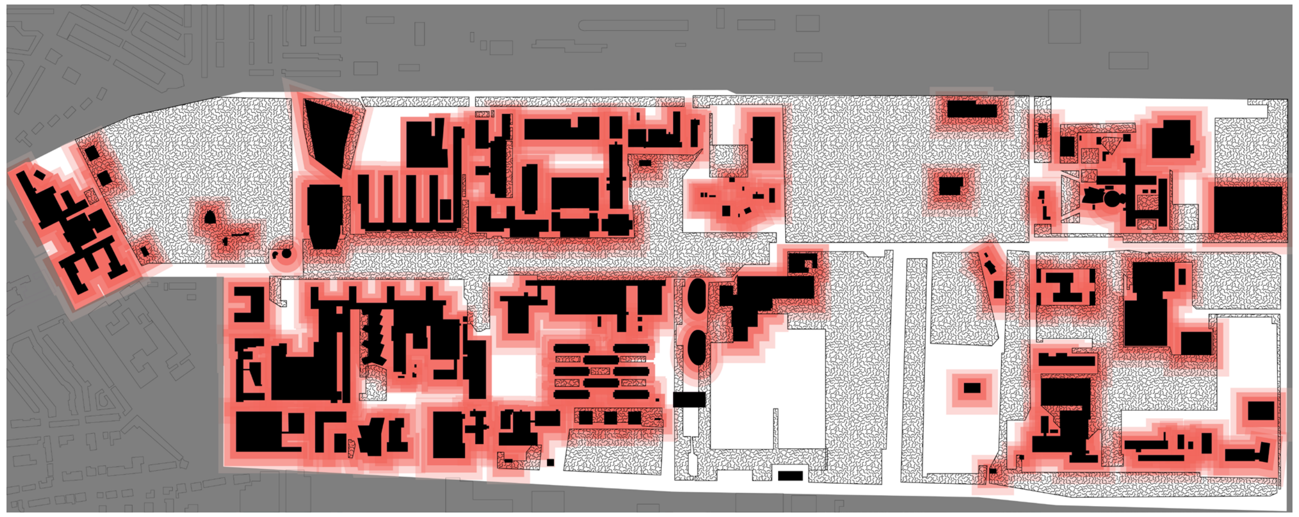


Figure 6.1 Radiant Vicinity Map (10m - 20m - 30 m)

Waste Outputs

The next dataset that needs to be gathered is waste outputs including their locations and quantities. This part of the analysis is based on the site analysis framework described in section x. Cafeterias, coffee parlours, model making spaces, conference halls, sloped roofs, data centres, and supermarkets are mapped. Information regarding food and coffee establishments is based on the Food&Beverage map of the campus. Conference halls' locations and people capacities are based on Education Spaces Viewer provided online by the university. After determining the locations of waste sources the next step is to determine the quantities of waste. Regarding the quantities, two methods are followed. The first one is contacting the catering providers directly while the second method is assuming the waste quantities based on simple calculations. It should be noted that some waste sources are not included as including them would overcomplicate the data collection. For instance, there are multiple coffee machines in each building on campus and these coffee machines are not included in the analysis. In addition, even though cafeterias also serve coffee they are considered as food waste sources only. If these simplifications are not made quantities of waste produced by each output point need to be known since assuming waste quantities for each coffee machine would be a major challenge.

Food Waste

Food waste is described previously as edible and non-edible parts of the crops therefore it covers kitchen waste and agricultural waste. However, in this analysis, only food waste from cafeterias is taken into account. Some of the data regarding the quantities are collected by contacting the catering company of the Faculty of Architecture & the Built Environment and Aula Cafeterias, Cirfood. Cirfood only started auditing waste a few years ago coinciding with the start of the Covid-19 pandemic. Therefore, the data does not reflect the reality as the campus and cafeterias were closed for a while. In order to, have realistic numbers as much as possible the maximum quantities are utilised. In addition to these two cafeterias, there are smaller food joints. The food waste regarding these are assumed based on the faculty population estimations, the time students are expected to spend on campus and municipal waste quantities per capita provided by Statline. For some locations such as X (TU Delft's Sports and Activity Center), food trucks or independent enterprises, it was not possible to assume or reach data therefore these locations are not included in further steps. (See appendix E.10 and E.11 for details)

Total Food Waste = (Food Waste kg per Capita) x (Faculty Population)
Faculty Waste = (Total Food Waste) x (Time in School %)

Coffee Waste

According to the food and beverages map, there is at least one espresso bar in each faculty and there are a few other cafes in the Faculty of Architecture & the Built Environment, Aula, Pulse and Central Library. For the smaller coffee parlors in faculty buildings, Pulse and Central Library it is assumed that everyone would have 1.5 cups of coffee a day. And 11 grams of coffee grounds are used per cup (Biobean,2019) And faculty buildings are assumed to be used for 5 days during 40 week-long education period.

Total Coffee Waste = (Faculty Population) x Daily Coffee Consumption Per Student x School Year x Spent Coffee Grounds per Cup

Sawdust

Sawdust from the Model Hall in Architecture and Built Environment faculty (BK) is collected and sent to companies to make pressed wood products. According to the officials in the model hall, 2 20L bags of sawdust are collected per week during the school year. This volume corresponds to 3360 kg sawdust annualy as illustrated in table 6.1. Even though, this waste product on the campus is valorised and used for another purpose it is included in the analysis for research purposes.

Saw Dust	Annual Waste Volume (m3/ year)	Sawdust Density	Annual Sawdust (kg/year)
BK	16	210 kg/m3	3360

Table 6.1 Sawdust Output of BK

Paper

Paper waste quantities are estimated based on the population of each faculty, the time students are expected to spend on campus and municipal data on paper waste per capita. According to the statistics, 37 kg of paper waste was produced per capita in 2019. (Statistics Netherlands, 2021) Even though there are multiple paper waste bins in each faculty, only one waste source is placed in the building to simplify the calculations.

Total Paper Waste = (Food Paper kg per Capita) x (Faculty Population)
Faculty Waste = (Total Paper Waste) x (Time in School %)

Rainwater

Rainwater can be collected by roofs and in this research only sloped existing roofs are considered as rainwater sources. The surface areas of roofs are multiplied by the average rainfall in Delft which is 900mm per annum. (Geodan, n.d.)

Annual Rainwater Harvest (L) = Annual Rainfall (mm) x Roof Area (m2)

Residual Heat

There are two residual heat sources on campus one is a supermarket while the other one is a data centre. The amount of heat emitted by the supermarket is based on the electricity demand of 401 kWh/m2 for cooling, assuming a COP of 3. These values are multiplied by the surface area of the supermarket to calculate the total cooling load which corresponds to the residual heat. (Tess Blom, 2018) The Delft Data Center Group produces 31.050GJ (8625 MWh) heat annually as calculated by the same author.

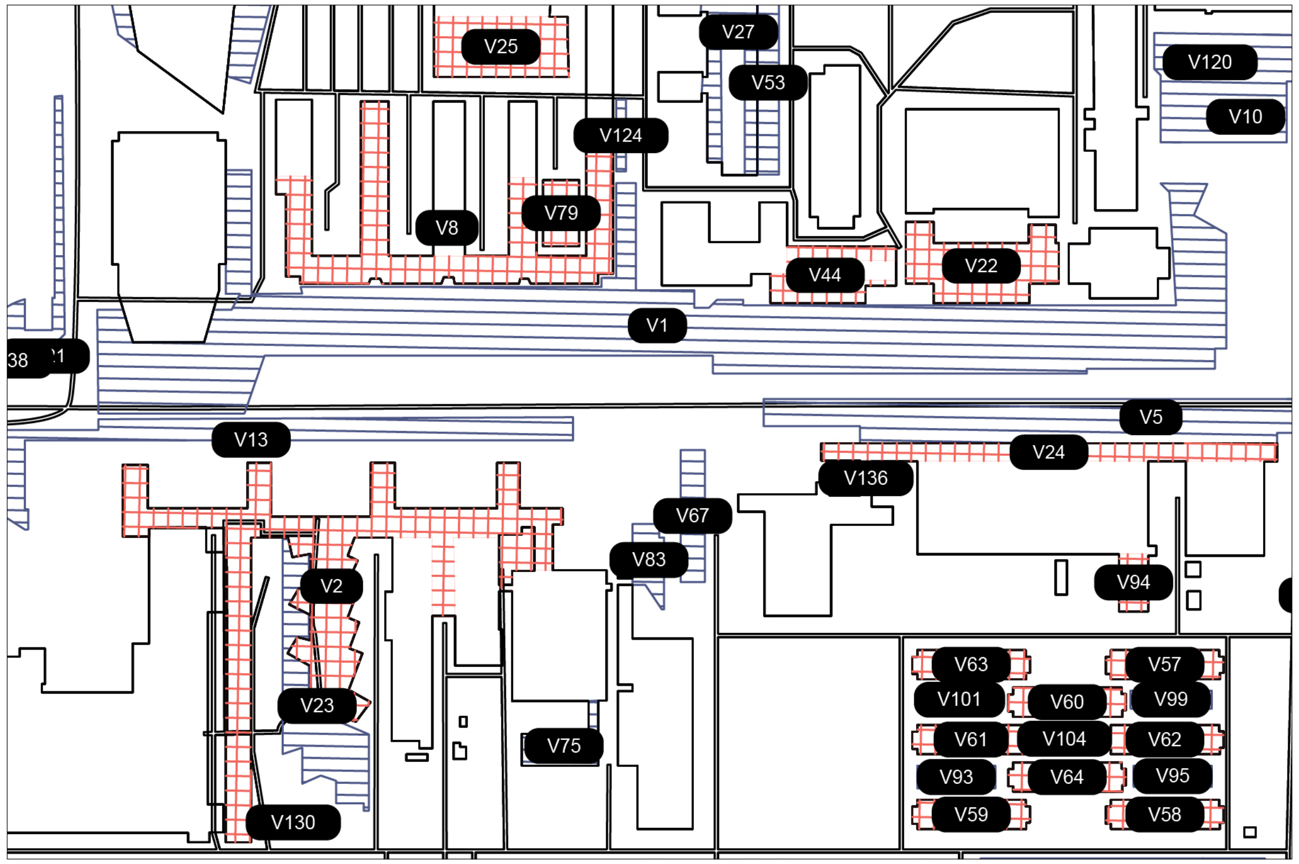


Figure 6.2 TU Delft Vacant Space Nodes (see appendix E.9 for the complete map)

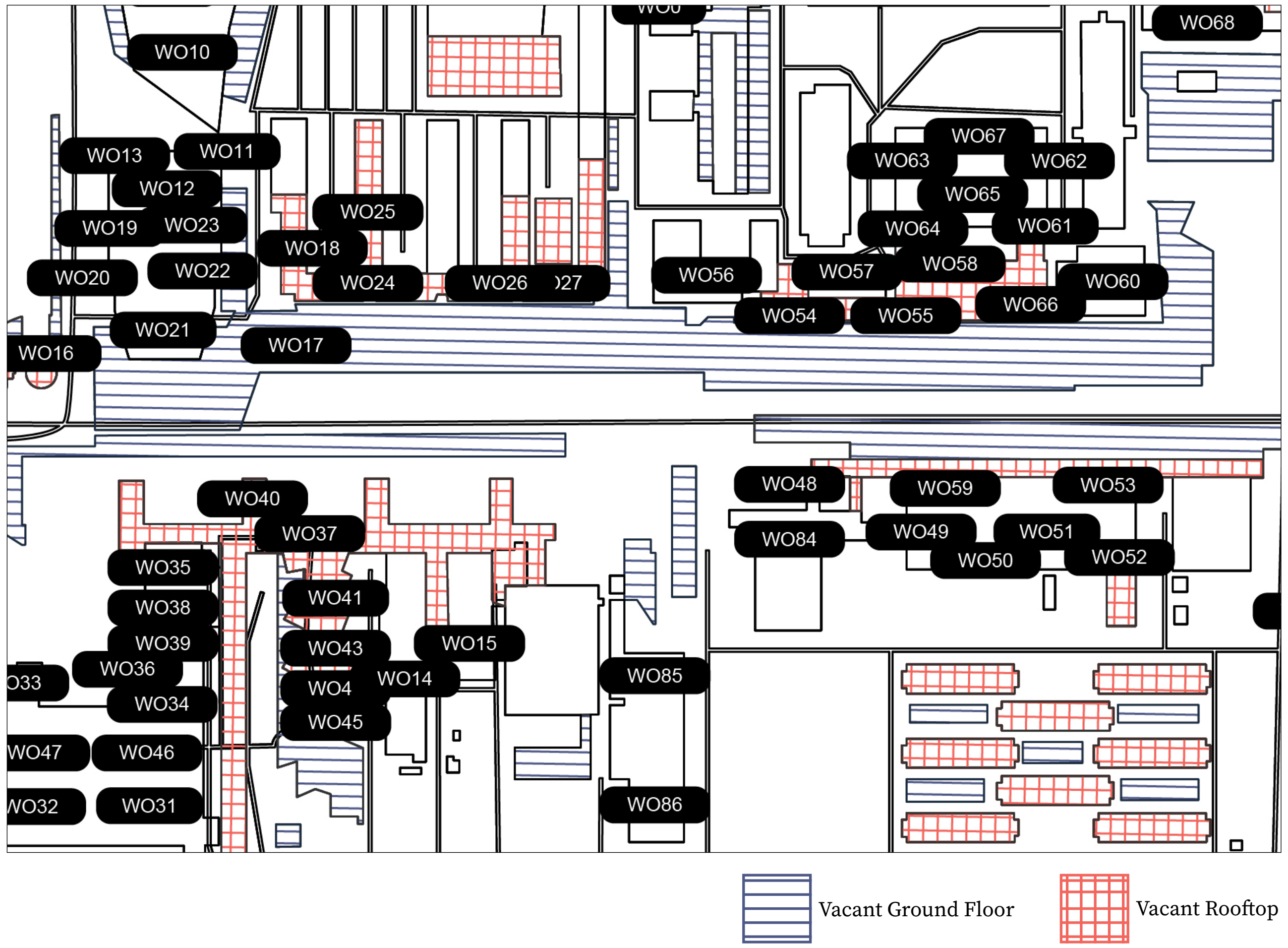


Figure 6.3 TU Delft Waste Output Nodes (see appendix E.11 for the complete map)

Vacant Space & Waste Output Nodes

After collecting the data all the vacant spaces and waste sources are mapped to start the decision-making process as shown in figures 6.2 and 6.3. In this stage, identifiers are given to nodes and spreadsheets are filled in after exporting the identifier and location data in xls. format. Then the waste quantities are filled into the spreadsheet. (See appendix E.9-E.12 for complete tables) These datasets are used for decision making.

6.1.2 Case Study Settings

Prior to running the tool, a set of decisions need to be made including search radius, whether the search radius can be increased and how many times as well as whether all the vacant spaces should be occupied. For the TU Delft Campus case, a search radius of 100 m, 200 m and 500 m are used. During the first 3 stages, a system is assigned only if a minimum of one resource is found, with a maximum of 2 missing non-critical resources. Foodcycle is run 3 times while the search radius is increased step by step. It is observed that not all the spaces are occupied. Therefore to illustrate the food production potential of the campus, stage 4 design rules are applied. In stage 4, Foodcycle looks for at least one found resource first. If there is a found item and if the missing resources are not critical for that system's productivity, the system gets assigned to the vacant space. And the decision-making tool runs again to assign food-producing supplementary and supplementary systems. Lastly, as there were still unoccupied spaces left, the rules are eased even more and a system is assigned even if there are no found resources based on the number of missing items. Foodcycle proceeds to assign food-producing supplementary and supplementary systems. After this stage, the yields, system decisions and used waste quantities are calculated.

6.1.3 Case Study Conclusions

The last stage is to illustrate the decisions both visually and quantitatively. During the first stage, 13 vacant spaces are occupied and 29 waste exchange connections are created within a 100 m radius.

Daily Production of Leafy Greens	22811.24356
Recommended Daily Fruit & Vegetable Consumption (kg)	0.25 (RIVM, 2017)
People #	91244.97425
Delf Population	103581 (Statistiek, n.d.)

Table 6.2 Total Yield of Leafy Greens

Stage	**Symbiosis Rate % of Decisions**
Stage 1 (100 m)	52.56410256
Stage 2 (200 m)	50
Stage 3 (500 m)	50
Stage 4 (500 m + occupy all)	13.1969697
Total	17.88

Table 6.3 Average Symbiosis Rate At Each Stage

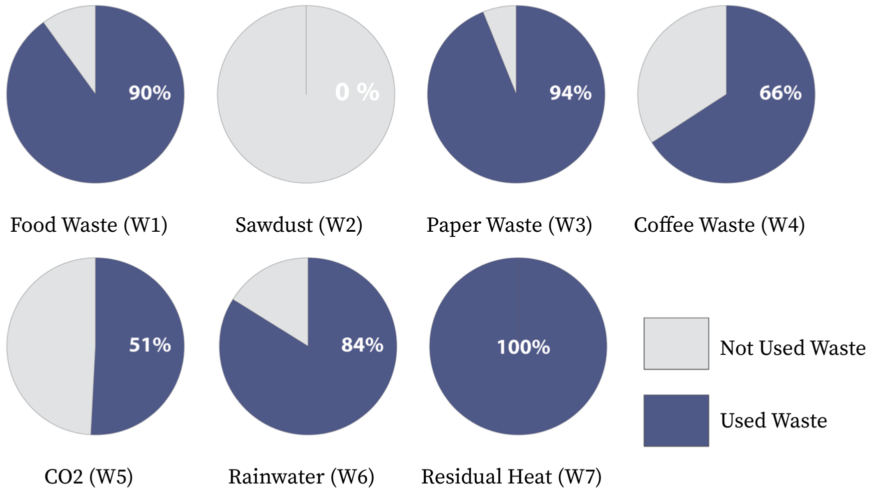


Table 6.4 Used Waste Quantities & Percentage

The majority of the systems assigned are NFTs, while mostly CO2 is used as a resource. The average symbiosis rate is close to 53% in this stage relating to how much of the necessary waste sources are found for assigned farms. On the other hand, in stage 2, 1 vacant space is occupied additionally while 3 more waste exchange connections are created within a 200 m radius. In this stage, a mushroom farm is assigned while coffee waste is exchanged. In the 3rd stage, another mushroom farm is assigned. After easing the design rules, mostly raised beds are assigned to ground-floor spaces since raised beds only have one input which can be supplied externally. In the 2nd and 3rd stages symbiosis rates of decisions made in the respective stages decrease to 50% however after stage 4 it drops drastically to 17%. (see Table 6.3) By utilising the decision-making tool, 125 vacant spaces are used for farming purposes reaching a total of 22.18 hectares. The most assigned urban farming system is NFTs with 62 farming locations while raised beds are second on the list with 40 locations due to the lightweight of NTFs and limited waste demands of raised beds as illustrated in figure 6.4. Other than these, 9 vermicomposting areas, 2 aquaculture spaces, 9 mushroom production areas and 1 plant factory are suggested. By covering almost 22 hectares of the campus, enough vegetables for almost 88% of the Delft population can be produced at TU Delft Campus as shown in Table 6.2. These numbers are based on daily fruit and vegetable consumption recommendations by the government. Lastly, while producing food, more than 50% of waste of each waste type, except sawdust, can be used on campus. (see Table 6.4) Lastly, from a more architectural point of view more productive greenery can be incorporated to the campus encouraging people's engagement with food production as illustrated in figure 6.6. Food production can even function as an attraction in some cases and increase awareness.

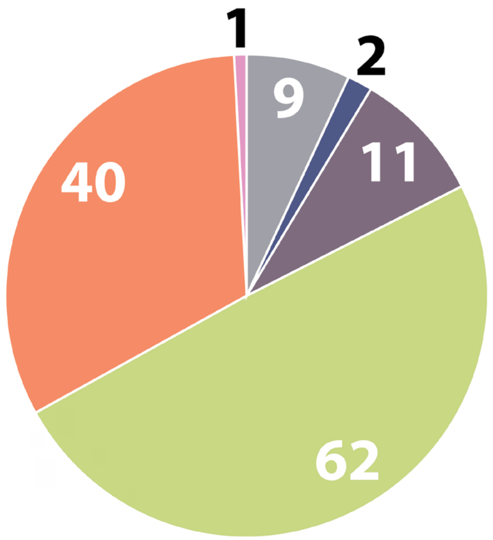
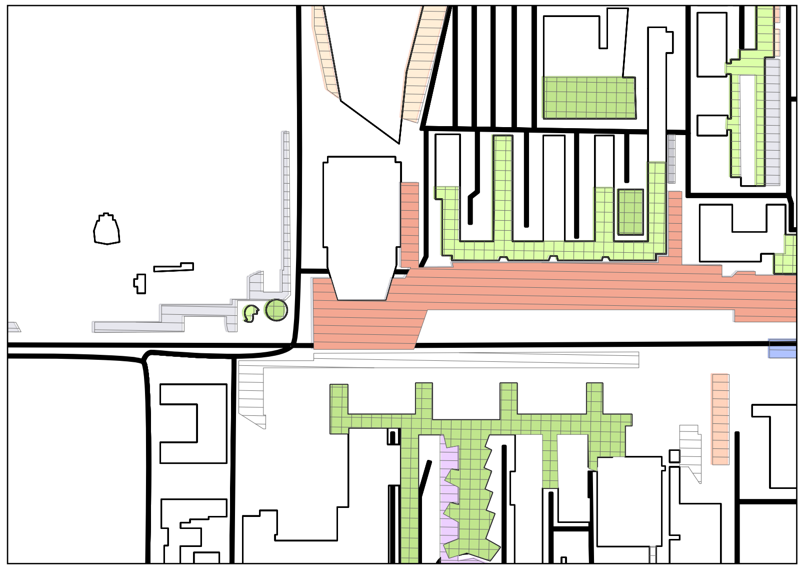
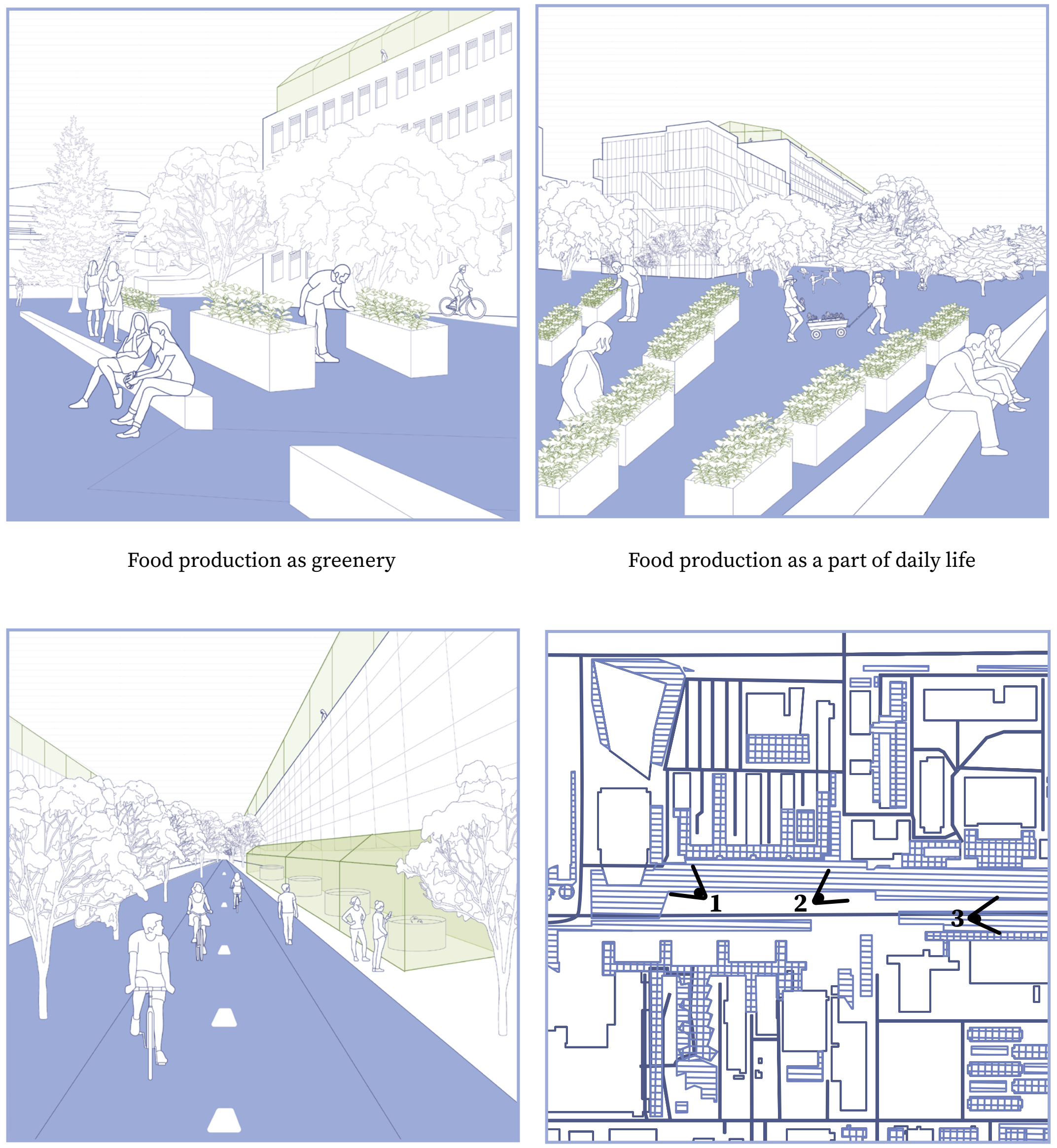


Figure 6.4 Assigned Systems and Distribution of Produced Crops



- 🟨 Aquaponics
- 🟪 Plant Factory
- 🟩 Water Culture
- 🟧 Raised Bed
- 🟩 Media Bed
- 🟩 NFT
- 🟪 Mushroom Farm
- 🟦 Aquaculture
- ⬜ Vermicompost

Figure 6.5 Stage 4 Design Decisions (See appendix F.1-F.8 for the full map and each stage)



Food production as greenery

Food production as a part of daily life

Food production as an attraction

Figure 6.6 TU Delft Campus Producing Food

6.2 East Capitol Farm, Washington DC

As a second case study Urban Greenhouse Challenge's site is selected. The challenge is organised by Wageningen University and this year focuses on making a social impact through urban farming in Ward 7 of Washington DC. Ward 7 is one of the most underprivileged wards in the city in terms of social-economic status as well as food security. Since the site is not fully developed and not an existing condition, the analysis is based on TU Delft's student team's design taking place in the competition. The waste quantities are not available therefore simplified waste quantities are estimated first.

6.2.1 Case Study Settings

In this case study, the preferences are kept the same as in the primary case study. Therefore, the maximum amount of missing resources is 2, while at least 1 resource should be found in the first stage. Since there are not many waste sources around only one search radius of 700 m is tested. Since this is a relatively small site and it is based on a design concept, the rule regarding non-transferable items is lifted. With a 700 m search radius, the furthest waste sources can be included in the decision making. After running the tool it is observed no waste exchanges are generated, due to a mismatch of waste quantities and vacant space sizes. In order to assess the full potential of the site, the waste quantities are manipulated to match most of the vacant space sizes.

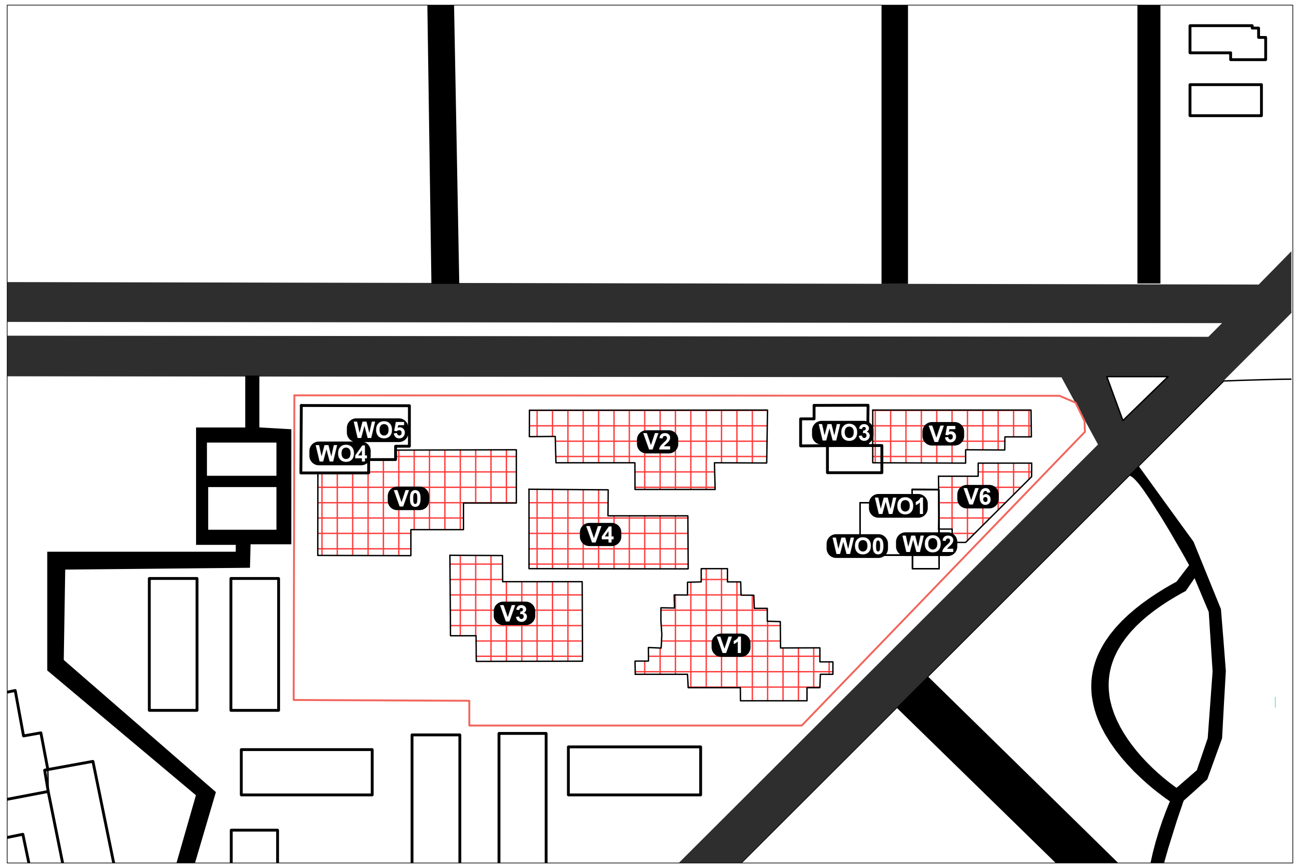


Figure 6.4 Ward 7 Vacant Space & Waste Output Nodes (see appendix G.1 for The Complete Map)

6.2.2 Case Study Conclusions

As a result of decision-making, NFT systems are assigned to 3 locations in stage 1. In this stage, a total of 3 waste exchanges are generated carrying rainwater to NFTs. After the "occupy all the spaces" feature is activated, 2 farms with raised beds are generated with dependency on the external supply of water. 2 locations are not assigned any system due to their solar exposure. Their solar exposure is medium (2) due to their orientation which is only sufficient for aquaponics. However, aquaponics cannot be assigned to those spaces by the prototype because the sizes of the supplement-needing spaces are not in the same range as the available spaces. Even if the design rule regarding vacant space and waste source size is eased further adjustments need to be done to relax the design rules since all the resources should be available nearby to assign food-producing supplementary and supplementary systems. It can be concluded that Foodcycle works better with bigger data sets with many waste sources of different types and quantities in addition to vacant spaces with varying characteristics.

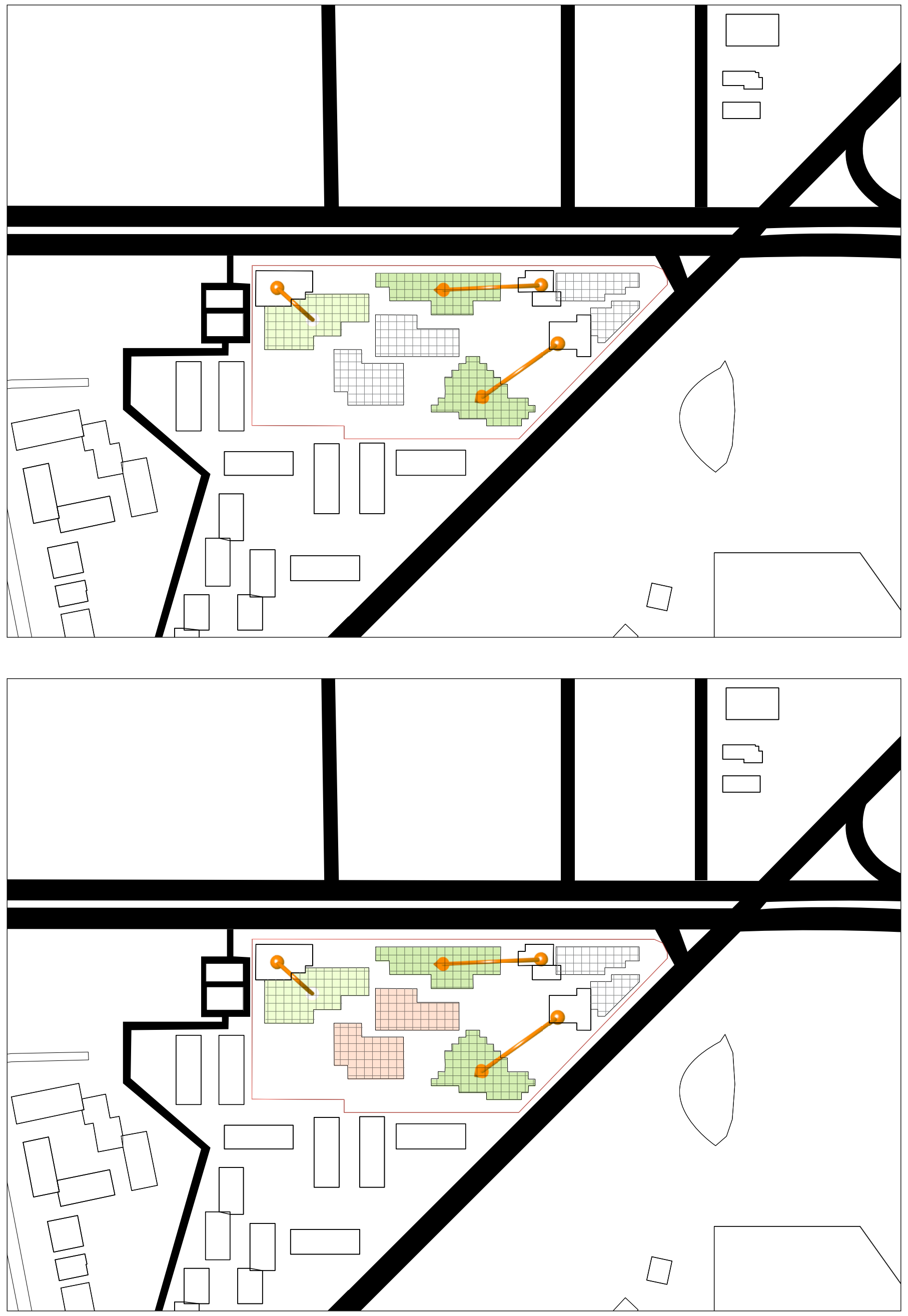


Figure 6.5 Ward 7 Design Decisions In Stage 1 (top) and Stage 2 (bottom)

07

CONCLUSIONS & AREAS OF FURTHER DEVELOPMENT

7.0 CONCLUSIONS & AREAS OF FURTHER DEVELOPMENT

In the previous chapters, the research framework, the literature review, the design task and methodology and case studies are described. Testing Foodcycle on case studies of different scales lead to some findings. Firstly, Foodcycle generates a more complex network with bigger data sets due to strict design rules and availability of waste. If the tool is utilised for a smaller data set with fewer vacant spaces and fewer waste sources then the design rules need to be eased. Otherwise, a network may not be generated mainly due to the disproportion between waste quantities and space sizes. However, for a larger set of data like TU Delft Campus since there are more possibilities the design rules do not obstruct the decision making instead they ensure viable solutions. Secondly, as demonstrated by TU Delft Campus case study urban settings have the potential to produce vast quantities of food while using readily available waste sources. If underutilised spaces are employed as potential urban farms, 22 tonnes of leafy vegetables can be produced on campus which is enough for 88 % of Delft on daily basis. Lastly, one of the most noteworthy findings of this research is if the design rules are too strict, they do not allow for decision-making. Therefore, either these rules need to be eased gradually with each stage of decision making or these rules should be customised for the context. For instance, in the Washington site, there are not many waste sources nor many vacant spaces. Therefore aiming to match space sizes to quantities obstructs the decision making. For such cases, this rule can be eased by looking for waste quantities bigger than space size to ensure enough resources. It should be noted that the design rules described in chapter 3 are established in order to ensure efficient use of resources and are based on a simplified representation of space characteristics and waste quantities.

Foodcycle is designed to choose an urban farming system based on existing conditions such as waste flows and underused spaces, as well as to generate a network of farms. It does what it is aimed for efficiently and fast. Another strength is that the reasoning behind each decision can be tracked. From a technical point of view, even if the tools used for Foodcycle such as python and grasshopper become outdated in the future, the methodology remains valid. And the same methodology can be used in the background of the userinterface. Currently, the prototype gives 1 system option for each vacant space however with further developments it can be designed to give a set of options to let the designer make an informed decision.

Due to the time constraints and the scope of the project, there are aspect which Foocycle does not consider as listed in figure 7.1. For instance, financial concerns are not considered even though they are largely influential in projects. Currently, it gives a breakdown of how much food can be produced and how much of the existing was is repurposed on site. However, it should be noted that these calculations are simplified and assumes that 100% of the vacant space's area is used for food production and weather conditions are always optimal for food production. Therefore prior to designing the spaces themselves, a more precise calculation can be done. Moreover, the decision-making tool does not optimise the network but iterates over spaces, systems and wastes to assign a system. Therefore, the next system decision is always influenced by the previous decisions. There might be a more symbiotic design however the prototype does not look for the best design currently. It is undoubtful that further improvements are needed in certain areas as shown in figure 7.1. However, all of the improvements can be done as an addition to the tool with ease.

To conclude, such a combined approach to waste management and food production has potential as demonstrated by the TU Delft case study. The aforementioned potential includes social impact, valorisation of the vast amount of waste on-site, reducing food miles and reducing emissions associated with food production and consumption. By addressing the weaknesses of Foodcycle, it can be improved and be carried to the next level.

STRENGTHS

- Decision making for large datasets
- Adaptable design rules
- Applicable to every site
- It suggests urban farming systems for vacant spaces while utilising existing waste flows
- Relatively fast decision making
- It can assist designers in making decisions
- Results can be compared to each other after changing the design rules
- The logic behind it is not dependent on specific software
- It can be used along with CAD software which is familiar to designers
- Simple data types (.txt files, xsls. files) are used to transfer data
- Easy to interact with by using the interface
- Decisions and reasons behind them can be tracked easily
- Explanations of the terms and questions are available in the interface

WEAKNESSES

- Decision making for small data sets
- Strict design rules to work with if available waste and space are limited
- It is not an optimisation tool
- Vacant space characteristics are simplified
- Waste quantities are simplified
- It doesn't include agricultural waste from assigned farms
- It assumes 100% use of the space for farming, does not include space for circulation
- It does not design the farm itself
- The design rules may need to be customised for the site
- It does not include financial aspects as criteria for decision making
- Fertiliser from mushroom farms cannot be used as a supplement with the current decision-making tool
- Currently, switching between software /platforms is needed

OPPORTUNITIES

- It can be the decision-making mechanism behind a new software or website
- It can be improved to accommodate excluded features
- As suggested, it can be used along with automated data collection methods
- It can give suggestions instead of definitive answers if the script is adjusted
- A collaboration with ArcGIS can allow for automated data collection and site selection

THREATS

- Existing conditions are susceptible to change
- Waste quantities can decrease drastically due to external reasons
- Waste sources can be removed or moved to another location
- The results can go over budget since financial aspects are not included in the tool
- The decision making tool may need to be scripted again if the current platform becomes unavailable
- The suggestions may not be financially feasible

Figure 7.1 SWOT Analysis of the approach

Comparison With Other Tools

In order to have an understanding of what Foodcycle is capable of and where it is lacking, it is compared to 2 online decision making tools. First one is Agritecture Designer, which is a tool mostly focused on developing business plans and comparing financial benefits of different concepts. The second tool is Delphy's QMS Tool which is available for different crops and is aimed to be used during the detailed design phase. Lighting, CO2, climate requirements are factored in to guide designers towards the most feasible design option. However, these tools aims are completely different than Foodcycle's goal. As illustrated in figure 7.2, there are features that Foodcycle is lacking however its features relate to the purpose off the tool and differ from existing tools. All these 3 tools are used in different stages of design. Foodcycle is used in preliminary stages while Agritecture Designer and Delphy are used in later stages of design. It is also possible to use these 3 tools together for a data backed design process.

Features	Agritecture Designer	Delphy - QMS	Foodcycle
Energy calculations	+	+	-
Yield estimation	+	+	+
Profit estimation	+	+	-
Business plan	+	+	-
Comparison between models	+	+	+
Using waste as a resource	-	-	+
Building a network of farms	-	-	+
Suggests growing techniques	-	?	+
Suggesting farming systems	+	?	+
Designing for more than one space at once	-	-	+
Different project aims	+	?	-
Designing with a budget	+	?	-
Different crops to select	urban/peri urban/ rural	?	urban
Site	+	+	-
Different design stages	+	-	-
Includes ease of running the farm	+	-	-
Concept report	+	+	+
Advice regarding farm design	+	+	-
Climate control	-	+	-

Figure 7.2 Comparing Foodcycle

Further development

As previously mentioned, the methodology and the prototype can be improved in many ways. Firstly, the real sizes, structural capacity, solar exposure and waste quantities can be used instead of simplified ranges. This would complicate the data collection process and decision making however would result in more realistic scenarios. In addition, the current lifecycle of each waste stream can be assessed prior to decision making. If a waste stream is already being valorised, it does not need to be included in the dataset. Secondly, an optimisation algorithm can be introduced to generate the most symbiotic design. In addition, the design rules for all kinds of urban farming systems can be eased gradually as discussed previously. This would increase the generated waste exchange connections. Waste products of urban farms should be included in the decision making in order to valorise agricultural waste. This would significantly increase the number of vermicompost on-site if the conditions are suitable. In addition, dividing waste into smaller portions can be one of the design decisions the user takes while using the tool. The search radius rule can be improved by drawing the maximum distance from the periphery of the space instead of from the geometric center of the space. This improvement would result in better search results when it comes to nearby waste as it excluded the space itself and includes a larger area of search. Lastly, in order to quantify the environmental benefits, an extensive life cycle assessment can be conducted based on existing conditions including waste management, food production and consumption compared to the design generated by the decision-making tool.

It should be noted that, after carefully reviewing the maps, a visualisation error was detected. Even though while assigning food production systems, the created connection seem correct, there is an error with the waste exchange points in the maps. This error seems to be only occurring in Stage 4. Therefore the connections visualised in stage 4 do not match with reality and waste flows do not reach where they are intended to be used. Due to time constraints the main source of the error could not be detected and the error could not be fixed. It should be noted that each software naturally has some errors and mistakes in the beginning and these errors can be fixed along the way with updated. Consequently, in the future, reviewing the decisions once again is recommended as well as testing Foodcycle on multiple cases to detect and possible mistakes.

08

REFLECTION

8.0 REFLECTION

Making a shift to a more resilient, less dependent urban metabolism can be achieved by introducing urban farming systems to urban development by utilising the waste flows as resources for the urban farm. A suitable urban farming system is highly dependent on the location, the situation demands, existing space, and existing waste flows. Creating such a network of waste flows and deciding according to existing conditions gets complicated for humans brains to handle especially when the datasets are big. In order to assist designers on this quest a decision making tool is needed. The objective of this thesis is to provide a decision-making strategy, therefore a decision-making tool for urban farming systems utilising various waste streams from the site with the main aim of creating a network of waste and resources while increasing the symbiosis in urban contexts

Graduation process

This research aims to bridge climate design & sustainability to design informatics by searching for a solution to a sustainable design problem by employing a computational design approach. In building technology track, we are trained to approach design problems methodologically and provide clear arguments for design decisions. This research relates to the aforementioned teachings within the program as it provides guidelines for a well built methodological approach rather than an intuition-based design approach.

As per the objective of this research, a decision making strategy is designed to address the problem of integrating urban farming solutions into urban contexts by seeing waste flows and vacant spaces as resources. The decision making strategy has both theoretical and realised parts due to time constraints within the scope of the graduation project. During the design process, a design by research approach is employed to design the decision-making tool. Literature review findings are used to formulate design rules. Then these design rules are applied to a case study in order to illustrate the results.

The decision-making tool is tested on the TU Delft Campus after collecting data regarding waste flows and vacant spaces on campus. The decision-making tool suggested varying urban farming systems for 86% of the vacant spaces and over half of the waste quantities of each waste type are used for farming purposes on the campus. In total 33 hectares of space are used for farming of various kinds including vermiculture, aquaculture, mushroom production, soft fruits and leafy vegetable production. As a result of intensive farming on campus enough vegetables for 88% of the Delft population can be produced only on TU Delft Campus by using rooftops and empty ground floor areas. As the objective of the research is to develop a methodological approach using waste flows for urban farming and create a network of waste and resource exchange in urban contexts, after analysing the results of incorporating such a strategy it can be concluded that the research succeeded within its scope and the given timeframe. It should be noted that some aspects of decision making are either simplified or suggested theoretically.

The main outcome of the project is the decision making strategy to assist designers in the quest of making decisions regarding urban farming systems and existing site conditions. Firstly, the technology for data collection methods that are theoretically suggested to be used for this strategy exists however they should be modified to fit this specific purpose. Secondly, for research purposes, some of the data used by the decision-making tool are simplified. Therefore, further improvements are necessary to achieve precise results. On the other hand, if simplified results are satisfactory then the decision-making tool can be used for preliminary design purposes. Or if the decision-making tool is developed further to include automated data collection by using drones, GIS and deep learning and to be used within a user interface while allowing for user interaction, the decision-making tool can be used in practice. In addition, the features the tool does not have currently, should be addressed. These features include using agricultural waste, fertiliser from mushroom farming. In the future, rainwater can be

suggested to be harvested from the roof of the assigned farms since currently only sloped roofs are taken into account for rainwater harvesting. Lastly, due to time constraints a data error in the stage 4 could not be fixed, this error sould be revisited and solved before use. Thankfully, it is observed that the error only hinders the visualisation of waste exchange in stage 4. Finaly, it should be noted that such errors may happen with any software or program therefore Foodcycle should be tested on more case studies to detect any possible mishap.

Ethical Issues & Societal impact

During the preliminary stages of this project, one of the most prominent driving forces of the research was making a societal impact by improving food security and the socio-economical state of nearby inhabitants. However, social impact is challenging to quantify and address with a building technology project as it falls outside of the scope of the master's program. In addition, due to business-related reasons, food produced on urban farms is not always more accessible than others economically even though food accessibility is one of the potential benefits of urban farms. Due to time constraints and knowledge limitations on the subject, social aspects are acknowledged as potential benefits of urban farms and disregarded as a primary outcome.

Both urban farming and decision-making strategy are topics of interest and they are widely discussed. This research delivers innovation by combining these popular subjects to achieve symbiosis in the built environment by seeing urban waste as a resource for urban farms. In addition, it aims to make a positive impact by finding innovative solutions to improve the well-being of the planet while benefiting society. One of the main problems this project tackles is the sustainable development of cities and urban farming in the cities by employing principles of considering waste as a resource. This way, consumption of valuable natural resources can be decreased for the sake of a healthy planet and sustainable development goals. Meanwhile, this research foresees social impact as food security issues can potentially be addressed by integrating farming into cities. Even though due to the project scope and timing, the social benefits of such an approach are perceived as a secondary benefits, the potential to improve food security was one of the driving forces starting this research. However, looking at the results of the TU Delft Campus case study, and recognising the quantity of food which could be grown on campus it can be concluded that by growing such a magnitude of food, food accessibility, security and availability can be improved. Moreover, this research aims to provide guidelines for designers who may not be experts in urban farming, different systems or waste, to make decisions to ensure sustainability and circularity of built environments. Therefore, it can potentially increase the commonality of utilising urban farming as an exchange hub of waste and food. Secondly, the decision-making tool can make suggestions based on the design rules which would be too complex for human cognition to tackle such as rules regarding waste quantities, and vacant space sizes. By using the tool to develop a plan for urban farming, the data regarding yields, and used waste quantities can be reached quickly In addition, the results of applying different rules can be compared theoretically.

Urban farming is a multifaceted topic that corresponds to technical aspects of agriculture as well as the built environment since it inherently aims to integrate food production into cities. Due to the complex nature of cities, urban farming should be also considered as an architectural solution rather than purely agricultural. Urban farming can contribute to and potentially improve the sustainability and biodiversity of cities if integrated and designed carefully which falls under the scope of architecture, building technology and design.

Mentor Feedback

During the mentor meetings, both my mentors Andy Jenkins and Michela Turrin gave constructive criticism to carry the project further. There were times aspects I failed to think about or simply missed were pointed out. By paying attention to those aspects, I believe that my graduation project is improved significantly thanks to my mentors. During the design of the decision-making tool since I was mostly working in Python language and the design rules are complex to grasp immediately, from time to time it was difficult to communicate the decision-making process. One of the prominent comments on my project was to explain the ideas clearly and to design an user interface in order to help the audience imagine how the tool can be used. In order to make the process easier to understand for anyone regardless of their background, I designed a representative user interface in addition to highlighting the logic behind each step and explaining them step by step. In this part of the project, how the decisions are made, and the rules are presented clearly.

Self Development

During the last 8 months, I gained experience and knowledge regarding different urban farming systems, waste sources, different ways of dealing with waste, designing methodologies and putting methodologies to test.

First of all, when I started my graduation project, I was not knowledgeable about different food production systems or how they work. In the beginning, it was partially clear to me that waste can be used as a resource for food production however I only learned how it can be used during the literature review. As a result of working on a decision-making methodology, I realised that especially with different parameters to consider, the rules behind decision making can get quite complex. It is possible to say that decision making is a puzzle with many dimensions. Solving that puzzle with computational approaches is a challenging yet viable option since an artificial mind can iterate over the same rules many times and it would give the same results as long as the rules are set. However, even though the rules should be set and all scenarios should be thought through these rules should not be too strict. Otherwise, system selection is only done in perfect conditions that do not reflect the reality of working with existing conditions. Lastly, in addition to technical and academic knowledge, I had the chance to put my coding skills, which I developed earlier this year, to the test. Since a large portion of the prototype is developed using Python, I learned how to divide tasks or rules into smaller components in a logical order. And to be completely honest, by doing so I eventually understood how to work with Grasshopper which I aspired to learn for many years. Therefore overall, I believe that this project taught me about both academic and practical aspects which will be useful in the near future.

09

BIBLIOGRAPHY

BIBLIOGRAPHY

10” 12 Foot Channel. (n.d.). Cropking Incorporated. Retrieved May 11, 2022, from https://cropking.com/catalog/cha9/10-12-foot-channel

Abbasi, S. A., Hussain, N., Tauseef, S. M., & Abbasi, T. (2018). A novel FLippable Units Vermireactor Train System FLUVTS for rapidly vermicomposting paper waste to an organic fertilizer. Journal of Cleaner Production, 198, 917–930. https://doi.org/10.1016/j.jclepro.2018.07.040

Abbasi, S. A., Nayeem-Shah, M., & Abbasi, T. (2015). Vermicomposting of phytomass: Limitations of the past approaches and the emerging directions. Journal of Cleaner Production, 93, 103–114. https://doi.org/10.1016/j.jclepro.2015.01.024

Ackerman, K. (2012). 7 - Urban agriculture: Opportunities and constraints. In F. Zeman (Ed.), Metropolitan Sustainability (pp. 118–146). Woodhead Publishing. https://doi.org/10.1533/9780857096463.2.118

Ahmad Zakil, F., Xuan, L. H., Zaman, N., Alan, N. I., Salahutheen, N. A. A., Sueb, M. S. M., & Isha, R. (2022). Growth performance and mineral analysis of Pleurotus ostreatus from various agricultural wastes mixed with rubber tree sawdust in Malaysia. Bioresource Technology Reports, 17, 100873. https://doi.org/10.1016/j.biteb.2021.100873

All About Growing Dwarf Tomatoes Guide and Q&A. (n.d.). Bountiful Gardener. Retrieved May 11, 2022, from https://www.bountifulgardener.com/growing-dwarf-tomatoes/

Bao, J., Lu, W.-H., Zhao, J., & Bi, X. T. (2018). Greenhouses for CO2 sequestration from atmosphere. Carbon Resources Conversion, 1(2), 183–190. https://doi.org/10.1016/j.crcon.2018.08.002

Baras, T. (2018, June 7). The Quality Standards for Hydroponic Lettuce. Urban Ag News. https://urbanagnews.com/blog/exclusives/the-quality-standards-for-hydroponic-lettuce/

Beirao, J. N., Nourian Ghadi Kolaee, P., & Mashhoodi, B. (2011). Parametric urban design: An interactive sketching system for shaping neighborhoods. ECAADe 2011: Proceedings of the 29th Conference on Education and Research in Computer Aided Architectural Design in Europe &quot;Respecting Fragile Places&quot;, Ljubljana, Slovenia, 21-24 September 2011. https://repository.tudelft.nl/islandora/object/uuid%3A23079c5f-d0ea-40e1-af1c-800c85b294c0

Biobean. (2019, August 28). The significant value of spent coffee grounds. Bio-Bean. https://www.bio-bean.com/news-post/the-significant-value-of-spent-coffee-grounds/

Blom, T. (2018). Delft in Transition: Towards a sustainable energy system for Dutch Municipalities. https://repository.tudelft.nl/islandora/object/uuid%3Aedefd4bb-ec6f-4e4e-947b-2161552cd9f9

Blom, T. J., Straver, W. A., Ingratta, F. J., Khosla, S., & Brown, W. (2012). Carbon Dioxide In Greenhouses. Ontario Ministry of Agriculture, Food and Rural Affairs. http://www.omafra.gov.on.ca/english/crops/facts/00-077.htm

Bundesinstitut Für Risikobewertung. (2020). Reclaimed waste water: Preventing bacterial pathogens on fresh fruit and vegetables: BfR Opinion No 021/2020 of 21 April 2020. https://doi.org/10.17590/20200624-073439

Caetano, I., Santos, L., & Leitão, A. (2020). Computational design in architecture: Defining parametric, generative, and algorithmic design. Frontiers of Architectural Research, 9(2), 287–300. https://doi.org/10.1016/j.foar.2019.12.008

Calculating the Amount of Rainwater Capturable from Your Roof. (2018, June 14). National Poly Industries. https://www.nationalpolyindustries.com.au/2018/06/14/calculating-the-amount-of-rainwater-capturable-from-your-roof/

Campisano, A., Butler, D., Ward, S., Burns, M. J., Friedler, E., DeBusk, K., Fisher-Jeffes, L. N., Ghisi, E., Rahman, A., Furumai, H., & Han, M. (2017). Urban rainwater harvesting systems: Research, implementation and future perspectives. Water Research, 115, 195–209. https://doi.org/10.1016/j.watres.2017.02.056

Chance, E., Ashton, W., Pereira, J., Mulrow, J. S., Norberto, J., Derrible, S., & Guilbert, S. (2021). The Plant—An experiment in urban food sustainability. https://doi.org/10.25417/uic.17159402.v1

Chatzikonstantinou, I. (2021). Architectural Design Performance Through Computational Intelligence: A Comprehensive Decision Support Framework. https://doi.org/10.4233/uuid:b0661996-af2d-4bd2-9127-48c678435f68

Chunkao, K., Nimpee, C., & Duangmal, K. (2012). The King’s initiatives using water hyacinth to remove heavy metals and plant nutrients from wastewater through Bueng Makkasan in Bangkok, Thailand. Ecological Engineering, 39, 40–52. https://doi.org/10.1016/j.ecoleng.2011.09.006

CO2 Meter. (2015). CO2 Calculator for Grow Room or Indoor Greenhouse. CO2 Meter. https://www.co2meter.com/blogs/news/41003521-co2-calculator-for-grow-room-or-indoor-greenhouse

Cubukcuoglu, C., Kirimtat, A., Ekici, B., Tasgetiren, F., & Suganthan, P. N. (2019). Evolutionary Computation for Theatre Hall Acoustics. In S. Datta & J. P. Davim (Eds.), Optimization in Industry: Present Practices and Future Scopes (pp. 55–83). Springer International Publishing. https://doi.org/10.1007/978-3-030-01641-8_4

Cycloponics. (n.d.). Cycloponics—We Farm the Underground. Cycloponics. Retrieved January 18, 2022, from http://cycloponics.co/

DC Health Matters. (n.d.). DC Health Matters: Demographics :: Ward :: Ward 7 :: Population. Retrieved May 11, 2022, from https://www.dchealthmatters.org/demographicdata?id=131494&sectionId=935

de Peyster, E. (2014). How much water does my food garden need? University of California Agriculture and Natural Resources. Retrieved May 11, 2022, from https://sonomamg.ucanr.edu/files/185639.pdf

Delft (Municipality, Zuid-Holland, Netherlands)—Population Statistics, Charts, Map and Location. (n.d.). Retrieved May 11, 2022, from https://www.citypopulation.de/en/netherlands/admin/zuid_holland/0503__delft/

Deng, Y. (2021). Pollution in rainwater harvesting: A challenge for sustainability and resilience of urban agriculture. Journal of Hazardous Materials Letters, 2, 100037. https://doi.org/10.1016/j.hazl.2021.100037

Density of Sawdust. (n.d.). Retrieved May 11, 2022, from https://www.aqua-calc.com/page/density-table/substance/sawdust

Dorr, E., Koegler, M., Gabrielle, B., & Aubry, C. (2021). Life cycle assessment of a circular, urban mushroom farm. Journal of Cleaner Production, 288, 125668. https://doi.org/10.1016/j.jclepro.2020.125668

Duarte, J. P. (2005). A discursive grammar for customizing mass housing: The case of Siza's houses at Malagueira. Automation in Construction, 14(2), 265–275. https://doi.org/10.1016/j.autcon.2004.07.013

Edwards, C. A., & Dominguez, J. (2010). Biology and ecology of earthworm species used for vermicomposting. Vermiculture Technology: Earthworms, Organic Waste and Environmental Management, 25–37.

Engineering ToolBox, (2004). Carbon Dioxide Emission from People vs. Activity. [online] Available at: https://www.engineeringtoolbox.com/co2-persons-d_691.html

Engineering Toolbox. (n.d.). Carbon Dioxide Emission from People vs. Activity. Retrieved May 11, 2022, from https://www.engineeringtoolbox.com/co2-persons-d_691.html

European Commission. (2020). 2050 long-term strategy. https://ec.europa.eu/clima/eu-action/climate-strategies-targets/2050-long-term-strategy_en

Fong, J., & Hewitt, P. (n.d.). Six Easy Steps to Setting Up a Worm Bin—Cornell Composting. Cornell Composting. Retrieved May 11, 2022, from http://compost.css.cornell.edu/worms/steps.html

Fresco, L. O. (2009). Challenges for food system adaptation today and tomorrow. Environmental Science & Policy, 12(4), 378–385. https://doi.org/10.1016/j.envsci.2008.11.001

Fresh Tomato Weights (Ultimate Guide With Charts & Calculator). (2021, September 16). Weigh School. https://weighschool.com/tomato-weights/

from https://www.smallhold.com/about

Fun facts about compost worms. (n.d.). Compost Community. Retrieved May 11, 2022, from https://www.compostcommunity.com.au/fun-facts.html

Fun facts about compost worms. (n.d.). Compost Community. Retrieved May 11, 2022, from https://www.compostcommunity.com.au/fun-facts.html

Furlong, C., Rajapaksha, N. S., Butt, K. R., & Gibson, W. T. (2017). Is composting worm availability the main barrier to large-scale adoption of worm-based organic waste processing technologies? Journal of Cleaner Production, 164, 1026–1033. https://doi.org/10.1016/j.jclepro.2017.06.226

Gallego-Alarcón, I., Fonseca, C. R., García-Pulido, D., & Díaz-Delgado, C. (2019). Proposal and assessment of an aquaculture recirculation system for trout fed with harvested rainwater. Aquacultural Engineering, 87, 102021. https://doi.org/10.1016/j.aquaeng.2019.102021

Garg, P., Gupta, A., & Satya, S. (2006). Vermicomposting of different types of waste using Eisenia foetida: A comparative study. Bioresource Technology, 97(3), 391–395. https://doi.org/10.1016/j.biortech.2005.03.009

Geodan. (n.d.). Klimaateffectatlas. Retrieved May 11, 2022, from https://www.klimaateffectatlas.nl/en/

Geurts, M., van Bakel, A. M., van Rossum, M., de Boer, E., & Ocke, M. C. (2016). Food consumption in the Netherlands and its determinants, Background report to 'What's on our plate? Safe, healthy and sustainable diets in the Netherlands. (p. 72). RIVM. https://www.rivm.nl/bibliotheek/rapporten/2016-0195.pdf

Gillette, B. (n.d.). How to Space Tomato Plants. The Spruce. Retrieved May 11, 2022, from https://www.thespruce.com/how-to-space-tomato-plants-5219656

Gondhalekar, D., & Ramsauer, T. (2017). Nexus City: Operationalizing the urban Water-Energy-Food Nexus for climate change adaptation in Munich, Germany. Urban Climate, 19, 28–40. https://doi.org/10.1016/j.uclim.2016.11.004

Graamans, L. (2021). STACKED: The building design, systems engineering and performance analysis of plant factories for urban food production. A+BE | Architecture and the Built Environment, 05, 1–266. https://doi.org/10.7480/abe.2021.05.5666

Graamans, L., Baeza, E., van den Dobbelsteen, A., Tsafaras, I., & Stanghellini, C. (2018). Plant factories versus greenhouses: Comparison of resource use efficiency. Agricultural Systems, 160, 31–43. https://doi.org/10.1016/j.agsy.2017.11.003

GroCycle. (2021). Free Workshop: Starter Guide To Mushroom Farming | GroCycle—YouTube. https://www.youtube.com/watch?v=icKeO-kyiGk&feature=emb_imp_woyt

Gu, N., & Behbahani, P. A. (2018). Shape Grammars: A Key Generative Design Algorithm. In B. Sriraman (Ed.), Handbook of the Mathematics of the Arts and Sciences (pp. 1–21). Springer International Publishing. https://doi.org/10.1007/978-3-319-70658-0_7-1

Hamilton, D. W. (2017, February 1). The Basics of Vermicomposting—Oklahoma State University. Oklahoma State University. https://extension.okstate.edu/fact-sheets/the-basics-of-vermicomposting.html

Herr, C. M., & Ford, R. C. (2016). Cellular automata in architectural design: From generic systems to specific design tools. Automation in Construction, 72, 39–45. https://doi.org/10.1016/j.autcon.2016.07.005

Hirneisen, K., Sharma, M., & Kniel, K. (2012). Human Enteric Pathogen Internalization by Root Uptake into Food Crops. Foodborne Pathogens and Disease, 9, 396–405. https://doi.org/10.1089/fpd.2011.1044

Ho, C.-O., Nie, T., Su, L., Yang, Z., Schwegler, B., & Calvez, P. (2021). Graph-based algorithmic design and decision-making framework for district heating and cooling plant positioning and network planning. Advanced Engineering Informatics, 50, 101420. https://doi.org/10.1016/j.aei.2021.101420

Hofman, J. A. M. H., & Paalman, M. (2014). Rainwater harvesting, a sustainable solution for urban climate adaptation? (Wageningen University & Research - Library). KWR Watercycle Research Institute. https://edepot.wur.nl/345625

Hofman-Caris, R., Bertelkamp, C., de Waal, L., van de Brand, T., van der AA, R., & van der Hoek, P. (2018). Rainwater harvesting in the Netherlands: Useful or not? Water Solutions, 3, 9.

https://pure.qub.ac.uk/en/studentTheses/building-integrated-technical-food-systems

Jenkins, A. (2018). Building Integrated Technical Food Systems.

Jenkins, A., & Keeffe, G. (2017). The Integration of Urban Agriculture and the Socio-economic Landscape of Future Cities. https://www.researchgate.net/publication/325763933_The_Integration_of_Urban_Agriculture_and_the_Socio-economic_Landscape_of_Future_Cities

Jenkins, A., & Keeffe, G. (2019). Reduction in building energy use as a result of food production within a double-skinned glazed facade. https://www.researchgate.net/publication/337236203_Reduction_in_building_energy_use_as_a_result_of_food_production_within_a_double-skinned_glazed_facade

LeBoeuf, J. (2013). Tomato plant spacing research. Ontario Ministry of Agriculture, Food and Rural Affairs. http://www.omafra.gov.on.ca/english/crops/hort/news/hortmatt/2013/12hrt13a4.htm

Lehmann, S. (2011). Transforming the City for Sustainability: The Principles of Green Urbanism. Journal of Green Building, 6(1), 104–113. https://doi.org/10.3992/jgb.6.1.104

Leni, G., Caligiani, A., & Sforza, S. (2021). Chapter 40 - Bioconversion of agri-food waste and by-products through insects: A new valorization opportunity. In R. Bhat (Ed.), Valorization of Agri-Food Wastes and By-Products (pp. 809–828). Academic Press. https://doi.org/10.1016/B978-0-12-824044-1.00013-1

Li, D., Liu, H., Qiao, Y., Wang, Y., Cai, Z., Dong, B., Shi, C., Liu, Y., Li, X., & Liu, M. (2013). Effects of elevated CO2 on the growth, seed yield, and water use efficiency of soybean (Glycine max (L.) Merr.) under drought stress. Agricultural Water Management, 129, 105–112. https://doi.org/10.1016/j.agwat.2013.07.014

Lian, Y., Oyama, A., & Liou, M.-S. (2010). Progress in design optimization using evolutionary algorithms for aerodynamic problems. Progress in Aerospace Sciences, 46(5), 199–223. https://doi.org/10.1016/j.paerosci.2009.08.003

Lundy, L., Revitt, M., & Ellis, B. (2018). An impact assessment for urban stormwater use. Environmental Science and Pollution Research, 25(20), 19259–19270. https://doi.org/10.1007/s11356-017-0547-4

Macias-Corral, M. A., & Sanchez-Cohen, I. (2019). Rainwater harvesting for multiple uses: A farm-scale case study. International Journal of Environmental Science and Technology, 16(10), 5955–5964. https://doi.org/10.1007/s13762-018-02200-7

Magdy, H., & Eldaly, H. (2020). Applying Swarm Intelligence in Architectural Design. In S. Kamel, H. Sabry, G. F. Hassan, M. Refat, A. Elshater, A. S. A. Elrahman, D. K. Hassan, & R. Rashed (Eds.), Architecture and Urbanism: A Smart Outlook (pp. 77–87). Springer International Publishing. https://doi.org/10.1007/978-3-030-52584-2_6

Martínez-Carrera, D., Aguilar, A., Martínez, W., Bonilla, M., Morales, P., & Sobal, M. (2000). MUSHROOM CULTIVATION ON COFFEE PULP Commercial production and marketing of edible mushrooms cultivated on coffee pulp in Mexico. https://doi.org/10.1007/978-94-017-1068-8_45

Mcintosh, J. (2021). How to Grow Butterhead (Buttercrunch) Lettuce. The Spruce. https://www.thespruce.com/how-to-grow-and-care-for-buttercrunch-lettuce-4767592

Ministerie van Economische Zaken, L. en I. (2009). Policy Agenda for Sustainable Food Systems—Parliamentary document—Government.nl Ministerie van Algemene Zaken. https://www.government.nl/documents/parliamentary-documents/2009/11/17/policy-agenda-for-sustainable-food-systems

Ministry of Economic Affairs. (2017). Energy Agenda: Towards a low-carbon energy supply - Report - Government.nl [Rapport]. Ministerie van Algemene Zaken. https://www.government.nl/documents/reports/2017/03/01/energy-agenda-towards-a-low-carbon-energy-supply

Mirón-Mérida, V. A., Barragán-Huerta, B. E., & Gutiérrez-Macías, P. (2021). Chapter 9 - Coffee waste: A source of valuable technologies for sustainable development. In R. Bhat (Ed.), Valorization of Agri-Food Wastes and By-Products (pp. 173–198). Academic Press. https://doi.org/10.1016/B978-0-12-824044-1.00009-X

Muneer, A. M., & Narula, S. A. (2021). Chapter 30—Sustainability of agri-food supply chains through innovative waste management models. In R. Bhat (Ed.), Valorization of Agri-Food Wastes and By-Products (pp. 591–605). Academic Press. https://doi.org/10.1016/B978-0-12-824044-1.00039-8

Nanonets. (2018). Use Machine Learning APIs with Drones. https://nanonets.com/drone/?utm_source=Medium.com&utm_campaign=Object%20Detection%20on%20Aerial%20Imagery%20using%20Drones%20with%20Deep%C2%A0Learning

National Institute for Public Health and the Environment (RIVM). (n.d.). Food consumption in the Netherlands and its determinants, Background report to 'What's on our plate? Safe, healthy and sustainable diets in the Netherlands. 72.

Nayak, A., & Bhushan, B. (2021). Chapter 10 - Valorization of coffee wastes for effective recovery of value-added bio-based products: An aim to enhance the sustainability and productivity of the coffee industry. In R. Bhat (Ed.), Valorization of Agri-Food Wastes and By-Products (pp. 199–218). Academic Press. https://doi.org/10.1016/B978-0-12-824044-1.00040-4

Nelson, M., & Wolverton, B. C. (2011). Plants+soil/wetland microbes: Food crop systems that also clean air and water. Advances in Space Research, 47(4), 582–590. https://doi.org/10.1016/j.asr.2010.10.007

Nourian, P. (2017). Configraphics Graph Theoretical Methods for DesignandAnalysis of Spatial Configurations. In BK BOOKS. https://books.bk.tudelft.nl/press/catalog/book/isbn.9789461867209

NSW Government. (2021). Ventilation in greenhouses. Department of Primary Industries. https://www.dpi.nsw.gov.au/agriculture/horticulture/greenhouse/structures-and-technology/ventilation

Ortells, S. (2015). Health Impact Assessment of New Urban Water Concepts. https://repository.tudelft.nl/islandora/object/uuid%3A0e41d07b-9f44-4220-aaac-e22c73c5074a

Persily, A., & Polidoro, B. J. (2020, January 28). Residential Application of an Indoor Carbon Dioxide Metric. AIVC. https://www.aivc.org/resource/residential-application-indoor-carbon-dioxide-metric

Pharino, C. (2021). Chapter 31 - Food waste generation and management: Household sector. In R. Bhat (Ed.), Valorization of Agri-Food Wastes and By-Products (pp. 607–618). Academic Press. https://doi.org/10.1016/B978-0-12-824044-1.00045-3

Prill, R. (2000). Why Measure Carbon Dioxide Inside Buildings? Washington State University Extension Energy Program. https://www.energy.wsu.edu/documents/co2inbuildings.pdf

Rassmann, L. (2015, September 16). The Future of Farming Plants Roots in Newark. NJTV News Archive. https://www.njspotlightnews.org/news/uncategorized/the-future-of-farming-plants-roots-in-newark/

Roggema, R. (2009). Adaptation to Climate Change: A Spatial Challenge. https://doi.org/10.1007/978-1-4020-9359-3

Roggema, R. E., Keeffe, G., & Hogeschool Van Hall Larenstein (Eds.). (2014). Why we need small cows : ways to design for urban agriculture. VHL University of Applied Sciences.

Sanyé-Mengual, E. (2015). Sustainability assessment of urban rooftop farming using an interdisciplinary approach. https://doi.org/10.13140/RG.2.1.1346.6089

Sanyé-Mengual, E., Llorach-Massana, P., Sanjuan-Delmás, D., Oliver-Solà, J., Josa, A., Montero, J., & Rieradevall, J. (2014). The ICTA-ICP Rooftop Greenhouse Lab (RTG-Lab): Closing metabolic flows (energy, water, CO 2) through integrated Rooftop Greenhouses (pp. 693–701). https://doi.org/10.13140/RG.2.1.5016.7206

Sayner, A. (2020). A Complete Guide to Mushroom Substrates. GroCycle. https://grocycle.com/mushroom-substrate/

Sayner, A. (2021, May 31). How to Grow Mushrooms in a Bag. GroCycle. https://grocycle.com/grow-mushrooms-in-a-bag/

Schets, F. M., Italiaander, R., van den Berg, H. H. J. L., & de Roda Husman, A. M. (2009). Rainwater harvesting: Quality assessment and utilization in The Netherlands. Journal of Water and Health, 8(2), 224–235. https://doi.org/10.2166/wh.2009.037

Sharma, K., & Garg, V. K. (2019). Chapter 10 - Vermicomposting of Waste: A Zero-Waste Approach for Waste Management. In M. J. Taherzadeh, K. Bolton, J. Wong, & A. Pandey (Eds.), Sustainable Resource Recovery and Zero Waste Approaches (pp. 133–164). Elsevier. https://doi.org/10.1016/B978-0-444-64200-4.00010-4

Shields, T. (2017, February 16). Mushroom Grow Bags: The Ultimate Guide. FreshCap Mushrooms. https://learn.freshcap.com/growing/mushroom-grow-bags-the-ultimate-guide/

Smallhold. (n.d.). Smallhold. Retrieved January 18, 2022,

Statistiek, C. B. voor de. (n.d.). Bevolking; geslacht, leeftijd, nationaliteit en regio, 1 januari [Webpagina]. Centraal Bureau voor de Statistiek. Retrieved May 11, 2022, from https://www.cbs.nl/nl-nl/cijfers/detail/84727NED?q=delft%20population

StatLine—Municipal waste; quantities. (n.d.). Retrieved May 11, 2022, from https://opendata.cbs.nl/statline/#/CBS/en/dataset/83558ENG/table?ts=1648633605354

Stouffs, R., & Krishnamurti, R. (2001). Sortal grammars: A framework for exploring grammar formalisms. In Quality, Reliability, and Maintenance (ed. G.J. McNulty), pp. 367-370, Professional Engineering Publishing, Bury St. Edmunds, UK, 2000.

Sustainability at TU Delft. (n.d.). TU Delft. Retrieved May 11, 2022, from https://www.tudelft.nl/sustainability

Teleszewski, T., & Gładyszewska-Fiedoruk, K. (2019). The concentration of carbon dioxide in conference rooms: A simplified model and experimental verification. International Journal of Environmental Science and Technology, 16(12), 8031–8040. https://doi.org/10.1007/s13762-019-02412-5

ten Caat, N., Graamans, L., Tenpierik, M., & van den Dobbelsteen, A. (2021). Towards Fossil Free Cities—A Supermarket, Greenhouse & Dwelling Integrated Energy System as an Alternative to District Heating: Amsterdam Case Study. Energies, 14(2), 347. https://doi.org/10.3390/en14020347

ten Caat, N., Tillie, N., & Tenpierik, M. (2021). Pig Farming vs. Solar Farming: Exploring Novel Opportunities for the Energy Transition. In R. Roggema (Ed.), TransFEWmation: Towards Design-led Food-Energy-Water Systems for Future Urbanization (pp. 253–280). Springer International Publishing. https://doi.org/10.1007/978-3-030-61977-0_12

The Effects of Too Much CO2 In a Grow Room. (2021, July 16). Atlas Scientific. https://atlas-scientific.com/blog/effects-of-too-much-co2-in-grow-room/

Tocchetto, D., Rubenstein, M., Nelson, M., & Al-Asadi, J. (2022). Chapter 11 - Circular economy in the Mesopotamian Marshes: The Eden in Iraq Wastewater Garden Project. In A. Stefanakis & I. Nikolaou (Eds.), Circular Economy and Sustainability (pp. 181–198). Elsevier. https://doi.org/10.1016/B978-0-12-821664-4.00006-6

Topology Optimization. (n.d.). TU Delft. Retrieved January 22, 2022, from https://www.tudelft.nl/3me/over/afdelingen/precision-and-microsystems-engineering-pme/research/structural-optimization-and-mechanics-som/som-research/topology-optimization

Tsui, T., Peck, D., Geldermans, B., & van Timmeren, A. (2021). The Role of Urban Manufacturing for a Circular Economy in Cities. Sustainability, 13(1), 23. https://doi.org/10.3390/su13010023

TU Delft. (n.d.-a). Education Space Viewer. Retrieved May 11, 2022, from https://esviewer.tudelft.nl/

TU Delft. (n.d.-b). Statistieken TU Delft 2015. Retrieved May 11, 2022, from https://issuu.com/tudelft-mediasolutions/docs/statistieken-tudelft-2015

United Nations Environment Programme (2021). Global Chemicals and Waste Indicator Review Document. Nairobi. https://wedocs.unep.org/bitstream/handle/20.500.11822/36753/GCWIR.pdf?sequence=3&isAllowed=y

United Nations Environment Programme (2021, May 12). Indicator 12.3.1(b). UNEP - UN Environment Programme. http://www.unep.org/explore-topics/sustainable-development-goals/why-do-sustainable-development-goals-matter/goal-12-3

United States Environmental Protection Agency. (2002). Wastewater Technology Fact Sheet: The Living Machine. 7. https://www3.epa.gov/npdes/pubs/living_machine.pdf

Urban Green Blue Grids. (n.d.). Hoogeland, Naaldwijk, The Netherlands | Urban green-blue grids. Retrieved January 18, 2022, from https://www.urbangreenbluegrids.com/projects/hoogeland-naaldwijk-the-netherlands/

Van den Dobbelsteen, A., Martin, C. L., Keeffe, G., Pulselli, R. M., & Vandevyvere, H. (2018). From Problems to Potentials—The Urban Energy Transition of Gruž, Dubrovnik. Energies, 11(4), 922. https://doi.org/10.3390/en11040922

Vertical Grow Racks For Indoor Farming. (n.d.). Pipp Horticulture. Retrieved May 11, 2022, from https://pipphorticulture.com/

Viscon Group. (2021, February 24). Cultivation of lettuce in end-packaging. Viscon Group. https://viscongroup.eu/news/cultivation-of-lettuce-in-end-packaging/

Wang, X., Song, Y., & Tang, P. (2020). Generative urban design using shape grammar and block morphological analysis. Frontiers of Architectural Research, 9(4), 914–924. https://doi.org/10.1016/j.foar.2020.09.001

Wilson, R. J. (1996). Introduction to graph theory . Prentice Hall. https://www.maths.ed.ac.uk/~v1ranick/papers/wilsongraph.pdf

Woven Planet Holdings. (2021). Automated Mapping Platform. Woven Planet. https://woven-planet.global/

Wu, Y., Zhang, N., Wang, J., & Sun, Z. (2012). An integrated crop-vermiculture system for treating organic waste on fields. European Journal of Soil Biology, 51, 8–14. https://doi.org/10.1016/j.ejsobi.2012.03.005

Xin, M., Shuang, L., Yue, L., & Qinzhu, G. (2015). Effectiveness of gaseous CO2 fertilizer application in China's greenhouses between 1982 and 2010. Journal of CO2 Utilization, 11, 63–66. https://doi.org/10.1016/j.jcou.2015.01.005

Xu, L., Yang, S., Zhang, Y., Jin, Z., Huang, X., Bei, K., Zhao, M., Kong, H., & Zheng, X. (2020). A hydroponic green roof system for rainwater collection and greywater treatment. Journal of Cleaner Production, 261, 121132. https://doi.org/10.1016/j.jclepro.2020.121132

10

APPENDIX

APPENDIX A
URBAN FARMING SYSSTEMS

Appendix A | Urban Farming Systems

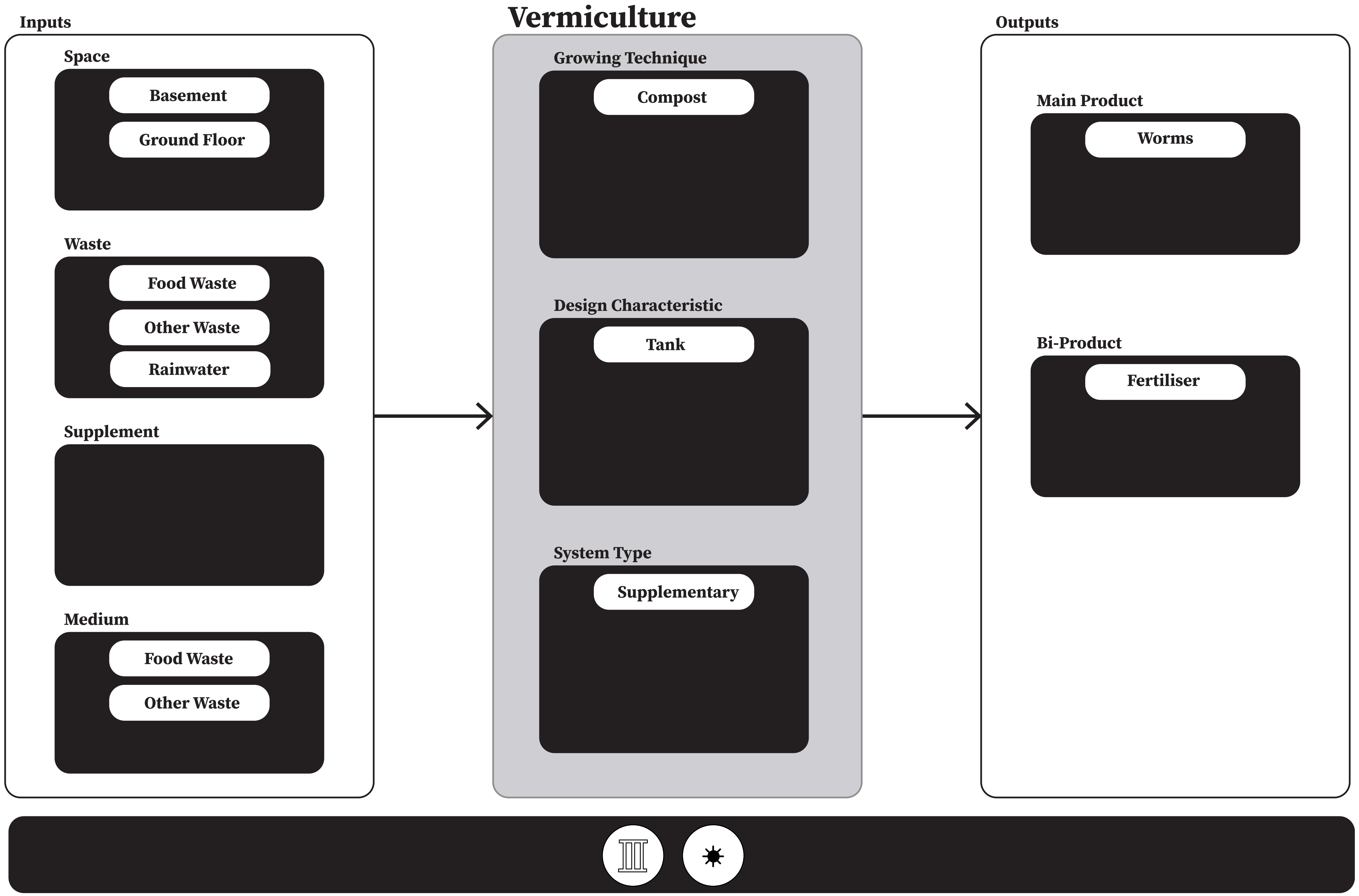
A.1 Table of Systems (to be updated)

Space	Waste	Supplement	Medium	Growing Technique	Design Characteristic	System Type	Main Product	Bi-Product
Rooftop ●●	Food Waste ●	Fertilser ●	Soil ●	Compost ●	Fish Tank ●	Food Production ●●●●●●●	Small Crops ●●●●●	Heat ●●●●●
Facade ●●	Coffee Waste ●	Nutrient Solution ●●●●●	Water ●●●●	Spawning ●	Tank ●	Supplementary ●	Large Crops ●●	Food Waste ●●●●●●
Intermediate Floor ●●●●	Other Waste ●●	Calcium ●	Fish Tank Water ●●●	Aquaculture ●	Stacked System ●	Food Producing Supplementary ●	Mushrooms ●	Spent Mushroom Substrate ●
Ground Floor ●●●●●●●●	CO2 ●●●●●●	Lime Bath* ●	Air ●	Raised Beds ●	Horizontal ●●●●●●		Worms ●	Fertiliser ●
Basement ●●●●●●●●	Rainwater ●●●●●●●		Food Waste ●	NFT ●●	Vertical ●●●●		Fish ●	Fish Tank Water ●
	Heat ●		Coffee Waste ●	Aeroponics ●	Modular Frame ●●●			
			Other Waste ●●	EBB & Flow ●				
			Clay Balls ●	Gravity Trickle ●				
				Water Culture ●●				

* Lime Bath is used for pasteurization of substrate.

● Vermiculture	● Mushroom	● Plant Factory
● Aquaculture	● Raised Bed	● Aeroponics
● Hydroponic - NFT	● Hydroponic - Water Culture	● Hydroponic - Media Bed

A.2 Vermiculture

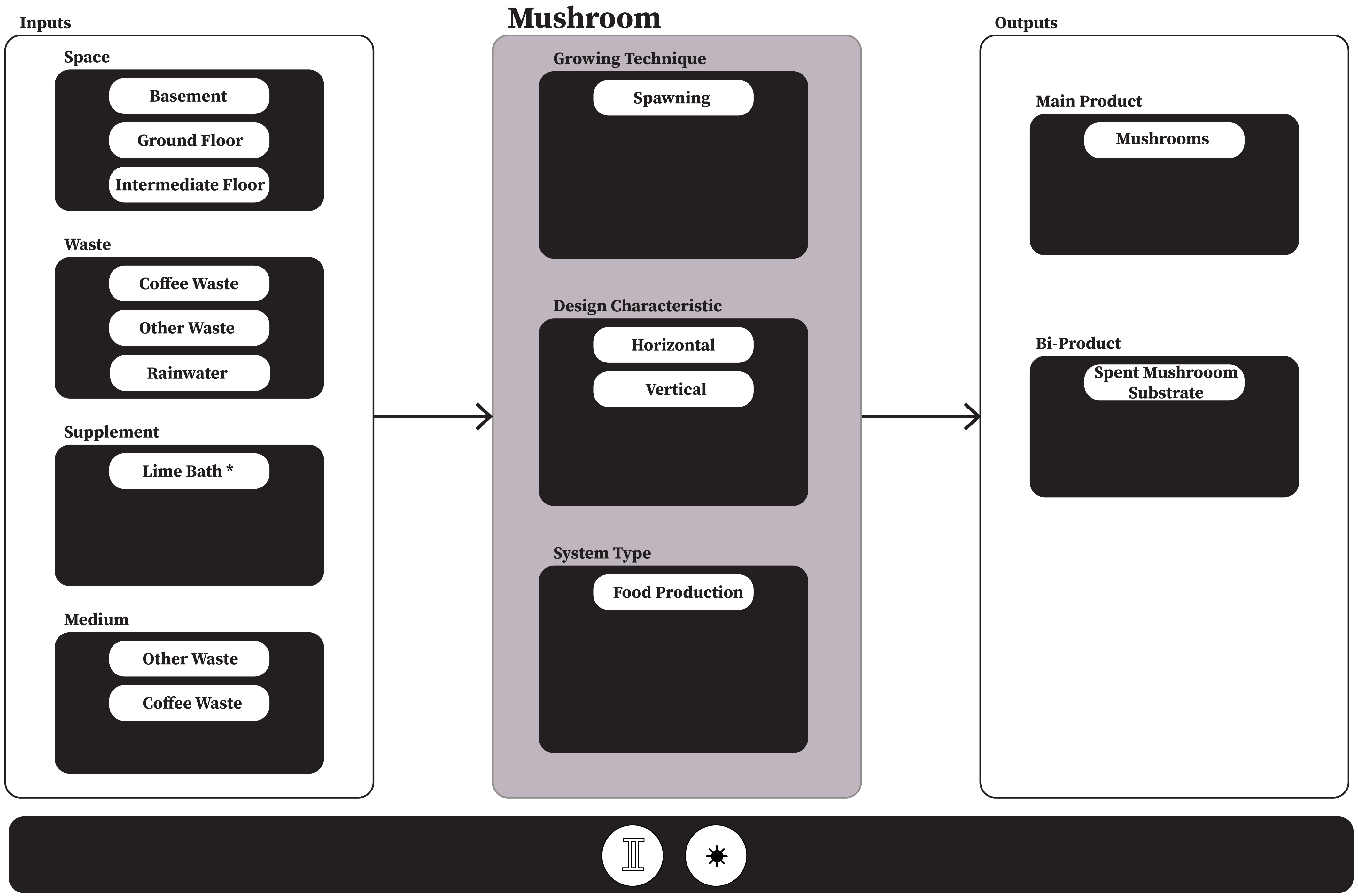


Notes:

- Carbon/Nitrogen ratio of 30:1 is favourable and mixing bulking substrate with waste is a viable approach (Sharma % Grag, 2019).
- Moisture should be between 50-90% while the temperatures vary depending on the earthworms species (Dominguez & Edwards, 2010).
- Temperature should range between 15-25 C for growth, below 10 C feeding slows down and under 4 C cocooning and growth stops (Dominguez & Edwards, 2010).

- Prior to vermicomposting, waste should be mixed with bulking substrate to balance carbon content and to overcome negative effects of toxic substances in waste (Sharma & Garg, 2019).
- Bulking substrate can be sourced from vermicomposting, composting, dry leaves or agricultural waste (Sharma & Garg, 2019).
- After mixing the bulking substrate and waste, the mixture should be precomposted to remove toxic gasses to earthworms, and to eliminate anaerobic conditions as explained by the same authors (Sharma & Garg, 2019).

A.3 Mushroom Production



Inputs

Space

Basement

Ground Floor

Intermediate Floor

Waste

Coffee Waste

Other Waste

Rainwater

Supplement

Lime Bath *

Medium

Other Waste

Coffee Waste

Mushroom

Growing Technique

Spawning

Design Characteristic

Horizontal

Vertical

System Type

Food Production

Outputs

Main Product

Mushrooms

Bi-Product

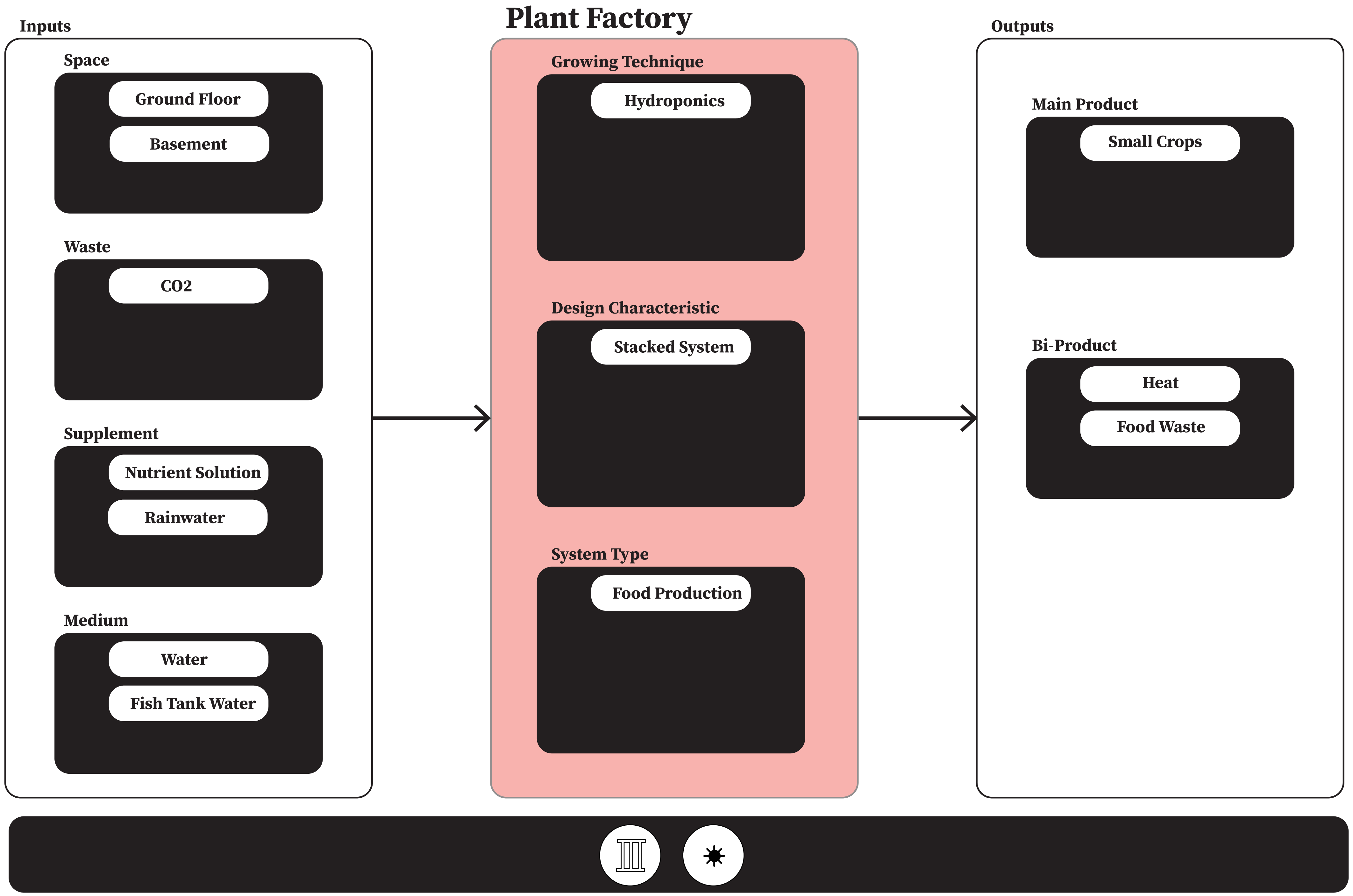
Spent Mushrooom Substrate

Notes:

- Preparing the substrate includes mixing the substrate and sterilising it (Dorr et al., 2021; GroCycle, 2021).
- The mixture is placed into plastic bags (Dorr et al.) or in reusable buckets (GroCycle, 2021) for incubation.
- Incubation takes place after the substrate mixture and mushroom spawned are mixed in a cultivation container (GroCycle, 2021).
- Incubation should be done in a dark environment since light triggers mushrooms to fruit before it is required as explained in the same source.
- The spawn-substrate mixture should incubate for 2 weeks at 70% relative humidity while room temperature is 17C(Dorr et al., 2021).

- Afterwards it should spend 7 weeks at 93% relative humidity and 16.5C room for fruiting (Dorr et al., 2021).
- Temperature requirements for fruiting depend on the mushroom variety and ranges between 10-32 C (GroCycle, 2021).
- One way of fruiting mushrooms is cutting holes in the bag or bucket and spraying them daily for high humidity levels as described in the same source (GroCycle, 2021).
- Harvesting takes place in the previously mentioned 7 weeks where mushrooms are harvested several times before the substrate loses its nutrient content (Dorr et al., 2021).

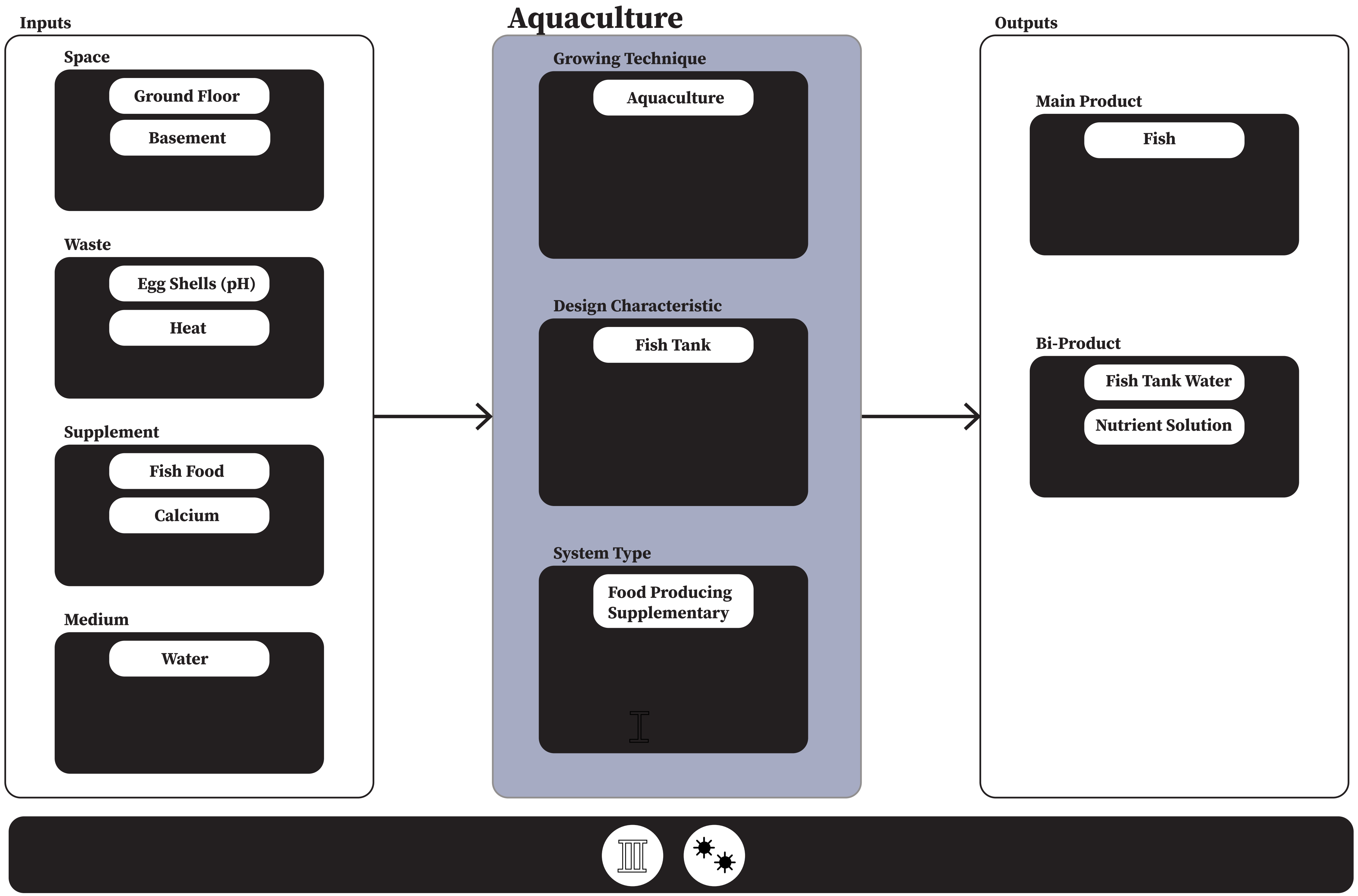
A.4 Plant Factory (to be updated)



Notes:

- The nutrient solution should be changed periodically, close attention should be given to keep the system stable and if some pathogens enter the systems all the crops should be discarded (Jenkins, 2018).
- Crops smaller than 300 mm are preferable in such settings to maximise vertical production (Jenkins, 2018).

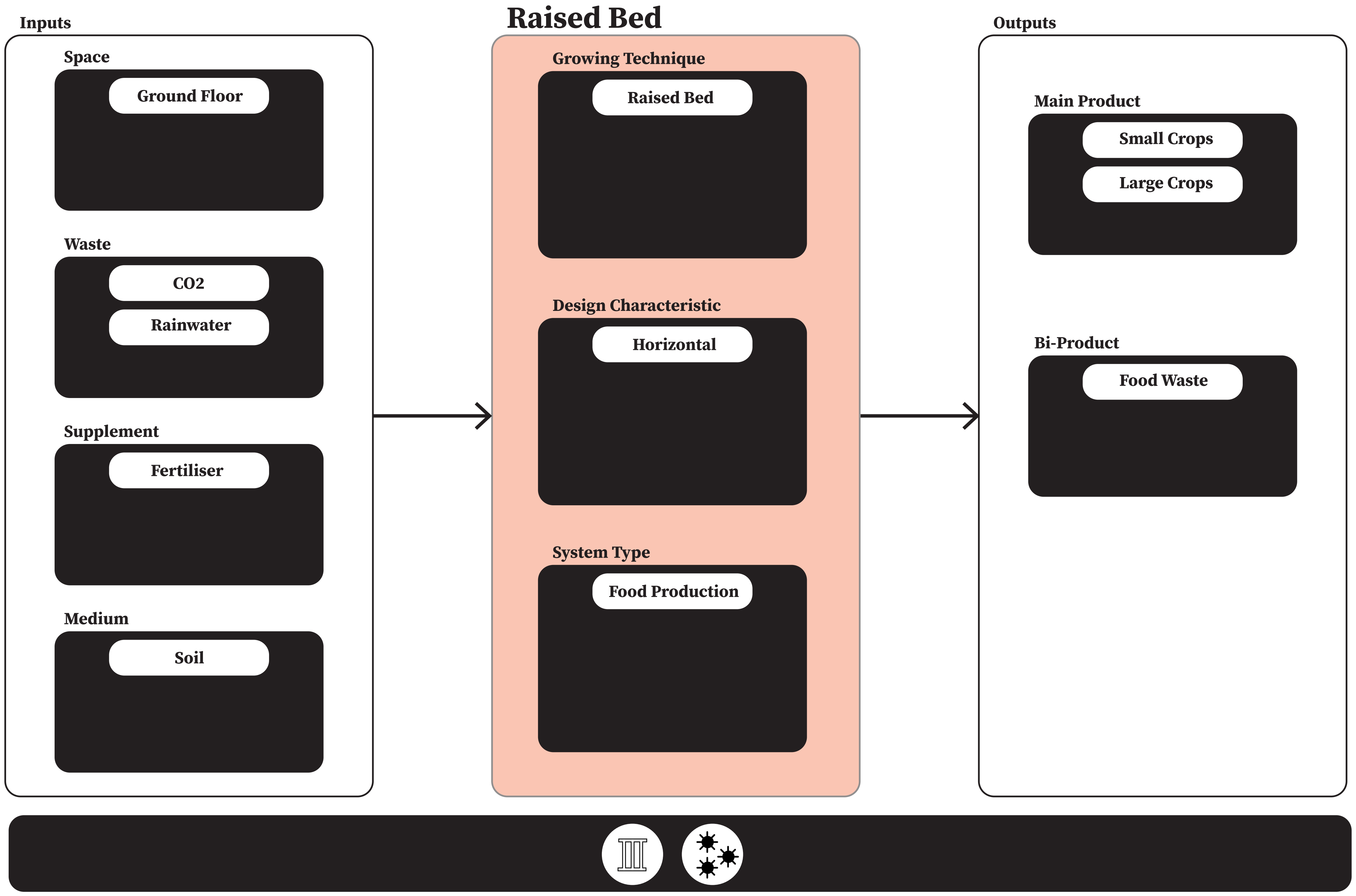
A.5 Aquaculture



Notes:

- Ammonia is essentially toxic to fish if it accumulates within the tank however it is first converted to nitrite then to nitrate by nitrobacter and nitrosomonas bacterias within the filtration system (Jenkins, 2018).
- The filtration medium should be a highly porous material such as expanded clay pellets or balls (Jenkins, 2018).

A.6 Raised Beds (to be updated)



Inputs

Space

Ground Floor

Waste

CO2

Rainwater

Supplement

Fertiliser

Medium

Soil

Raised Bed

Growing Technique

Raised Bed

Design Characteristic

Horizontal

System Type

Food Production

Outputs

Main Product

Small Crops

Large Crops

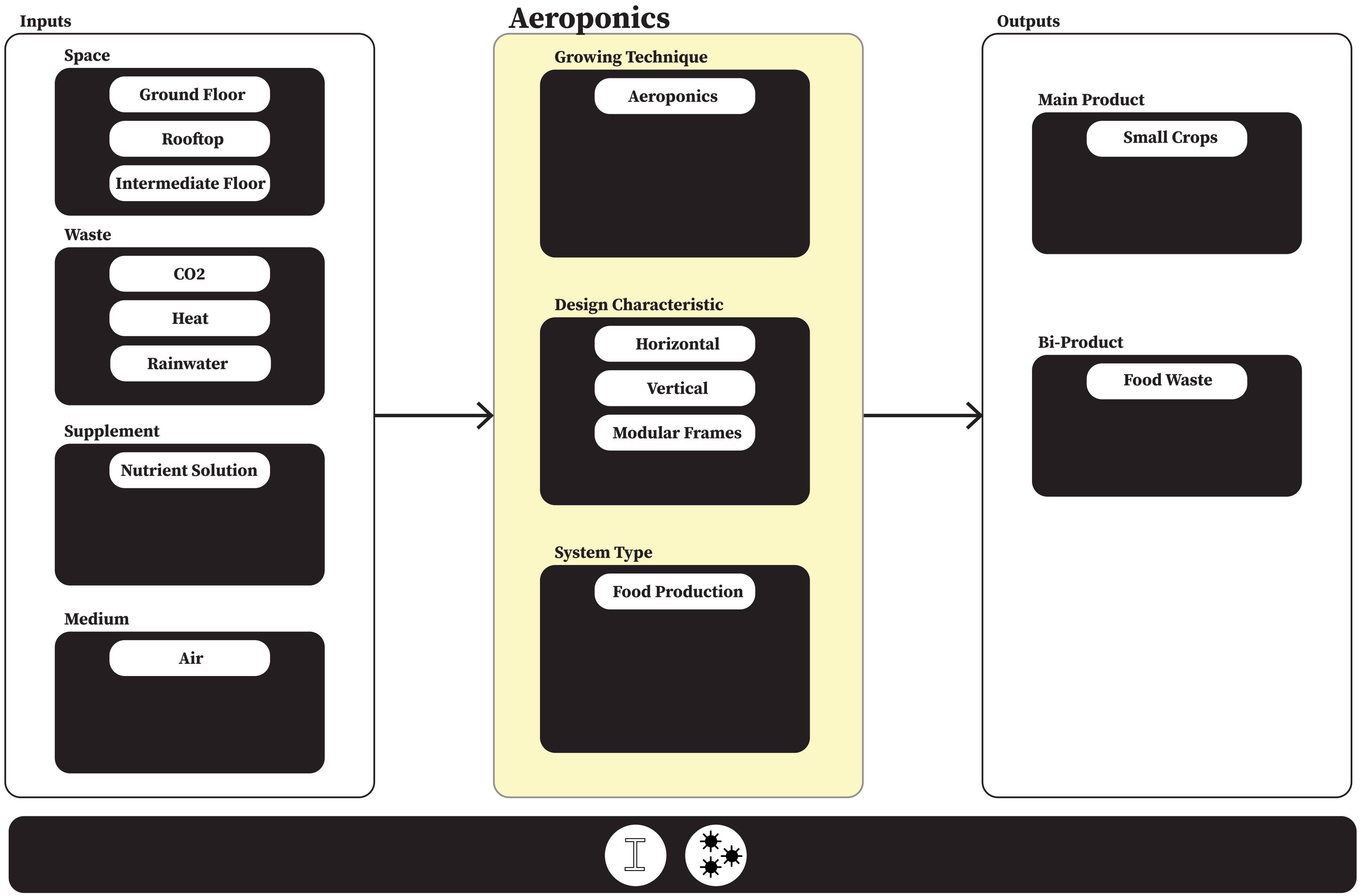
Bi-Product

Food Waste

Notes:

- They can be designed in many shapes and sizes
- Raised beds seperate the growing medium from contaminated soil by using impervious membranes (Jenkins, 2018)
- They offer elevated food production (Jenkins, 2018).

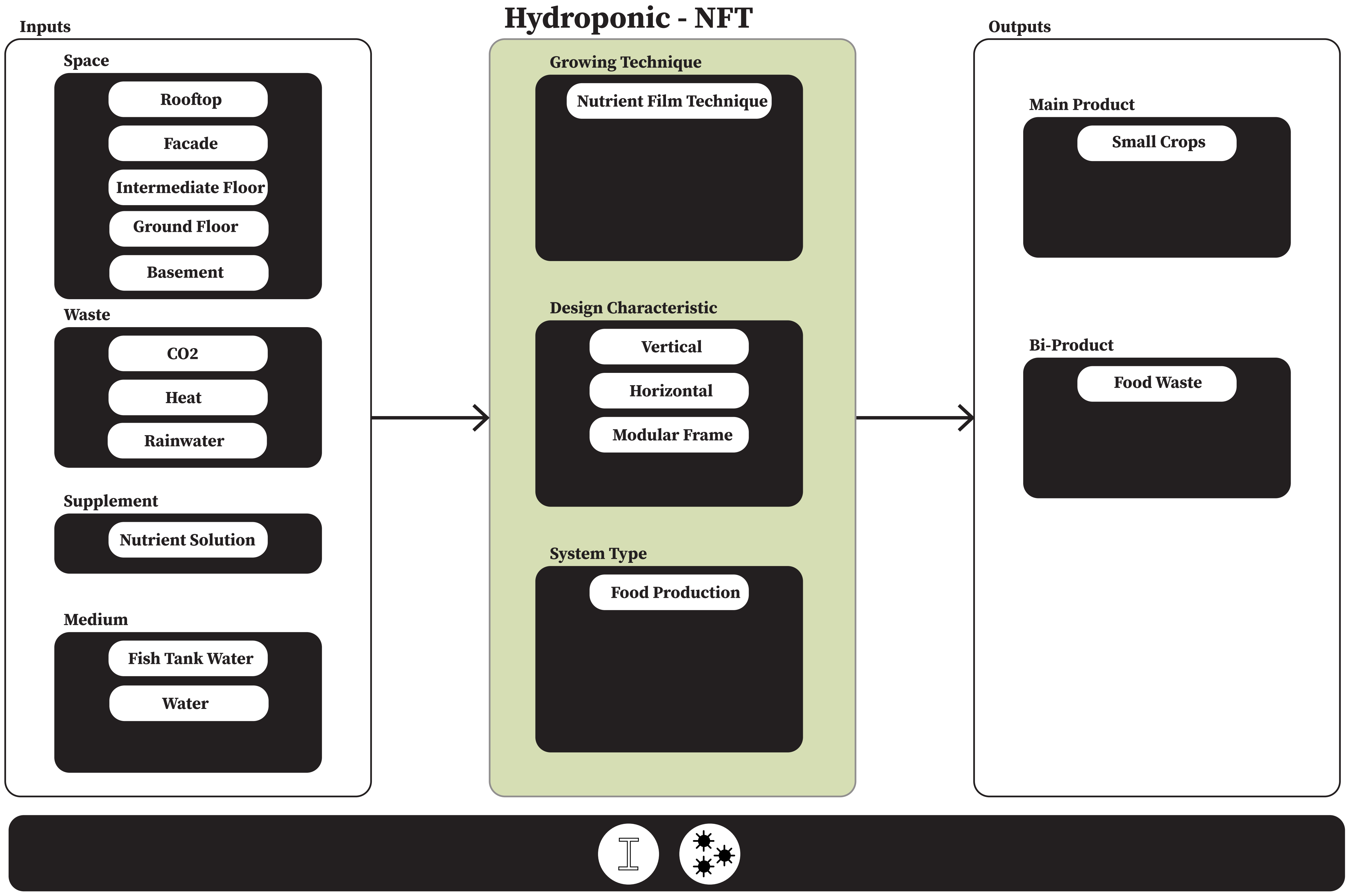
A.7 Aeroponics (to be updated)



Notes:

- High pressure nozzles under each plant spray nutrient rich mist to the roots directly (Jenkins, 2018).
- Crops are placed in a watertight box and their crowns are supported by the outer shell of the box (Jenkins, 2018).
- This system is not commonly used for conventional food production but it is used in laboratories (Jenkins, 2018).

A.8 Hydroponics - NFT (to be updated)



Hydroponic - NFT

Inputs

Space
- Rooftop
- Facade
- Intermediate Floor
- Ground Floor
- Basement

Waste
- CO2
- Heat
- Rainwater

Supplement
- Nutrient Solution

Medium
- Fish Tank Water
- Water

Growing Technique
- Nutrient Film Technique

Design Characteristic
- Vertical
- Horizontal
- Modular Frame

System Type
- Food Production

Outputs

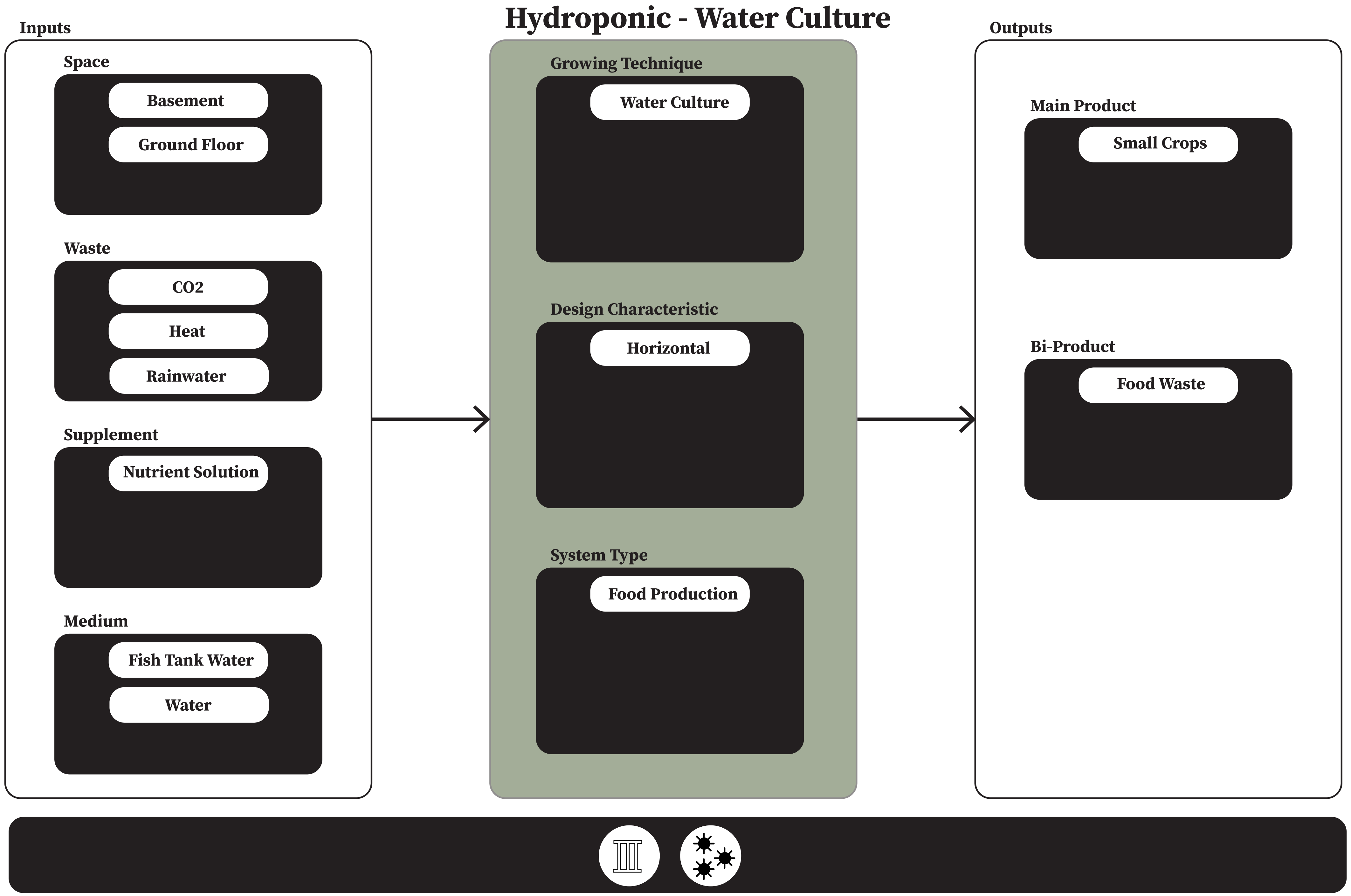
Main Product
- Small Crops

Bi-Product
- Food Waste

Notes:

- Total length of the growing channels should not exceed 15 meters in order to supply the last crop within the system with sufficient nutrient concentration (Jenkins, 2018)
- Each growing channel should be shorter than 4.6 m (Jenkins, 2018).
- To ensure adequate water flow the channels should have a minimum inclination of 2% as stated by the same author (Jenkins, 2018).

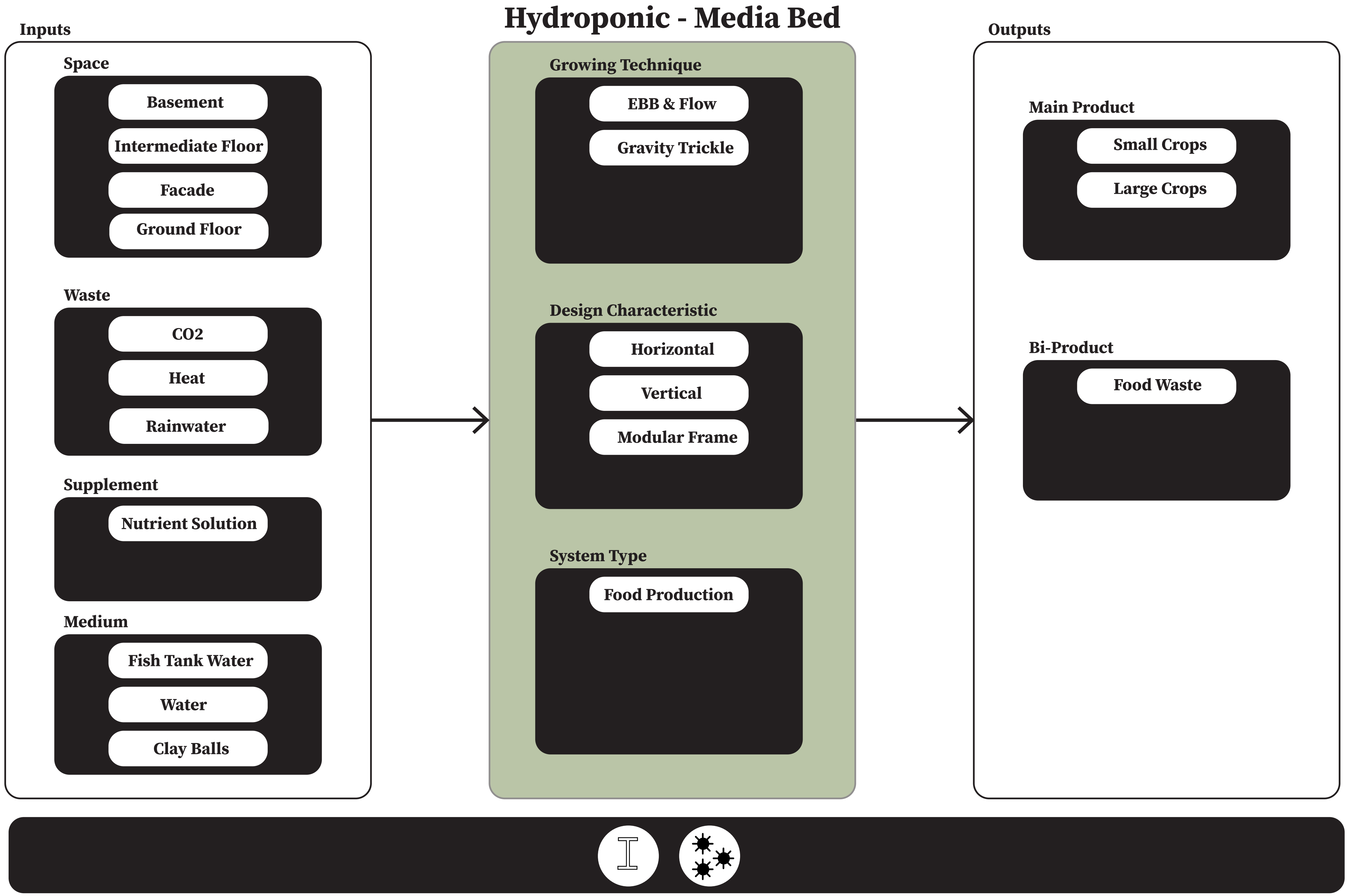
A.9 Hydroponics - Water Culture (to be updated)



Notes:

- Roots of the crops are suspended in nutrient rich water about 300mm, while the plants are supported with a floating raft above water (Jenkins, 2018).
- Nutrient rich water is pumped from one end of the raceway, it moves along the raceway to deliver necessary nutrients for crop growth, and exits the system at the end of the raceway to the reservoir (Jenkins, 2018).
- The water can be heated or kept cool to improve productivity of food production depending on the climate (Jenkins, 2018).

A.10 Hydroponics - Media Beds (to be updated)



Hydroponic - Media Bed

Inputs

Space
- Basement
- Intermediate Floor
- Facade
- Ground Floor

Waste
- CO2
- Heat
- Rainwater

Supplement
- Nutrient Solution

Medium
- Fish Tank Water
- Water
- Clay Balls

Hydroponic - Media Bed

Growing Technique
- EBB & Flow
- Gravity Trickle

Design Characteristic
- Horizontal
- Vertical
- Modular Frame

System Type
- Food Production

Outputs

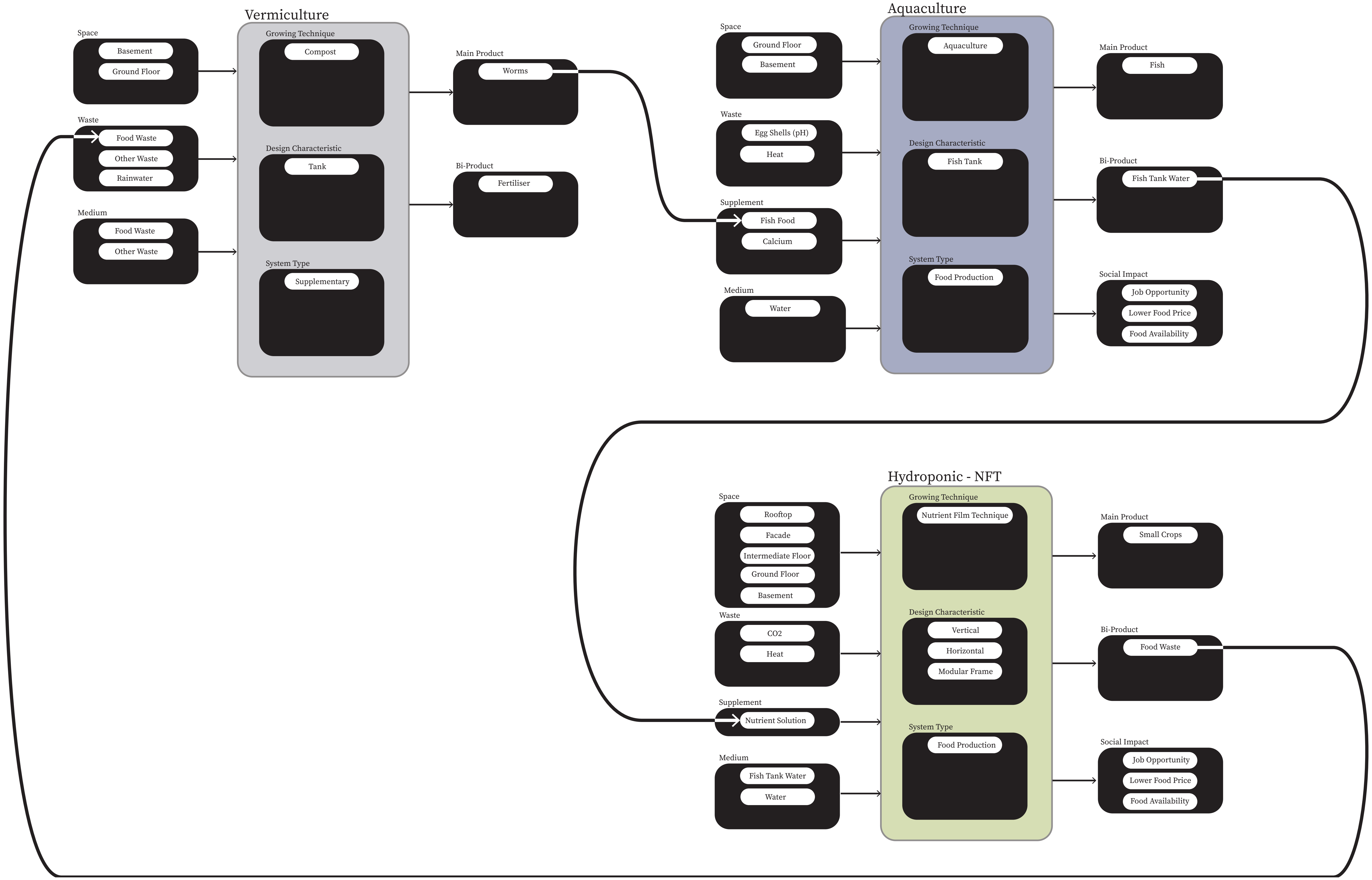
Main Product
- Small Crops
- Large Crops

Bi-Product
- Food Waste

Notes:

- These systems are scalable therefore the bed can be any size but minimum 300mm deep (Jenkins, 2018).
- While selecting the growing media good water retention, drainage, sufficient root oxygenation should be considered (Jenkins, 2018).
- In addition the media should not decompose or alter the chemical composition of nutrient solution according to the same source(Jenkins, 2018).

A.11 Potential Interrelation of Systems (to be updated)



A.12 Table of System Characteristics

System Name	Sytem Number	Min. Structural Capacity	Solar Exposure	Waste Inputs	Supplement	System Type	Outputs
Vermiculture	UF1	High (3)	Low (1)	Food Waste (W1)	None	Supplementary	Fish Food (S5)
				Sawdust (W2)			Fertiliser (S4)
				Paper (W3)			
				Rainwater (W6)			
Aquaculture	UF2	High (3)	Medium (2)	Residual Heat (W7)	Fish Food (S5)	Food-Producing Supplementary	Fish
							Nutrient Dense Water (S2)
Mushroom	UF3	Medium (2)	Low (1)	Sawdust (W2)	None	Food Production	Oyster Mushrooms
				Paper (W3)			Fertiliser (S4)
				Coffe Waste (W4)			
				Rainwater (W6)			
NFT	UF4	Low (1)	High (3)	CO2 (W5)	Nutrient Dense Water (S2)	Food Production	Leafy Greens
				Rainwater (W6)			Agricultural Waste
				Residual Heat (W7)			
Media Bed	UF5	Low (1)	High (3)	CO2 (W5)	Nutrient Dense Water (S2)	Food Production	Soft Fruits
				Rainwater (W6)			Leafy Greens
				Residual Heat (W7)			Agricultural Waste

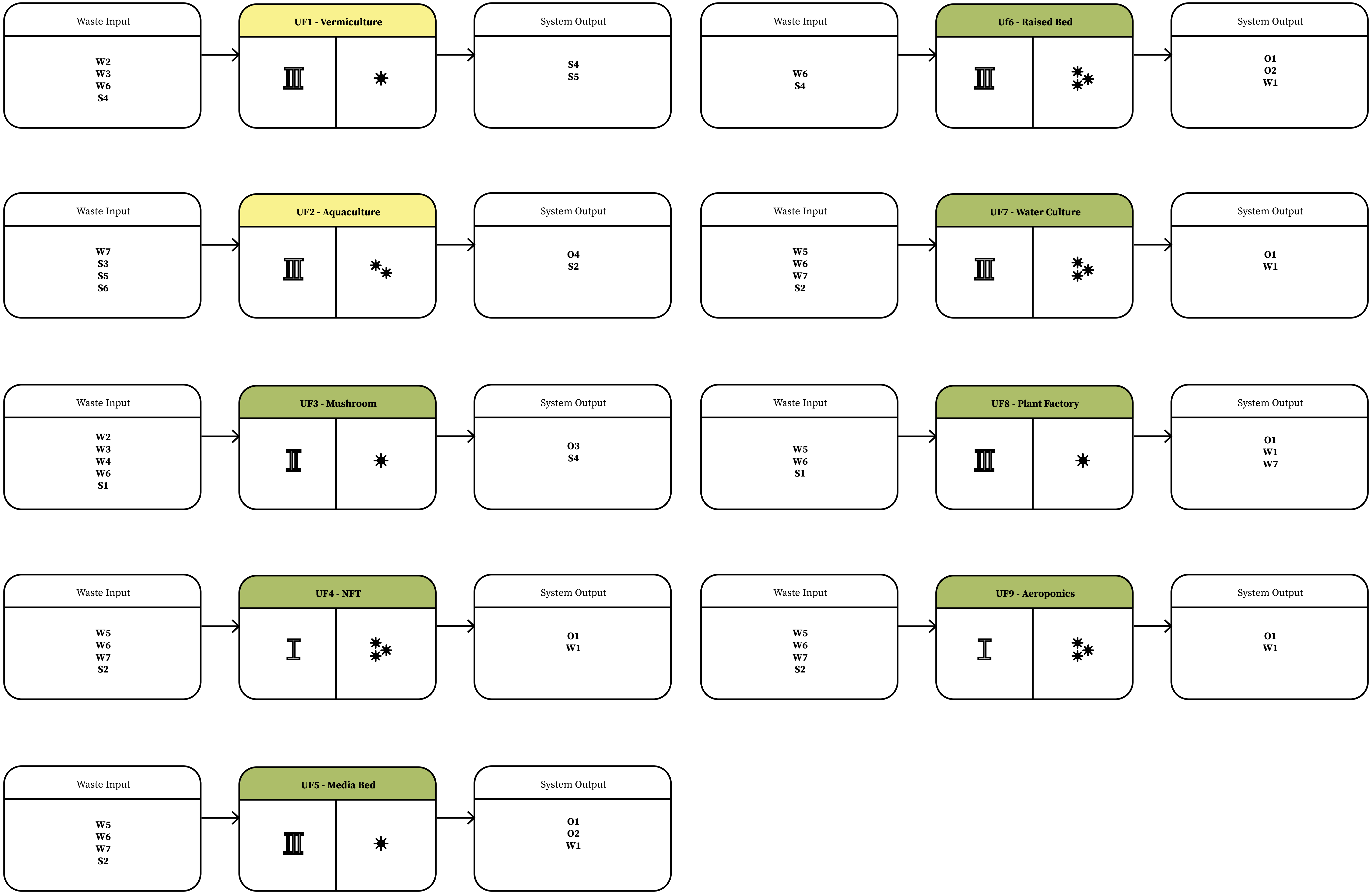
System Name	Sytem Number	Min. Structural Capacity	Solar Exposure	Waste Inputs	Supplement	System Type	Outputs
Raised Bed	UF6	High (3)	High (3)	Rainwater (W6)	Fertiliser (S4)	Food Production	Soft Fruits
							Leafy Greens
							Agricultural Waste
Water Culture	UF7	High (3)	High (3)	CO2 (W5)	Nutrient Dense Water (S2)	Food Production	Leafy Greens
				Rainwater (W6)			Agricultural Waste
				Residual Heat (W7)			
Plant Factory	UF8	High (3)	Low (1)	CO2 (W5)	Nutrient Dense Water (S2)	Food Production	Leafy Greens
				Rainwater (W6)			Agricultural Waste
							Residual Heat
Aeroponics	UF9	Low (1)	High (3)	CO2 (W5)	Nutrient Dense Water (S2)	Food Production	Leafy Greens
				Rainwater (W6)			Agricultural Waste
				Residual Heat (W7)			

Legend	
Waste Types	**Waste Number**
Organic Waste : Food	W1
Other Waste : Saw Dust	W2
Other Waste : Paper	W3
Organic Waste : Coffee	W4
CO2	W5
Rainwater	W6
Excess Heat Source	W7

Supplement Type	Supplement Number
Lime Bath	S1
Nutrient Solution	S2
Egg Shells	S3
Fertiliser	S4
Fish Food	S5
Calcium	S6

Output Types	Output Number
Small Crops	O1
Large Crops	O2
Mushroom	O3
Fish Food (Worms)	S5
Fish	O4
Heat	W7
Food Waste	W1
Fertilser (SMS/Vermicompost)	S4
Nutrient Solution (Fish Tank Water)	S2

A.13 System Characteristics



Waste Input
W2
W3
W6
S4
→ **UF1 - Vermiculture** Ⅲ ✱ →
System Output
S4
S5

Waste Input
W7
S3
S5
S6
→ **UF2 - Aquaculture** Ⅲ ✱✱ →
System Output
O4
S2

Waste Input
W2
W3
W4
W6
S1
→ **UF3 - Mushroom** Ⅰ ✱ →
System Output
O3
S4

Waste Input
W5
W6
W7
S2
→ **UF4 - NFT** Ⅰ ✱✱✱ →
System Output
O1
W1

Waste Input
W5
W6
W7
S2
→ **UF5 - Media Bed** Ⅲ ✱ →
System Output
O1
O2
W1

Waste Input
W6
S4
→ **Uf6 - Raised Bed** Ⅲ ✱✱✱ →
System Output
O1
O2
W1

Waste Input
W5
W6
W7
S2
→ **UF7 - Water Culture** Ⅲ ✱✱✱ →
System Output
O1
W1

Waste Input
W5
W6
S1
→ **UF8 - Plant Factory** Ⅲ ✱ →
System Output
O1
W1
W7

Waste Input
W5
W6
W7
S2
→ **UF9 - Aeroponics** Ⅰ ✱✱✱ →
System Output
O1
W1

A.14 PROJECT AIMS AND SYSTEM PRIORITISATION

Project Aims / Prioritisation	**Holistic Food Production**	**Maximum Productivity**	**Research**
Supplementary	Vermiculture	Vermiculture	Vermiculture
Food Producing Supplementary	Aquaculture	Aquaculture	Aquaculture
1	Mushroom	Plant Factory	Plant Factory
2	NFT	NFT	Aeroponics
3	Media Bed	Media Bed	Mushroom
4	Raised Bed	Mushroom	NFT
5	Water Culture	Aeroponics	Media Bed
6	Plant Factory	Raised Bed	Raised Bed
7	Aeroponics	Water Culture	Water Culture

Appendix B Computational Approaches

APPENDIX B COMPUTATIONAL APPROACHES

B.1 Suitability of Different Approaches

Approach Name	Description	Advantage	Why or Why Not Suitable?	Common Applications
🟢 Generative Design	(Caetano et al., 2020) In GD approaches, after starting the generative process, the system executes encoded instructions until the stop criterion is satisfied.	(Caetano et al., 2020) GD-based methods can generate complex outputs even from simple algorithmic descriptions	autonomous way of satisfying criterias	(Caetano et al., 2020) spatial configuration form finding rule based design decisions
🟠 Shape Grammar	(Gu & Behbahani, 2018) The main components of a shape grammar are shape rules that specify a set of spatial transformations of shapes.	(Gu & Behbahani, 2018) shape grammars are especially powerful when dealing with modularity efficient for reducing costs in mass customization of design	related to shapes and geometric entities	(Gu & Behbahani, 2018) form finding
🟠 Parallel Grammar	(Gu & Behbahani, 2018) An algebraic approach to implementing a shape grammar interpreter, where each rule set addresses a different aspect of the design, and together they are applied simultaneously	(Gu & Behbahani, 2018) set of rules running together to ensure all the rules are satisfied simultaneous application of criteria	applied with shape grammars	(Gu & Behbahani, 2018) spatial configuration
🟠 Parametric Grammar	(Gu & Behbahani, 2018) A selection of shapes and their transformations are generalized as one generic shape with parameterized features that can be adjusted according to a number of criteria and constraints			
🟢 Graph Grammar	(Gu & Behbahani, 2018) The shapes here are represented as a collection of vertices that are connected by edges. In the semantic approach, the graph's nodes are associated with the semantics of shapes.	(Gu & Behbahani, 2018) simplicity and abstractness represent shapes or forms in a topological manner reflects interrelations direction of connections shortest path weight of connection	doesnt include criteria fullfilment or rules of selection	(Gu & Behbahani, 2018) networks spatial configuration connectivity
🟢 Colour Grammar	(Gu & Behbahani, 2018) Uses colors as descriptive elements. A	(Gu & Behbahani, 2018) represent design attributes other than the actual colors, for example, materials, building elements	can be used to represent different systems hydroponics - shades of blue etc.	(Gu & Behbahani, 2018) visually rich rule based design spatial configuration form finding any attribute which can be represented with color
🟢 Sortal Grammar	(Stouffs and Krishnamurti, 2001) abstracts shapes (or descriptions) as sorts to optimize the manageability and flexibility of the grammar.	Sorts offer a representational flexibility where each sort additionally specifies its own match relation as a part of its behavior.	useful for matchin/selecting criteria if X space has X label = x then apply some rule	urban design generation describing attributes
🟢 Discursive Grammar	(Duarte, 2005) is a parallel parametric grammar with additional heuristics for managing rule selection.	(Duarte, 2005) can give weights to criteria A set of heuristics to guide generation towards designs that match the program	can give weights to criteria set of grammars for decision making and selecting the best fitting option based on criteria	(Duarte, 2005) space configuration decision making based on criteria
🟡 Heuristics	(Magdy & Eldaly, 2020) Heuristics are used to choose a rule for application at each step of the design generation or to constraint choice to a small number of rules	(Magdy & Eldaly, 2020) choosing the fittest option based on weight/importance of zone comparison of choices		

🟢 Suitable 🟠 Not Suitable 🟡 Maybe ⚫ Out of Scope

Approach Name	Description	Advantage	Why or Why Not Suitable?	Common Applications
🟠 **Replacement**	(Gu & Behbahani, 2018) a part of design is replaced with another to generate new alternatives, usually based on certain rules.			
🟠 L - System	(Gu & Behbahani, 2018) components in L-systems are mainly symbols (or symbolic geometries). Another difference is that L-systems predominantly feature multiple rule selection in each step.	(Gu & Behbahani, 2018) L-systems suitable for design situations with small components repeating in patterns over a relatively large extent.	related to geometry and repeated patterns	(Gu & Behbahani, 2018) plant modelling street design
🟠 Parametric Combination / Associative Geometry	(Gu & Behbahani, 2018) In parametric combination, the replacement affects features such as the proportions and dimensions of the design rather than the entire entity of shapes or components		related to geometric attributes	form finding
🟠 **Evolution**	(Gu & Behbahani, 2018) The design alternatives are matched against a set of fitness functions in each step; the next step continues from the "fittest" alternatives for maximal optimization. genetic algorithms is a evolutionary mechanism	(Gu & Behbahani, 2018) fittest / optimal solution is given		
🟠 Evolutionary Algorithms (EA)	(Lian et al., 2010) evolutionary algorhytmns mimic natural selection and natural genetics in which a biological population evolves over generations to adapt to an environmnet by selection crossover and random mutation	(Lian et al., 2010) suitable for multi objective optimisation	(Lian et al., 2010) inreased requirement in computation resources...	
🟠 **Agent Interaction**	(Gu & Behbahani, 2018) agent interaction, a society of agents navigate through the design space and generate design alternatives based on their interactions with the context and other related elements	(Gu & Behbahani, 2018) An obvious advantage of such a method is that the generated outcomes can be closely aligned to the requirement and context, minimizing the need for post-design evaluation.	(Gu & Behbahani, 2018) They can be visually or spatially restricted because their design space is geometrically limited (e.g., rigid grids for cellular automata).	
🟠 Swarm Intelligence	(Magdy & Eldaly, 2020) Swarm intelligence agents are working and living together according to a set of rules. These rules control the way they interact with each other	(Magdy & Eldaly, 2020) achieving design aims (when used with optimization algorithms), applying the strategy of learn and respond to provide alternatives, and generating alternatives for a design by managing the design process.	most common in spatial configuration	(Magdy & Eldaly, 2020) best order of rooms, open spaces, directions, view, privacy and modules configuration.
🟠 Cellular Automata	(Herr & Ford, 2016) collection of "colored" cells on a grid of specified shape that evolves through a number of discrete time steps according to a set of rules based on the states of neighboring cells.	(Herr & Ford, 2016) potential complex outcome simplicity of CA maechanism Adaptation of CA rulesAdaptation of CA cell shapes and scale	patterns generated across entire CA cell lattices are often intricate and difficult to predict	(Herr & Ford, 2016) spatial configuration form structural elements
🟠 **Performance Based Generative Design**	(Caetano et al., 2020) the designer sets a performance target, and an algorithm finds design solutions that best approximate the desired goal.			facade design climate design

🟢 Suitable 🟠 Not Suitable 🟡 Maybe ⚫ Out of Scope

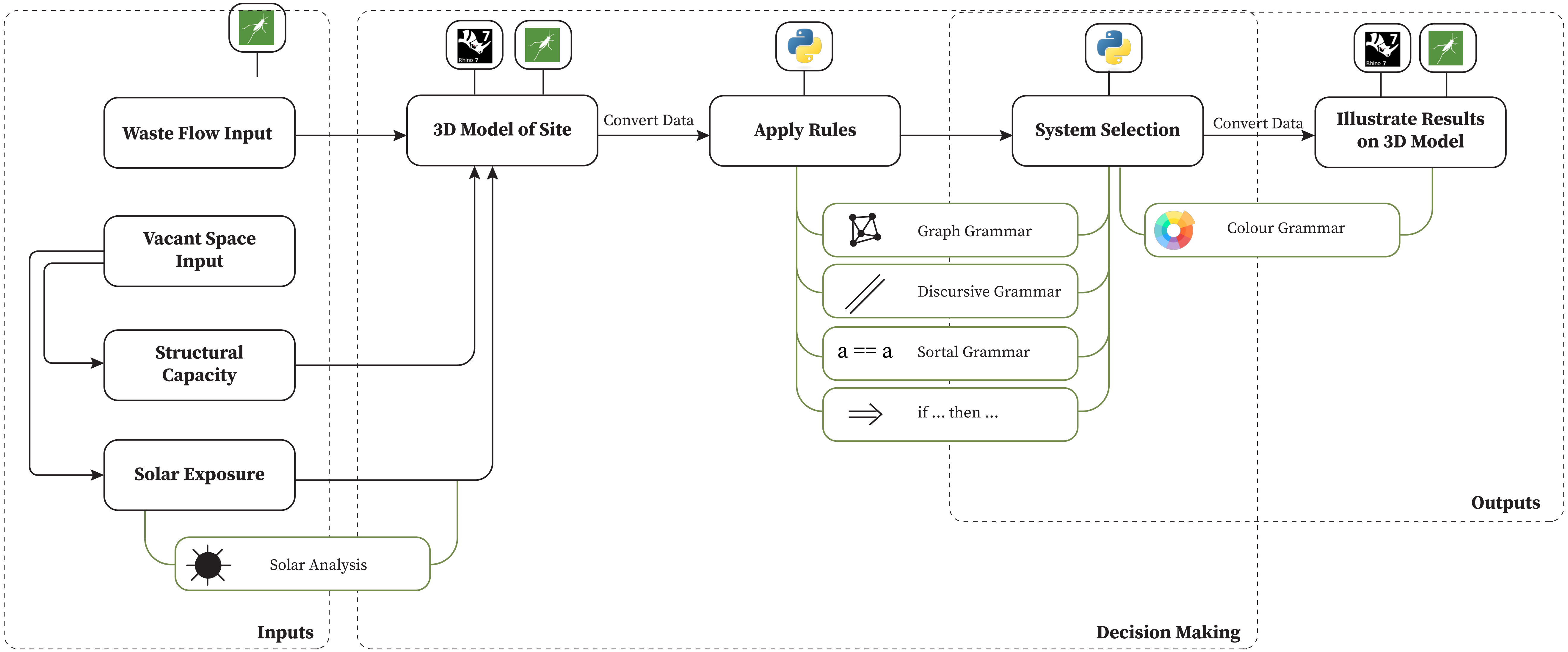
Approach Name	Description	Advantage	Why or Why Not Suitable?	Common Applications
Topology Optimisation	(Topology Optimization, n.d) Based on mathematical optimization algorithms, it is a systematic yet also highly creative design method, with a wide variety of industrial applications.			structral design
Algorithmic Design	(Caetano et al., 2020) "set of mathematical instructions or rules that [...] will help calculate an answer to a problem."			
Machine Learning	(Chatzikonstantinou, 2021) this field is concerned with how to construct computer programs that automatically improve with experience			
Artificial Neural Networks	(Chatzikonstantinou, 2021) similar to neural system in living organisms knowladge is stored in the weights of connections and learning occurs by adaptation of these weights	(Chatzikonstantinou, 2021) they can approximate any continuous function	(Chatzikonstantinou, 2021) complex functions may require vast networks for efficient approximation	
Feed Forward Networks	(Chatzikonstantinou, 2021) connection between adjacent layers connections between any layer to the next one the neuron in each layer receives a weighted summation of the output of each neuron in the previous layer as input			
Radial Basis Function Networks	(Chatzikonstantinou, 2021) have two layers 1st layer: radial basis functions that acts as pattern detectors at the input space 2nd layer: forms the output pf the network hrough a linear combination of the rbf outputs	(Chatzikonstantinou, 2021) allows linear least squares techniques to be used to determine the output weights without resorting to non linear optimization(which may be costly)		
Random Forests	(Chatzikonstantinou, 2021) prediction model based on ensembles of decision trees decision trees = hypotheses created by constructing a binary tree with simple decision functions at the internal nodes and output lvl of leaves			
Support Vector Machines	(Chatzikonstantinou, 2021) baed on principle of structural risk minimization which enable better prediction generilization while enabling limiting of the number of learning patterns			
Surrogate Modeling	(Chatzikonstantinou, 2021) appliaction in the field of machine learning in optimizatio problems where repeated sampling of the objective function is nontrivial due to computational complexity	(Chatzikonstantinou, 2021) if supplemented in another algorithms, faster calculation		(Chatzikonstantinou, 2021) daylight performance,energy consumption,visual comfort complex optimisation problems

● Suitable ● Not Suitable ● Maybe ● Out of Scope

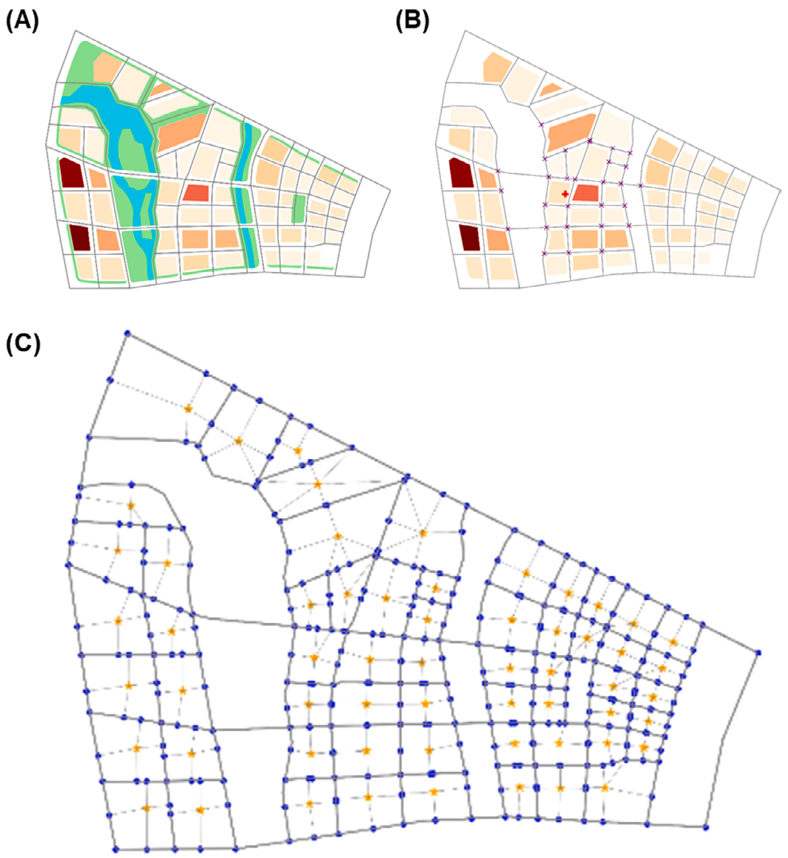
Approach Name	Description	Advantage	Why or Why Not Suitable?	Common Applications
Parametric Design	(Caetano et al., 2020) "a numerical or other measurable factor forming one of a set that defines a system or sets the conditions of its operation,"	(Caetano et al., 2020) parameter input rule based	(Caetano et al., 2020) use of parameters to design	(Caetano et al., 2020) form finding anything that can be designed with parameters
Rule Based Decision Making	(Cubukcuoglu et al., 2019) In other words, the decisions regarding the distribution of the neighborhood functions are based on "if-then" rules considering design goals,	(Cubukcuoglu et al., 2019) rule based	rules can be implemented to design the best option satisfying the rules	(Cubukcuoglu et al., 2019) Rule Based Decision Making

● Suitable ● Not Suitable ● Maybe ● Out of Scope

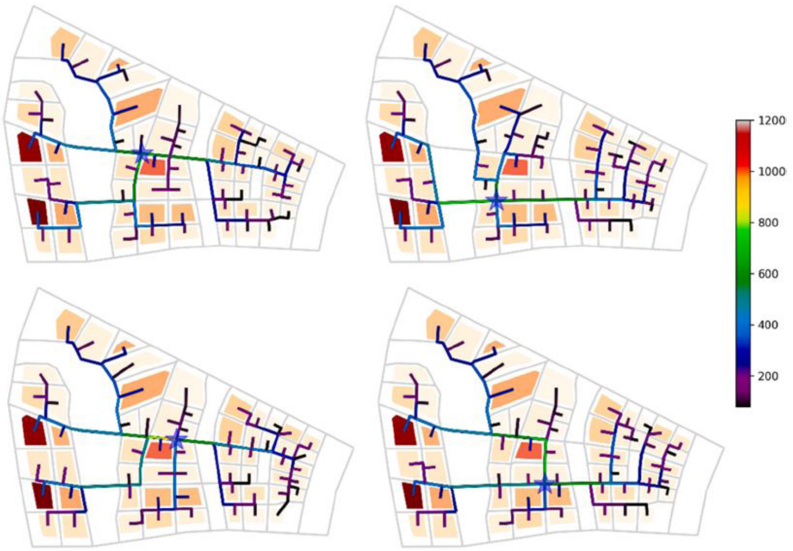
B.2 Draft Decision Making Process & Computational Approaches



B.3 Graph Based Approach for District Heating



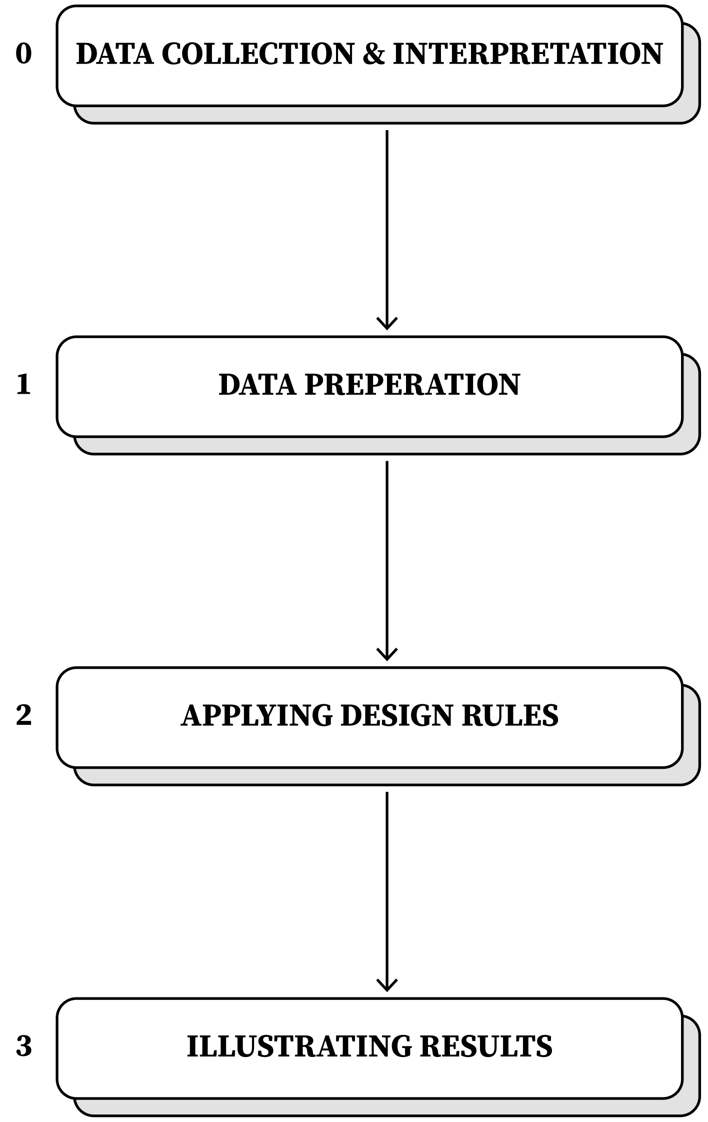
(A) Geographical information input including road network, river, green areas and building blocks
(B) Load center and site location candidates
(C) Graph's nodes and edges.by Ho et al., 2021



Good Performing Network Designs
Different pipeline colours represent pipe diameters
by Ho et al., 2021

Appendix C Methodology & User Interface

APPENDIX C METHODOLOGY & USER INTERFACE



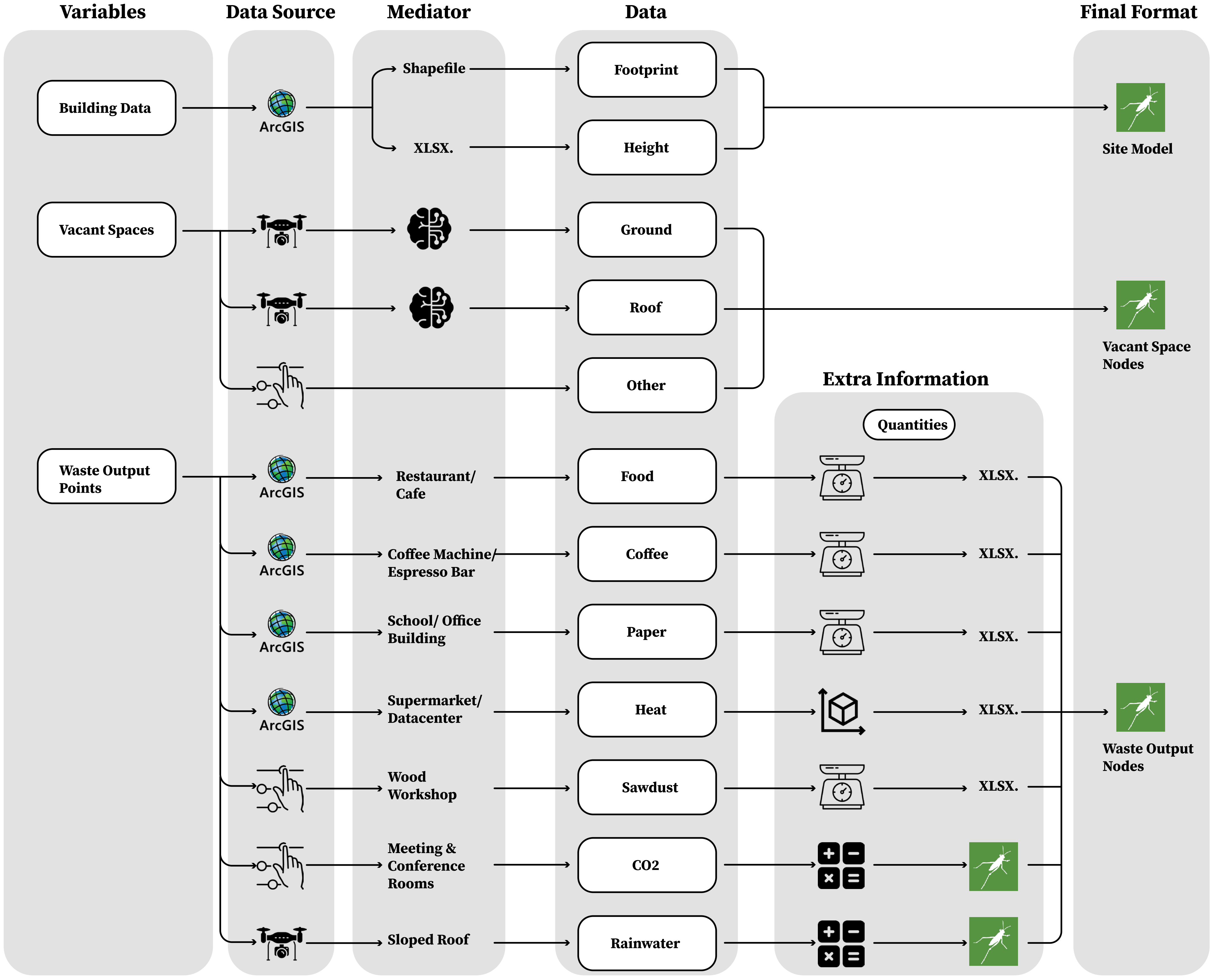
0 DATA COLLECTION & INTERPRETATION

1 DATA PREPERATION

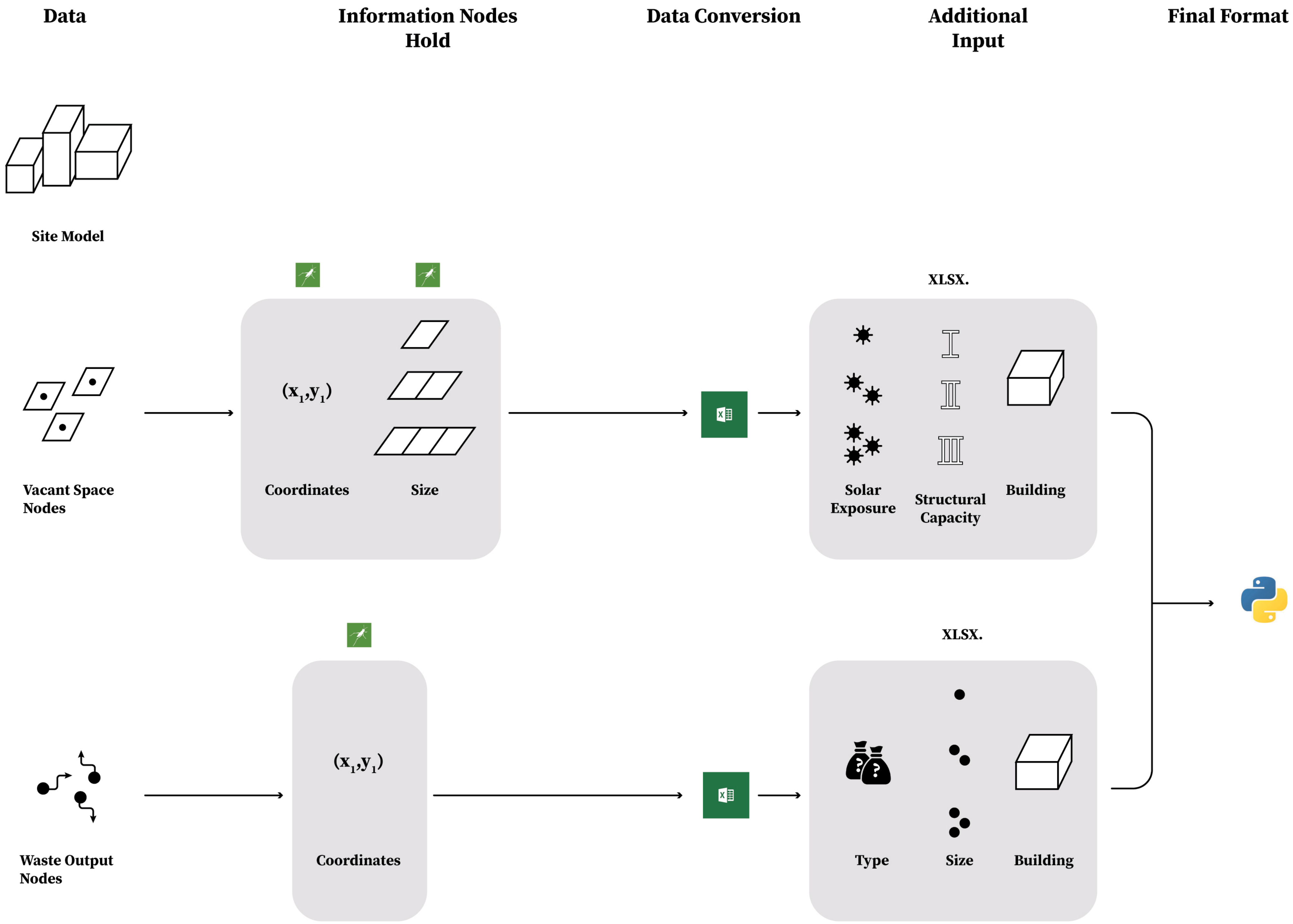
2 APPLYING DESIGN RULES

3 ILLUSTRATING RESULTS

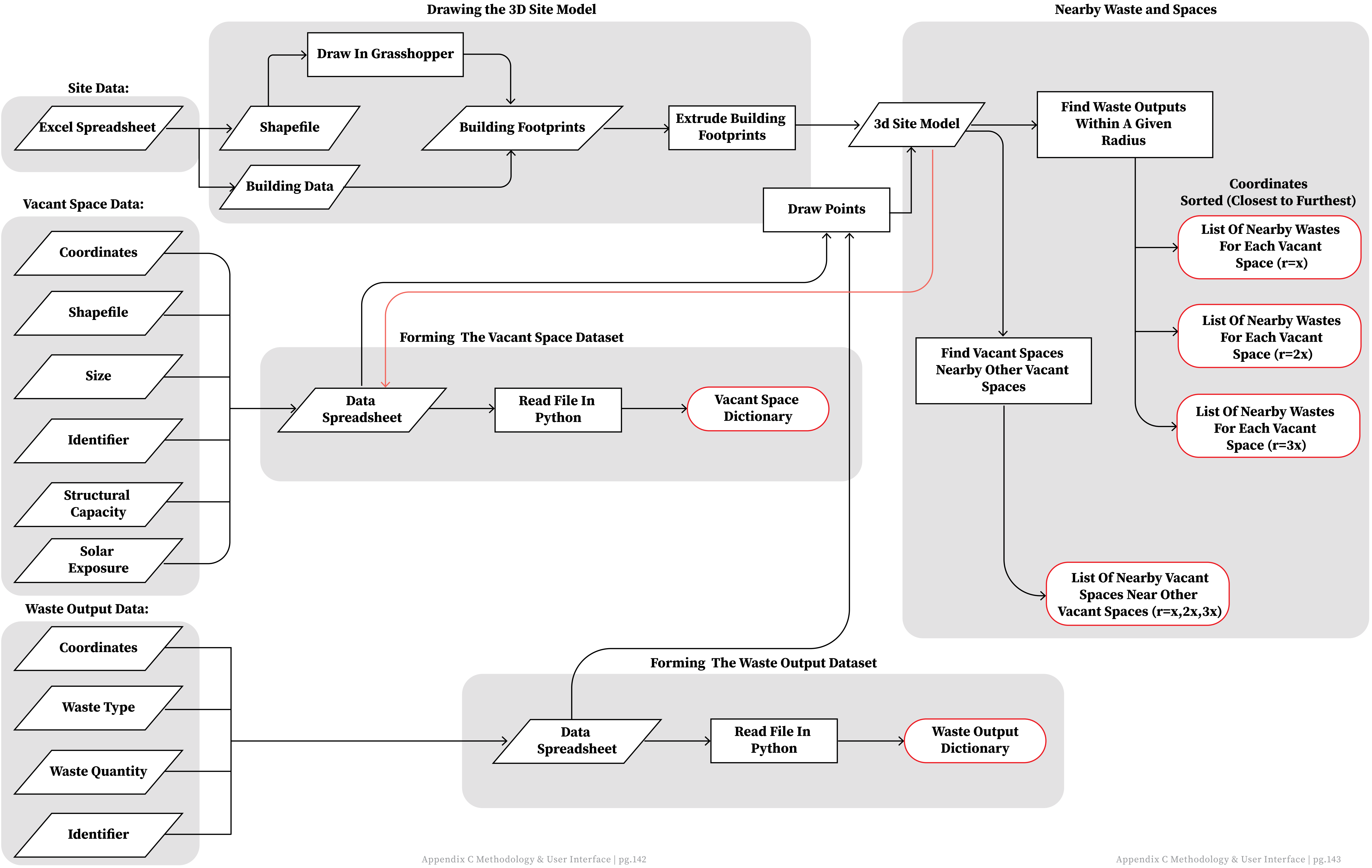
C.1 Automated Data Collection



C.2 Data Flow


Data
Information Nodes Hold
Data Conversion
Additional Input
Final Format
Site Model
Vacant Space Nodes
(x_1,y_1)
Coordinates
Size
XLSX.
Solar Exposure
Structural Capacity
Building
Waste Output Nodes
(x_1,y_1)
Coordinates
XLSX.
Type
Size
Building

C.3 Data Collection & Interpretation



Drawing the 3D Site Model

Nearby Waste and Spaces

Site Data:

Excel Spreadsheet

Shapefile

Building Data

Draw In Grasshopper

Building Footprints

Extrude Building Footprints

3d Site Model

Draw Points

Find Waste Outputs Within A Given Radius

Coordinates Sorted (Closest to Furthest)

List Of Nearby Wastes For Each Vacant Space (r=x)

List Of Nearby Wastes For Each Vacant Space (r=2x)

List Of Nearby Wastes For Each Vacant Space (r=3x)

Find Vacant Spaces Nearby Other Vacant Spaces

List Of Nearby Vacant Spaces Near Other Vacant Spaces (r=x,2x,3x)

Vacant Space Data:

Coordinates

Shapefile

Size

Identifier

Structural Capacity

Solar Exposure

Forming The Vacant Space Dataset

Data Spreadsheet

Read File In Python

Vacant Space Dictionary

Waste Output Data:

Coordinates

Waste Type

Waste Quantity

Identifier

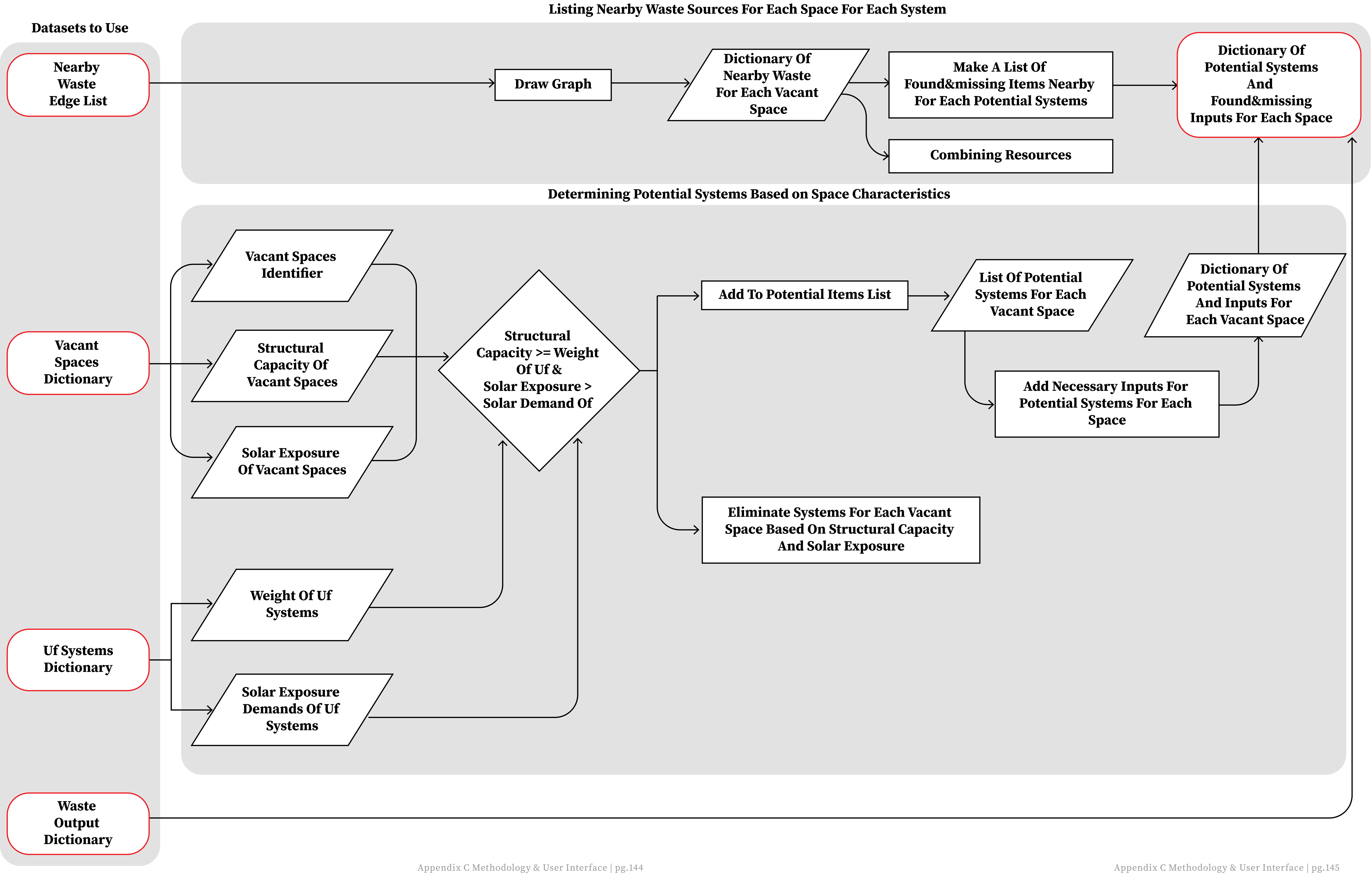
Forming The Waste Output Dataset

Data Spreadsheet

Read File In Python

Waste Output Dictionary

C.4 Data Processing



Datasets to Use

Nearby Waste Edge List

Vacant Spaces Dictionary

Uf Systems Dictionary

Waste Output Dictionary

Listing Nearby Waste Sources For Each Space For Each System

Draw Graph

Dictionary Of Nearby Waste For Each Vacant Space

Make A List Of Found&missing Items Nearby For Each Potential Systems

Combining Resources

Dictionary Of Potential Systems And Found&missing Inputs For Each Space

Determining Potential Systems Based on Space Characteristics

Vacant Spaces Identifier

Structural Capacity Of Vacant Spaces

Solar Exposure Of Vacant Spaces

Weight Of Uf Systems

Solar Exposure Demands Of Uf Systems

Structural Capacity >= Weight Of Uf & Solar Exposure > Solar Demand Of

Add To Potential Items List

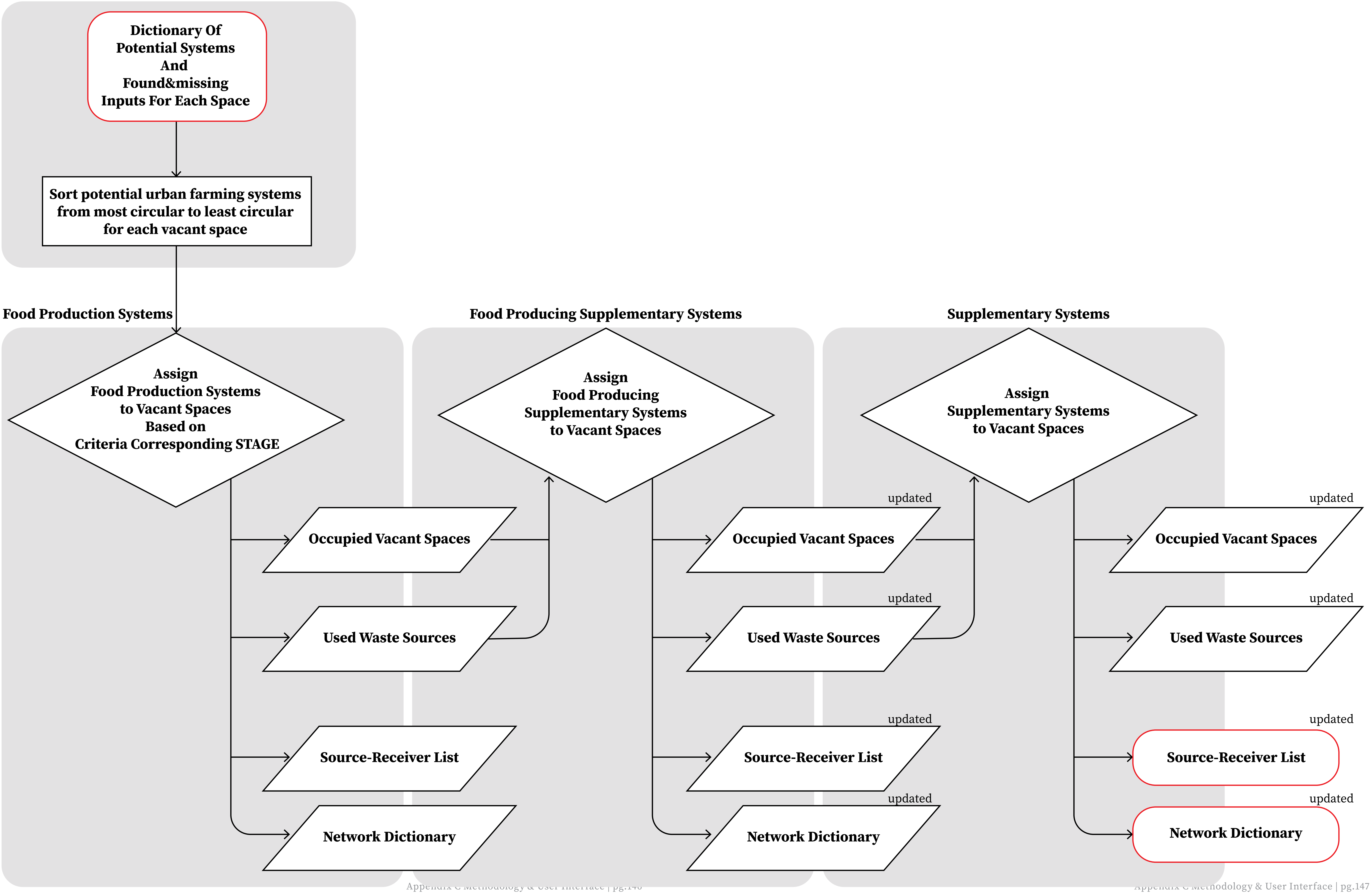
List Of Potential Systems For Each Vacant Space

Add Necessary Inputs For Potential Systems For Each Space

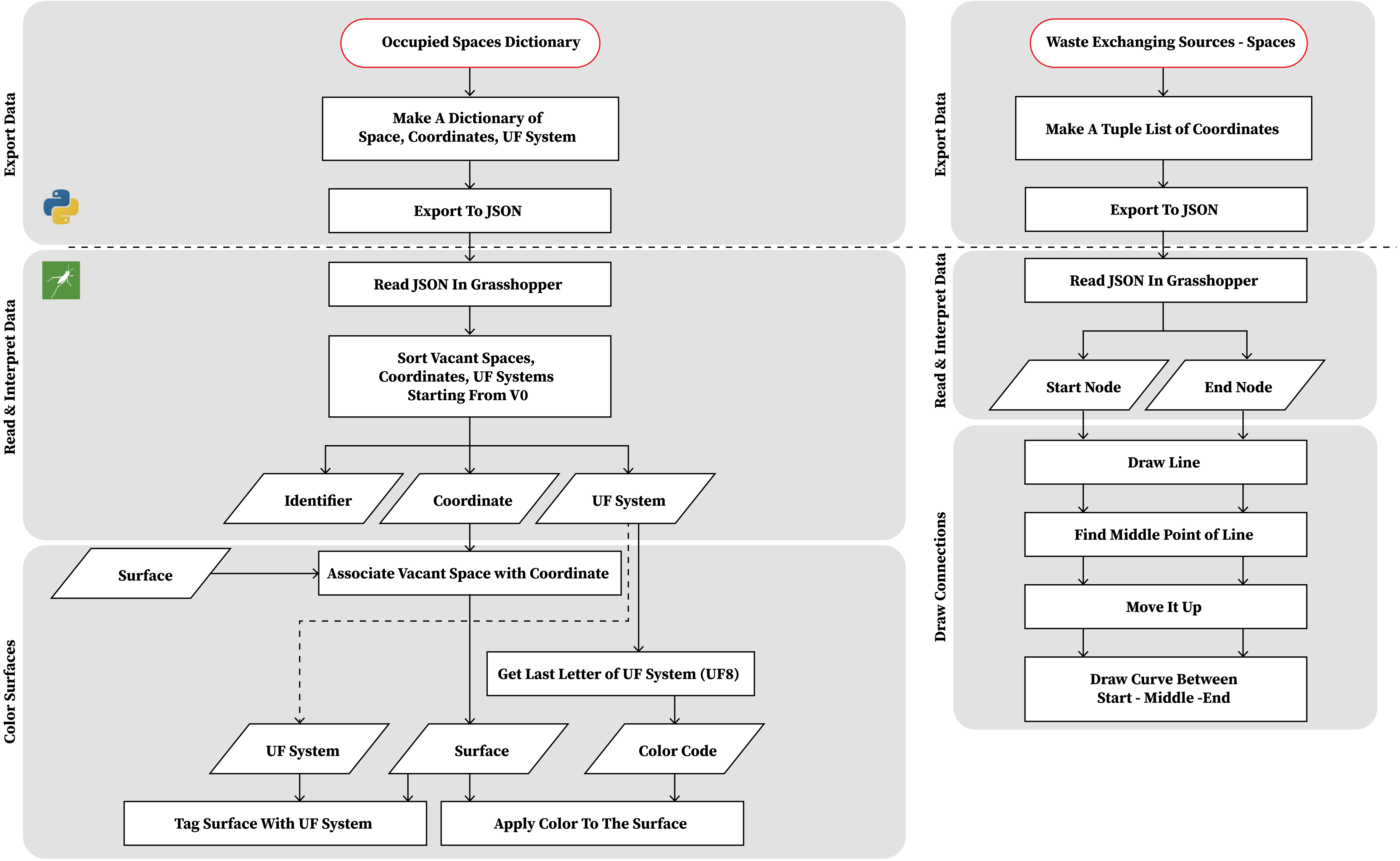
Dictionary Of Potential Systems And Inputs For Each Vacant Space

Eliminate Systems For Each Vacant Space Based On Structural Capacity And Solar Exposure

C.5 Applying Design Rules


Dictionary Of Potential Systems And Found&missing Inputs For Each Space
Sort potential urban farming systems from most circular to least circular for each vacant space
Food Production Systems
Food Producing Supplementary Systems
Supplementary Systems
Assign Food Production Systems to Vacant Spaces Based on Criteria Corresponding STAGE
Occupied Vacant Spaces
Used Waste Sources
Source-Receiver List
Network Dictionary
Assign Food Producing Supplementary Systems to Vacant Spaces
updated
Occupied Vacant Spaces
updated
Used Waste Sources
updated
Source-Receiver List
updated
Network Dictionary
Assign Supplementary Systems to Vacant Spaces
updated
Occupied Vacant Spaces
updated
Used Waste Sources
updated
Source-Receiver List
updated
Network Dictionary

C.6 Visualising



Export Data

Occupied Spaces Dictionary

Make A Dictionary of Space, Coordinates, UF System

Export To JSON

Read & Interpret Data

Read JSON In Grasshopper

Sort Vacant Spaces, Coordinates, UF Systems Starting From V0

Identifier

Coordinate

UF System

Color Surfaces

Surface

Associate Vacant Space with Coordinate

Get Last Letter of UF System (UF8)

UF System

Surface

Color Code

Tag Surface With UF System

Apply Color To The Surface

Export Data

Waste Exchanging Sources - Spaces

Make A Tuple List of Coordinates

Export To JSON

Read & Interpret Data

Read JSON In Grasshopper

Start Node

End Node

Draw Connections

Draw Line

Find Middle Point of Line

Move It Up

Draw Curve Between Start - Middle -End

C.7 Stages

2 **APPLYING DESIGN RULES**

STAGE 1 (search radius=x)

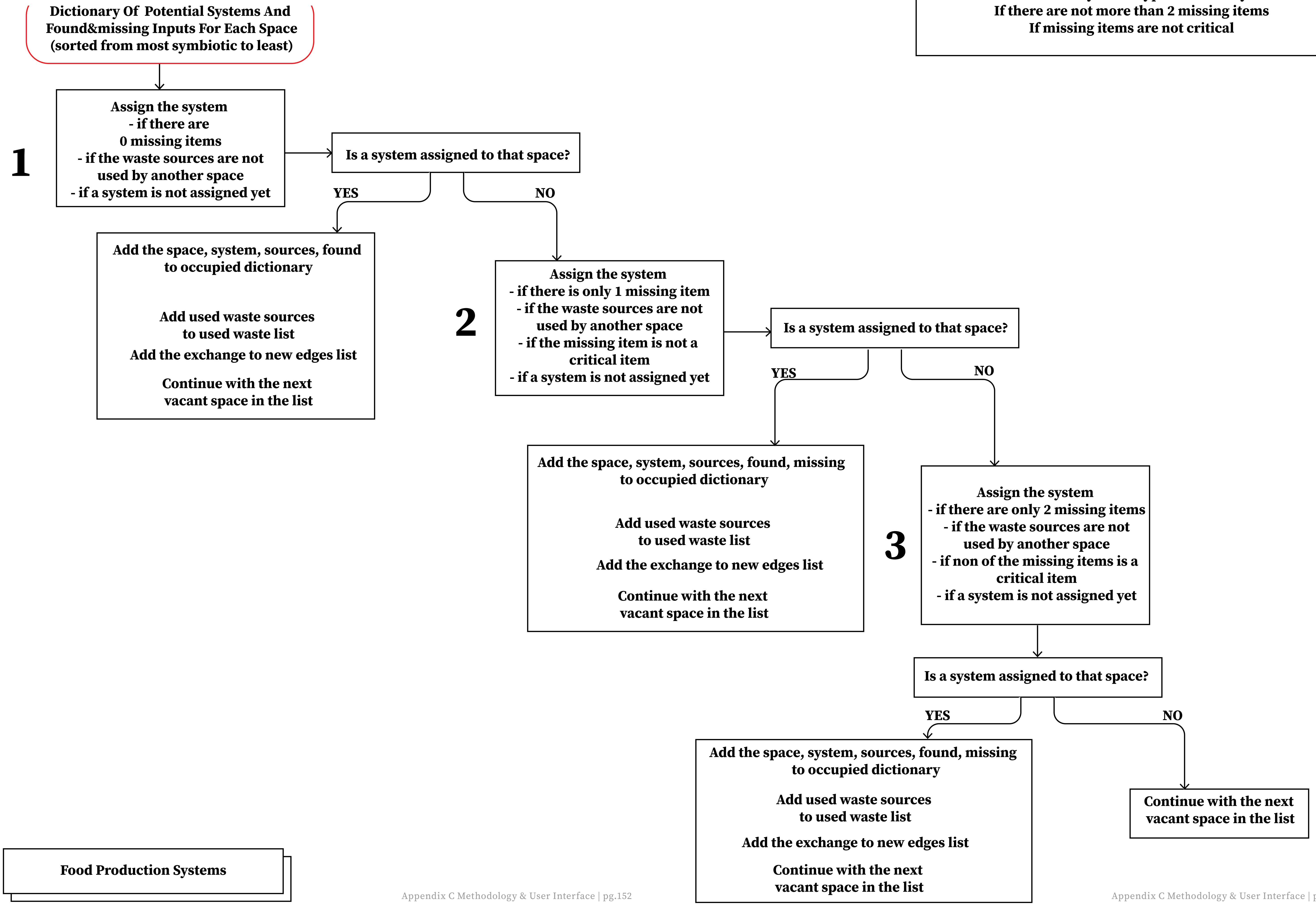
STAGE 2 (search radius=2x)

STAGE 3 (search radius=3x)

STAGE 4 (search radius=3x & occupy all)

C.7 Flowchart

Assign food production system to the vacant space
If necessary waste types are nearby
If there are not more than 2 missing items
If missing items are not critical



Dictionary Of Potential Systems And Found&missing Inputs For Each Space (sorted from most symbiotic to least)

1

Assign the system
- if there are 0 missing items
- if the waste sources are not used by another space
- if a system is not assigned yet

Is a system assigned to that space?

YES

Add the space, system, sources, found to occupied dictionary

Add used waste sources to used waste list

Add the exchange to new edges list

Continue with the next vacant space in the list

NO

2

Assign the system
- if there is only 1 missing item
- if the waste sources are not used by another space
- if the missing item is not a critical item
- if a system is not assigned yet

Is a system assigned to that space?

YES

Add the space, system, sources, found, missing to occupied dictionary

Add used waste sources to used waste list

Add the exchange to new edges list

Continue with the next vacant space in the list

NO

3

Assign the system
- if there are only 2 missing items
- if the waste sources are not used by another space
- if non of the missing items is a critical item
- if a system is not assigned yet

Is a system assigned to that space?

YES

Add the space, system, sources, found, missing to occupied dictionary

Add used waste sources to used waste list

Add the exchange to new edges list

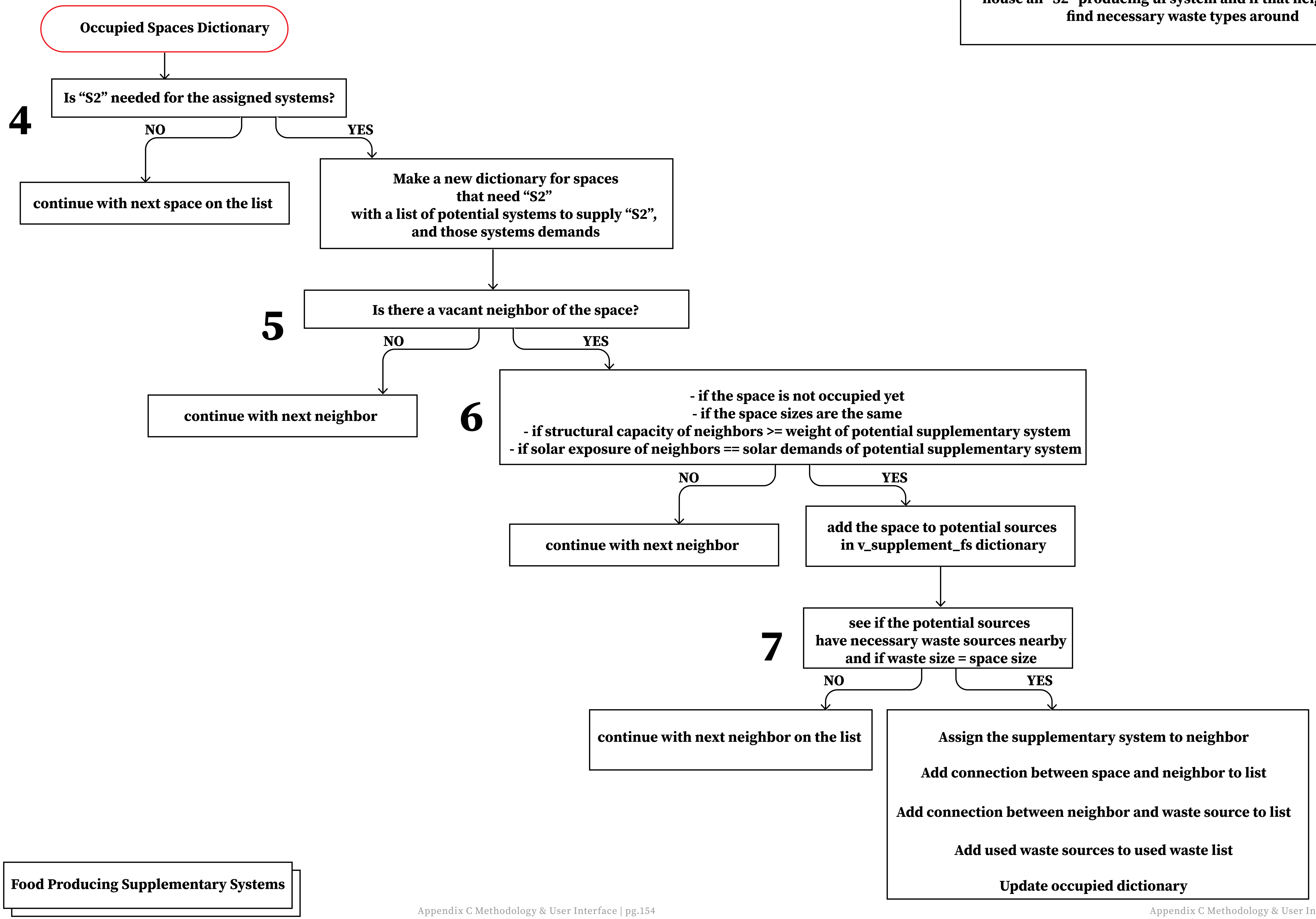
Continue with the next vacant space in the list

NO

Continue with the next vacant space in the list

C.7 Flowchart

Assign food producing supplementary system to the neighbor If the space needs “S2” , and has a neighbor which can house an “S2” producing uf system and if that neighbor can find necessary waste types around



Occupied Spaces Dictionary

4

Is “S2” needed for the assigned systems?

NO

continue with next space on the list

YES

Make a new dictionary for spaces
that need “S2”
with a list of potential systems to supply “S2”,
and those systems demands

5

Is there a vacant neighbor of the space?

NO

continue with next neighbor

YES

6

- if the space is not occupied yet
- if the space sizes are the same
- if structural capacity of neighbors >= weight of potential supplementary system
- if solar exposure of neighbors == solar demands of potential supplementary system

NO

continue with next neighbor

YES

add the space to potential sources
in v_supplement_fs dictionary

7

see if the potential sources
have necessary waste sources nearby
and if waste size = space size

NO

continue with next neighbor on the list

YES

Assign the supplementary system to neighbor

Add connection between space and neighbor to list

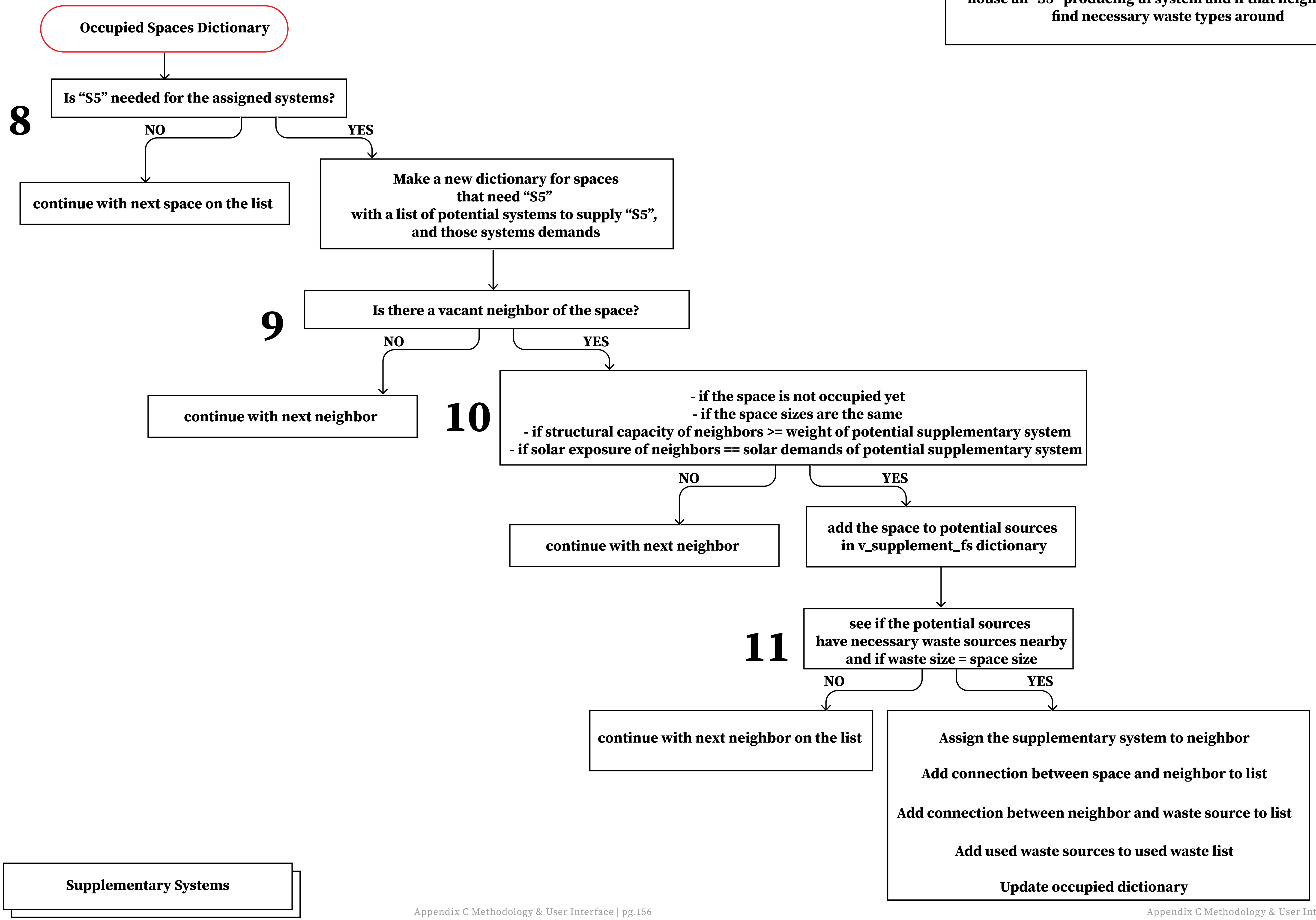
Add connection between neighbor and waste source to list

Add used waste sources to used waste list

Update occupied dictionary

C.7 Flowchart

Assign supplementary system to the neighbor
If the space needs “S5” , and has a neighbor which can house an “S5” producing uf system and if that neighbor can find necessary waste types around



Occupied Spaces Dictionary

8 Is “S5” needed for the assigned systems?

NO → continue with next space on the list

YES → Make a new dictionary for spaces that need “S5” with a list of potential systems to supply “S5”, and those systems demands

9 Is there a vacant neighbor of the space?

NO → continue with next neighbor

YES → 10

10
- if the space is not occupied yet
- if the space sizes are the same
- if structural capacity of neighbors >= weight of potential supplementary system
- if solar exposure of neighbors == solar demands of potential supplementary system

NO → continue with next neighbor

YES → add the space to potential sources in v_supplement_fs dictionary

11 see if the potential sources have necessary waste sources nearby and if waste size = space size

NO → continue with next neighbor on the list

YES →

Assign the supplementary system to neighbor

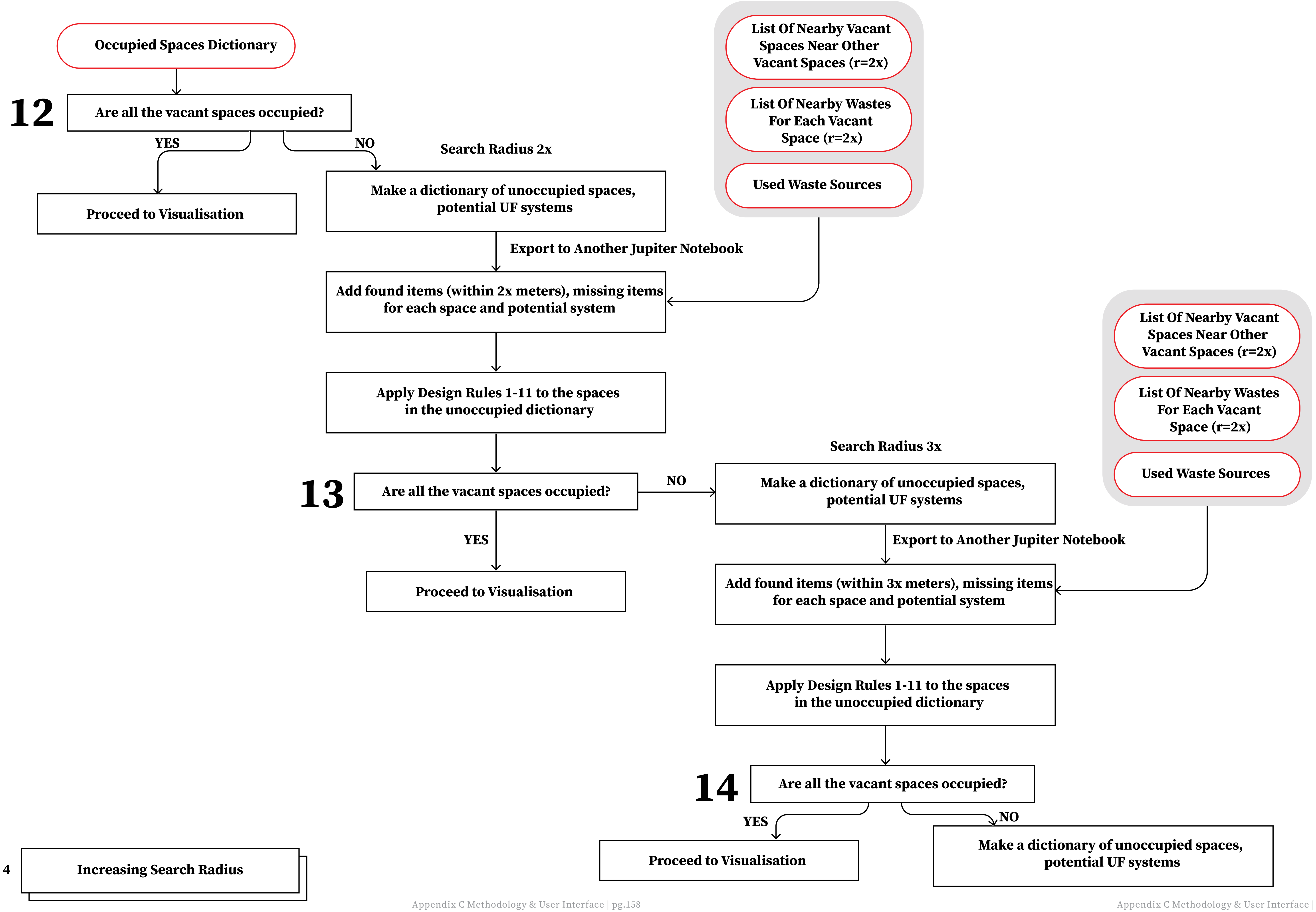
Add connection between space and neighbor to list

Add connection between neighbor and waste source to list

Add used waste sources to used waste list

Update occupied dictionary

C.7 Flowchart



Occupied Spaces Dictionary

12 Are all the vacant spaces occupied?

YES

Proceed to Visualisation

NO

Search Radius 2x

Make a dictionary of unoccupied spaces, potential UF systems

Export to Another Jupiter Notebook

List Of Nearby Vacant Spaces Near Other Vacant Spaces (r=2x)

List Of Nearby Wastes For Each Vacant Space (r=2x)

Used Waste Sources

Add found items (within 2x meters), missing items for each space and potential system

Apply Design Rules 1-11 to the spaces in the unoccupied dictionary

13 Are all the vacant spaces occupied?

YES

Proceed to Visualisation

NO

Search Radius 3x

Make a dictionary of unoccupied spaces, potential UF systems

Export to Another Jupiter Notebook

List Of Nearby Vacant Spaces Near Other Vacant Spaces (r=2x)

List Of Nearby Wastes For Each Vacant Space (r=2x)

Used Waste Sources

Add found items (within 3x meters), missing items for each space and potential system

Apply Design Rules 1-11 to the spaces in the unoccupied dictionary

14 Are all the vacant spaces occupied?

YES

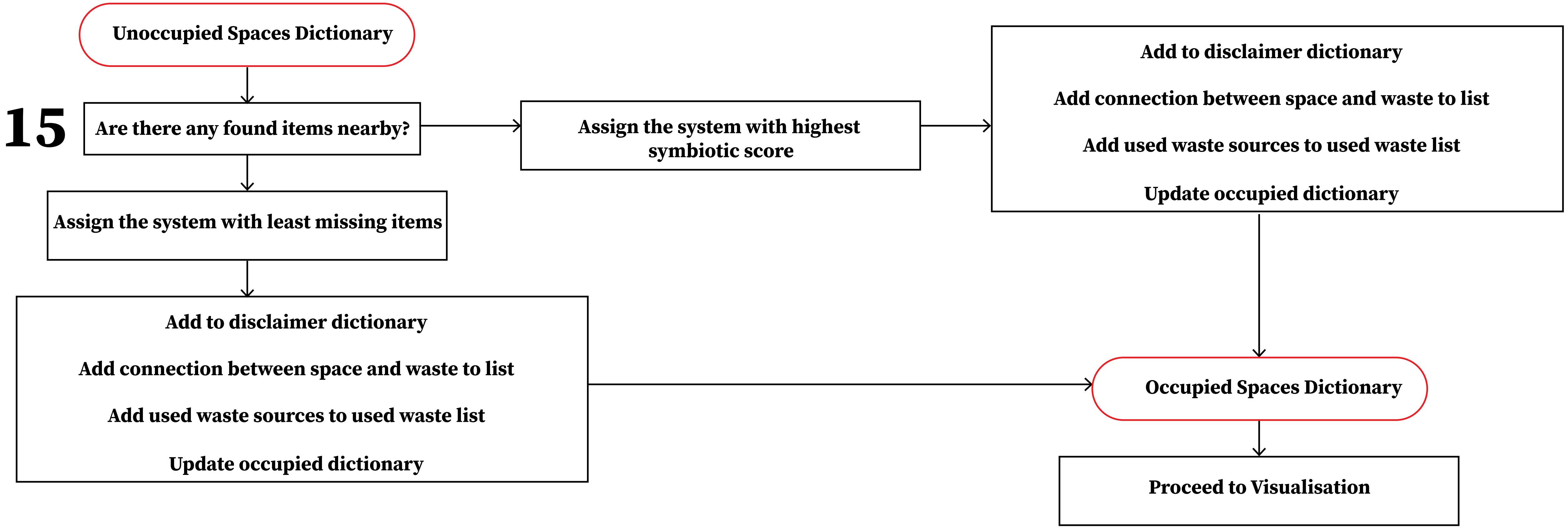
Proceed to Visualisation

NO

Make a dictionary of unoccupied spaces, potential UF systems

4 Increasing Search Radius

C.7 Flowchart



15

Unoccupied Spaces Dictionary

Are there any found items nearby?

Assign the system with highest symbiotic score

Add to disclaimer dictionary

Add connection between space and waste to list

Add used waste sources to used waste list

Update occupied dictionary

Assign the system with least missing items

Add to disclaimer dictionary

Add connection between space and waste to list

Add used waste sources to used waste list

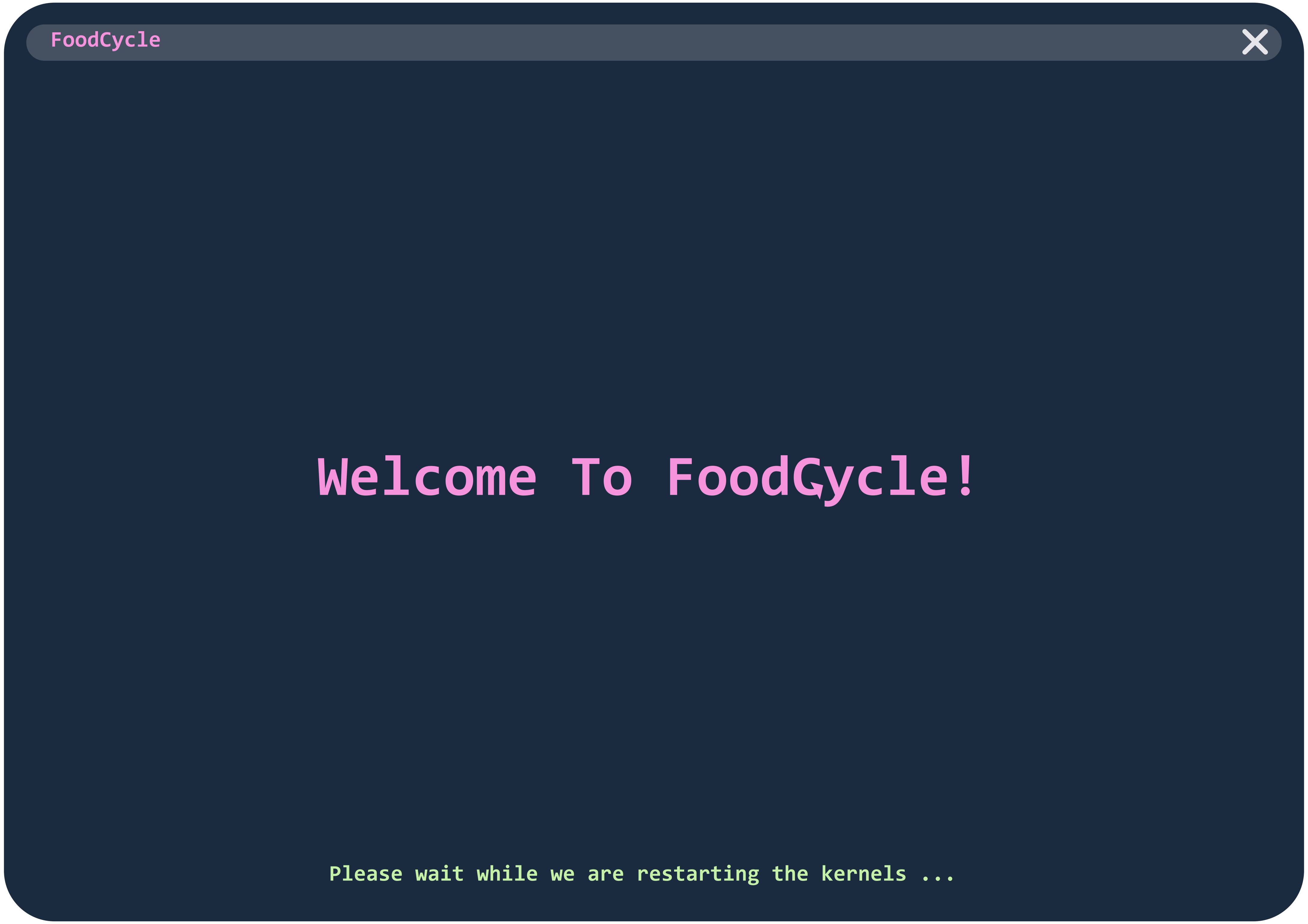
Update occupied dictionary

Occupied Spaces Dictionary

Proceed to Visualisation

5 Assigning The Best Fitting Option

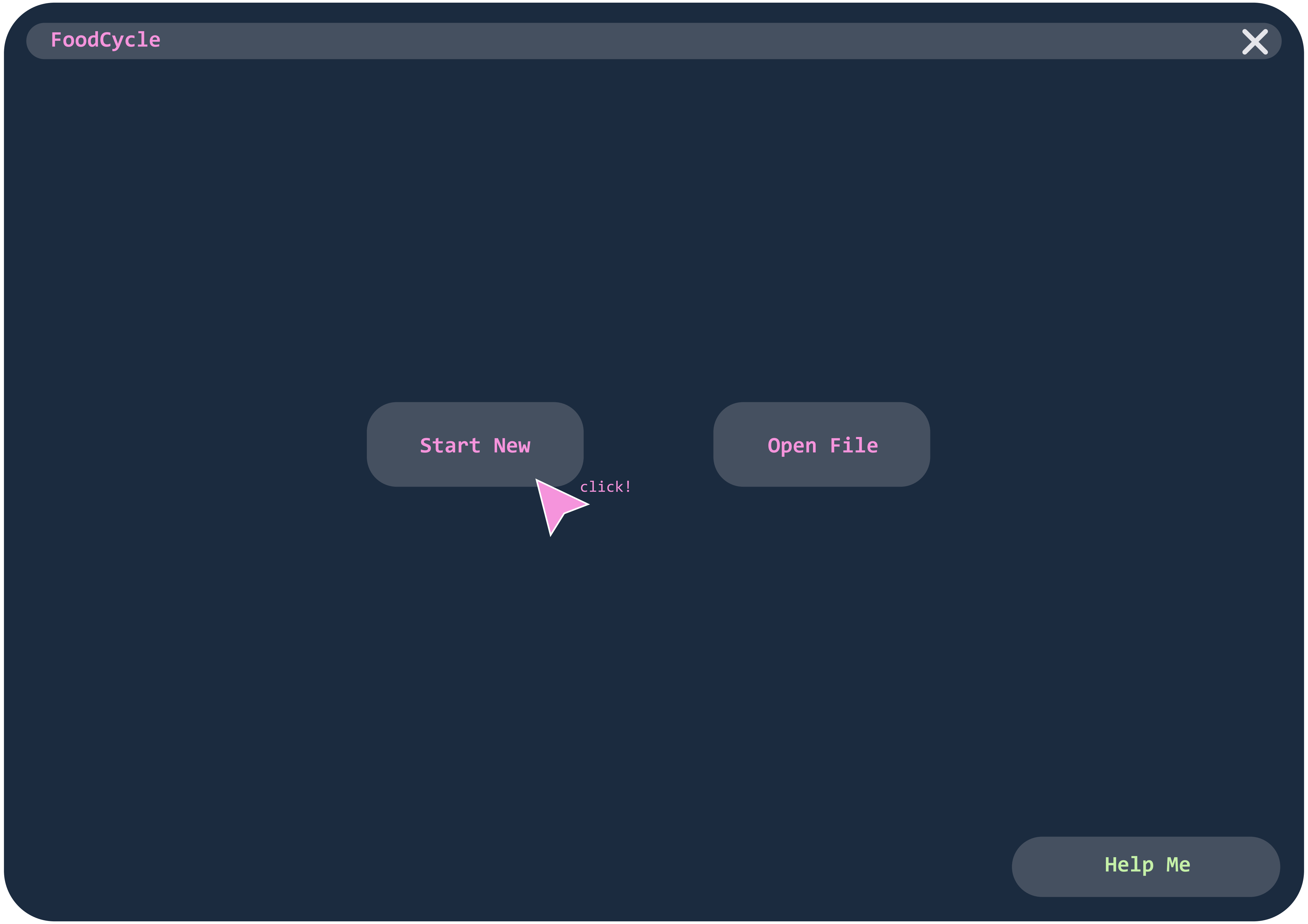
C.8 User Interface

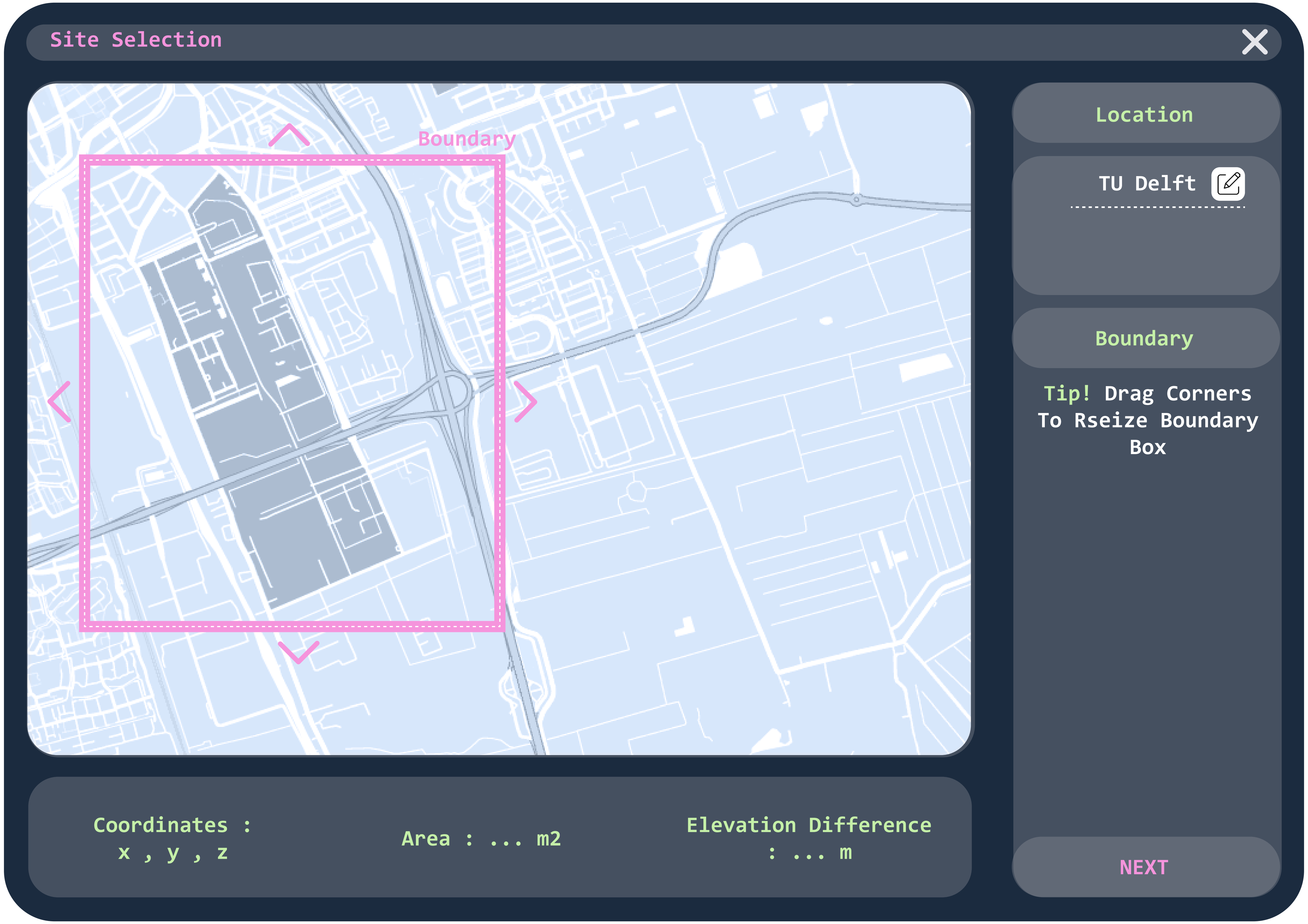


FoodCycle

Welcome To FoodCycle!

Please wait while we are restarting the kernels ...


FoodCycle
Start New
Open File
click!
Help Me



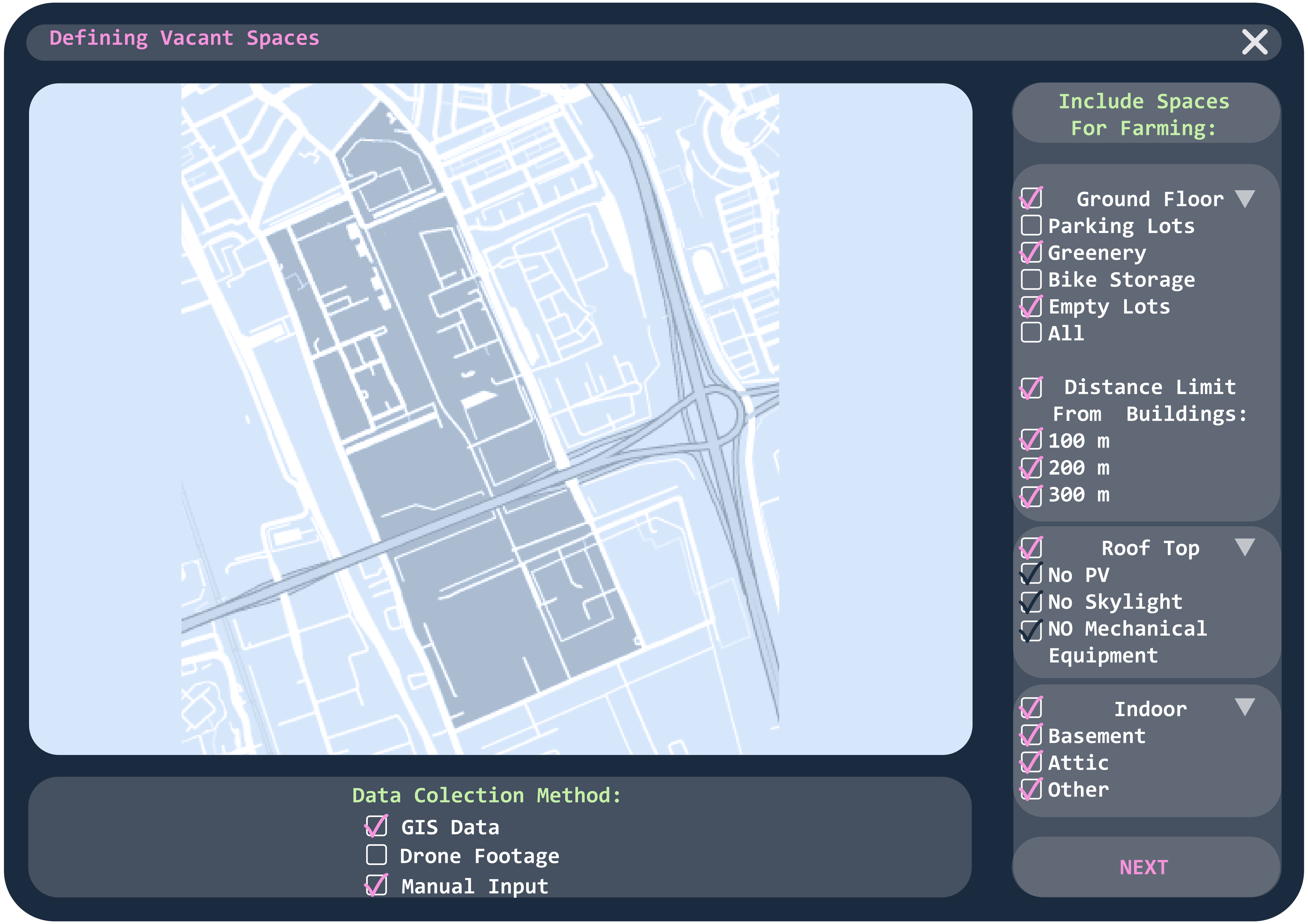
TU Delft

Tip! Drag Corners To Rseize Boundary Box

Coordinates :
x , y , z

Area : ... m2

Elevation Difference
: ... m



Defining Vacant Spaces

Include Spaces For Farming:

☑ Ground Floor ▼
□ Parking Lots
☑ Greenery
□ Bike Storage
☑ Empty Lots
□ All

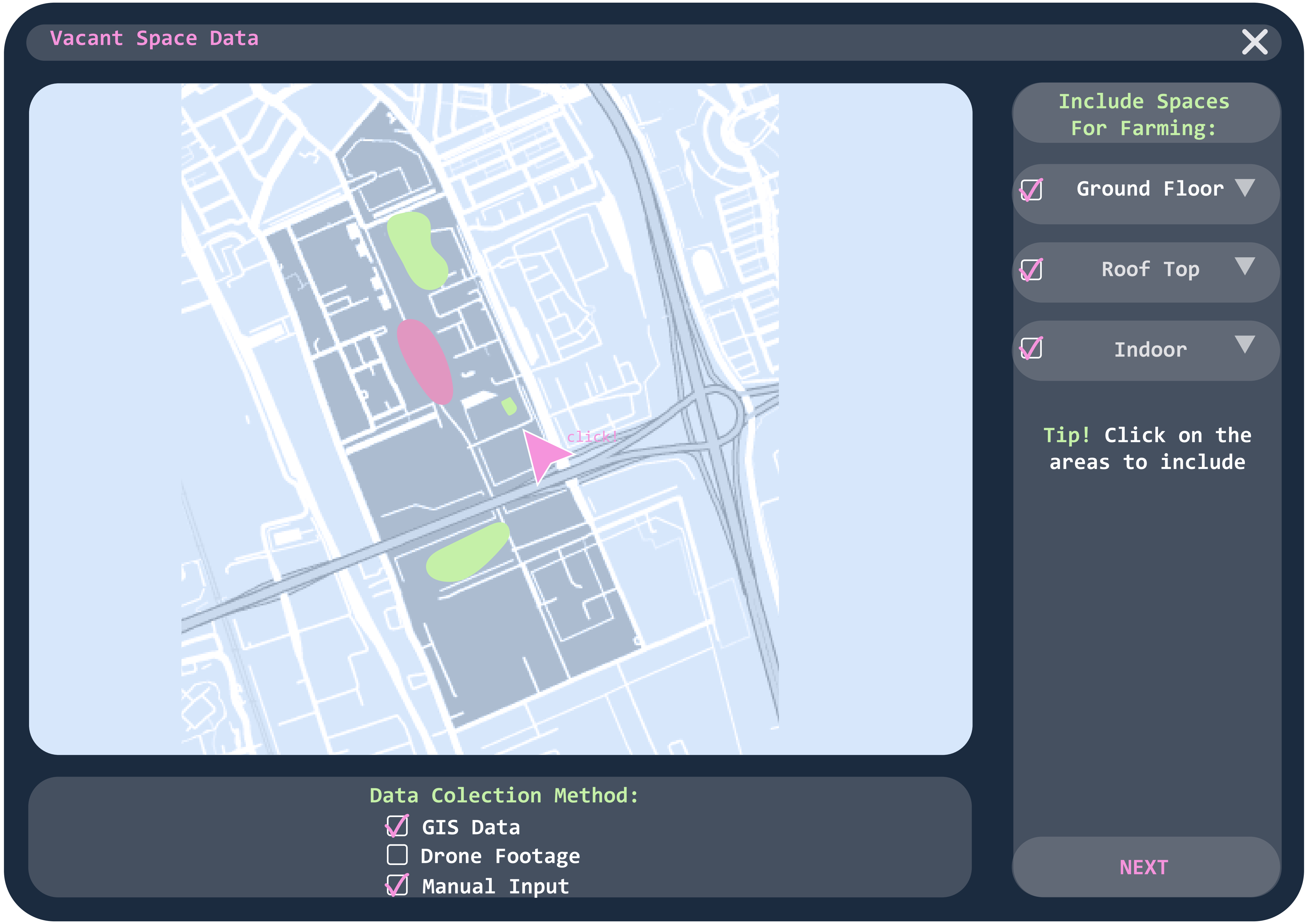
☑ Distance Limit From Buildings:
☑ 100 m
☑ 200 m
☑ 300 m

☑ Roof Top ▼
☑ No PV
☑ No Skylight
☑ NO Mechanical Equipment

☑ Indoor ▼
☑ Basement
☑ Attic
☑ Other

NEXT

Data Colection Method:
☑ GIS Data
□ Drone Footage
☑ Manual Input



Vacant Space Data

Include Spaces For Farming:

☑ Ground Floor ▼

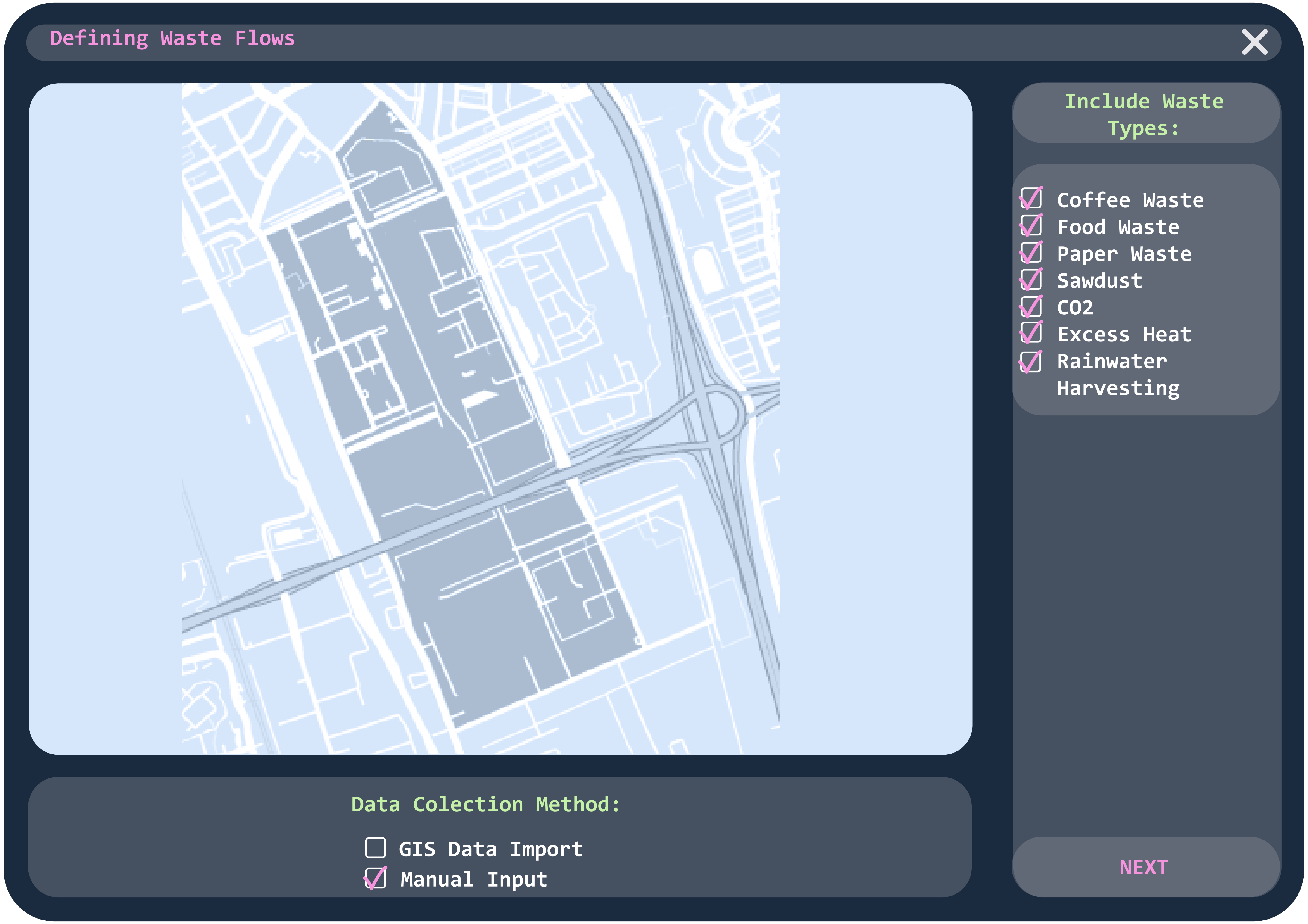
☑ Roof Top ▼

☑ Indoor ▼

Tip! Click on the areas to include

Data Colection Method:

☑ GIS Data
□ Drone Footage
☑ Manual Input



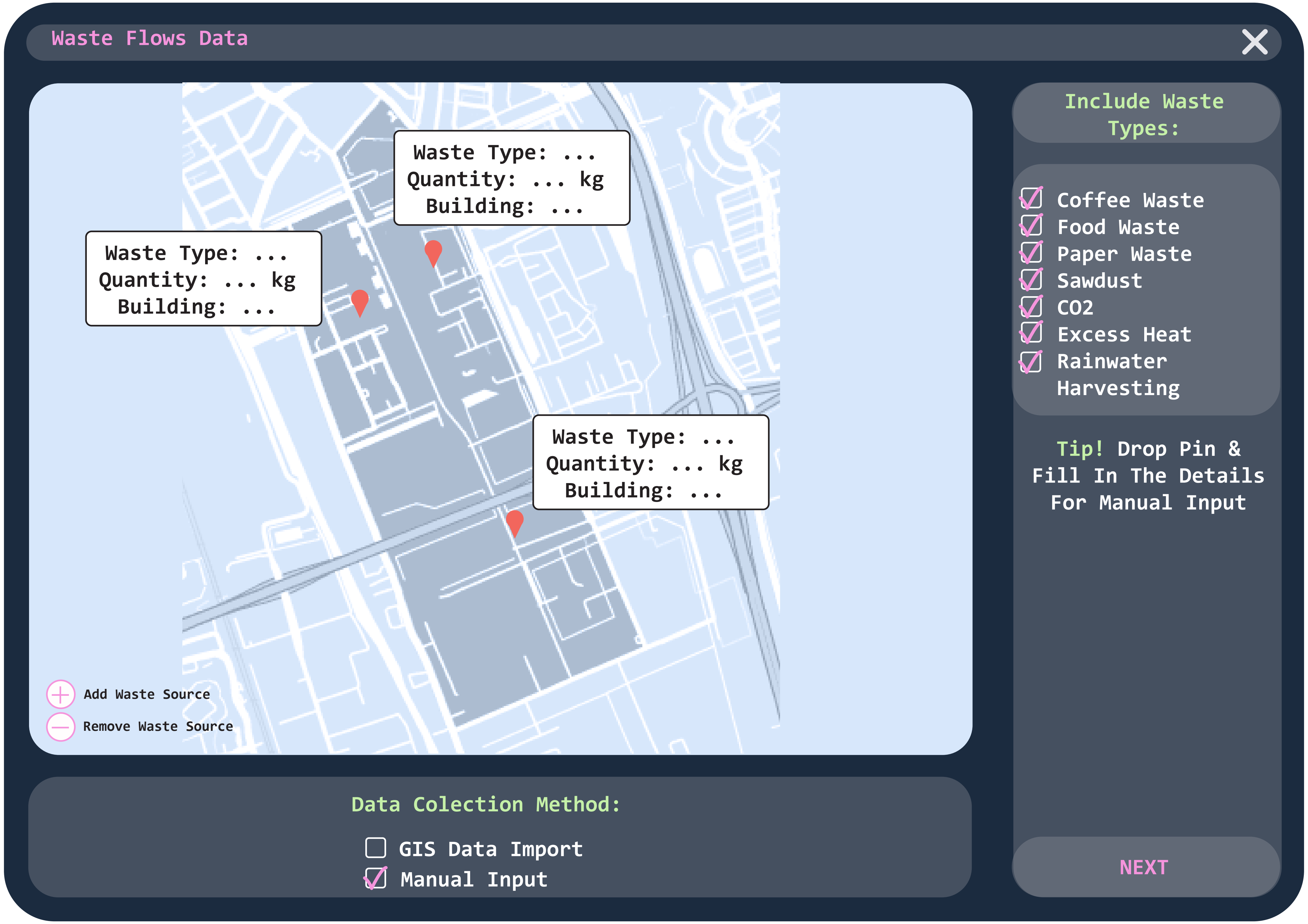
Defining Waste Flows

Include Waste Types:

☑ Coffee Waste
☑ Food Waste
☑ Paper Waste
☑ Sawdust
☑ CO2
☑ Excess Heat
☑ Rainwater Harvesting

Data Colection Method:

□ GIS Data Import
☑ Manual Input



Waste Flows Data

Waste Type: ...
Quantity: ... kg
Building: ...

Waste Type: ...
Quantity: ... kg
Building: ...

Waste Type: ...
Quantity: ... kg
Building: ...

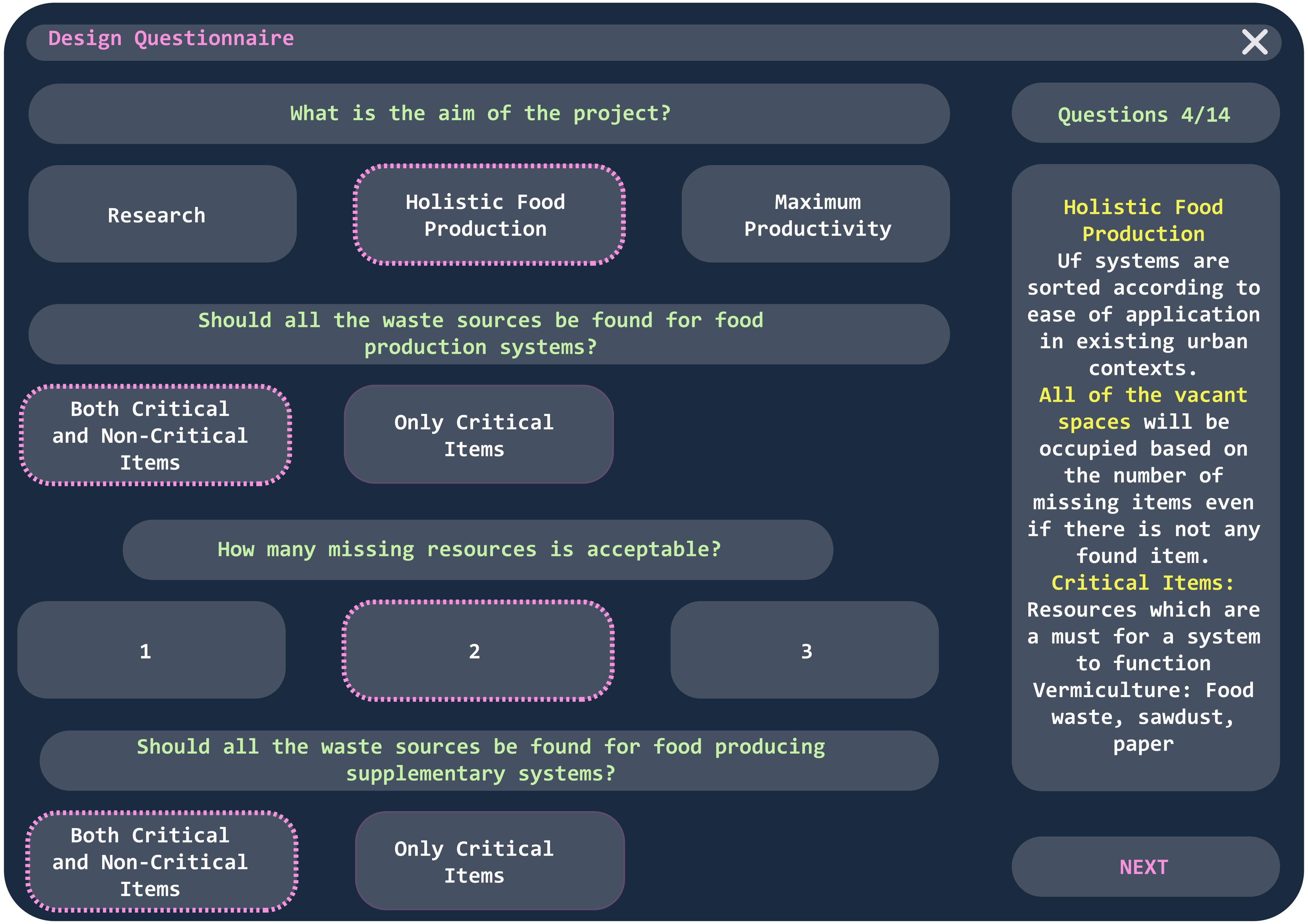
Include Waste Types:

☑ Coffee Waste
☑ Food Waste
☑ Paper Waste
☑ Sawdust
☑ CO2
☑ Excess Heat
☑ Rainwater Harvesting

Tip! Drop Pin & Fill In The Details For Manual Input

Data Colection Method:

□ GIS Data Import
☑ Manual Input



Design Questionnaire

What is the aim of the project?

Research [✓ Holistic Food Production] Maximum Productivity

Should all the waste sources be found for food production systems?

[✓ Both Critical and Non-Critical Items] Only Critical Items

How many missing resources is acceptable?

1 [✓ 2] 3

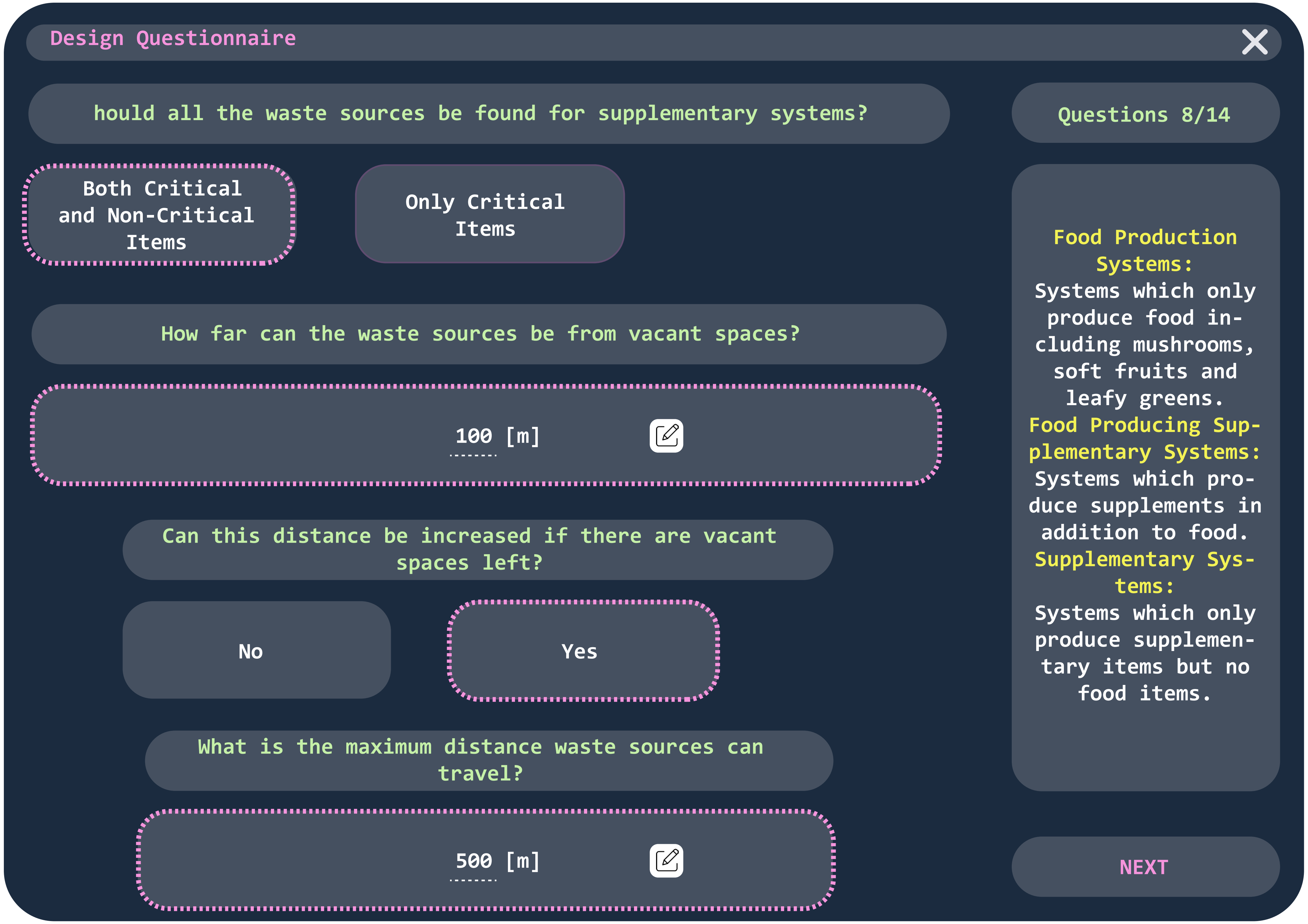
Should all the waste sources be found for food producing supplementary systems?

[✓ Both Critical and Non-Critical Items] Only Critical Items

Questions 4/14

Holistic Food Production
Uf systems are sorted according to ease of application in existing urban contexts.
All of the vacant spaces will be occupied based on the number of missing items even if there is not any found item.
Critical Items: Resources which are a must for a system to function
Vermiculture: Food waste, sawdust, paper

NEXT



Design Questionnaire

hould all the waste sources be found for supplementary systems?

[✓ Both Critical and Non-Critical Items] Only Critical Items

How far can the waste sources be from vacant spaces?

100 [m]

Can this distance be increased if there are vacant spaces left?

No [✓ Yes]

What is the maximum distance waste sources can travel?

500 [m]

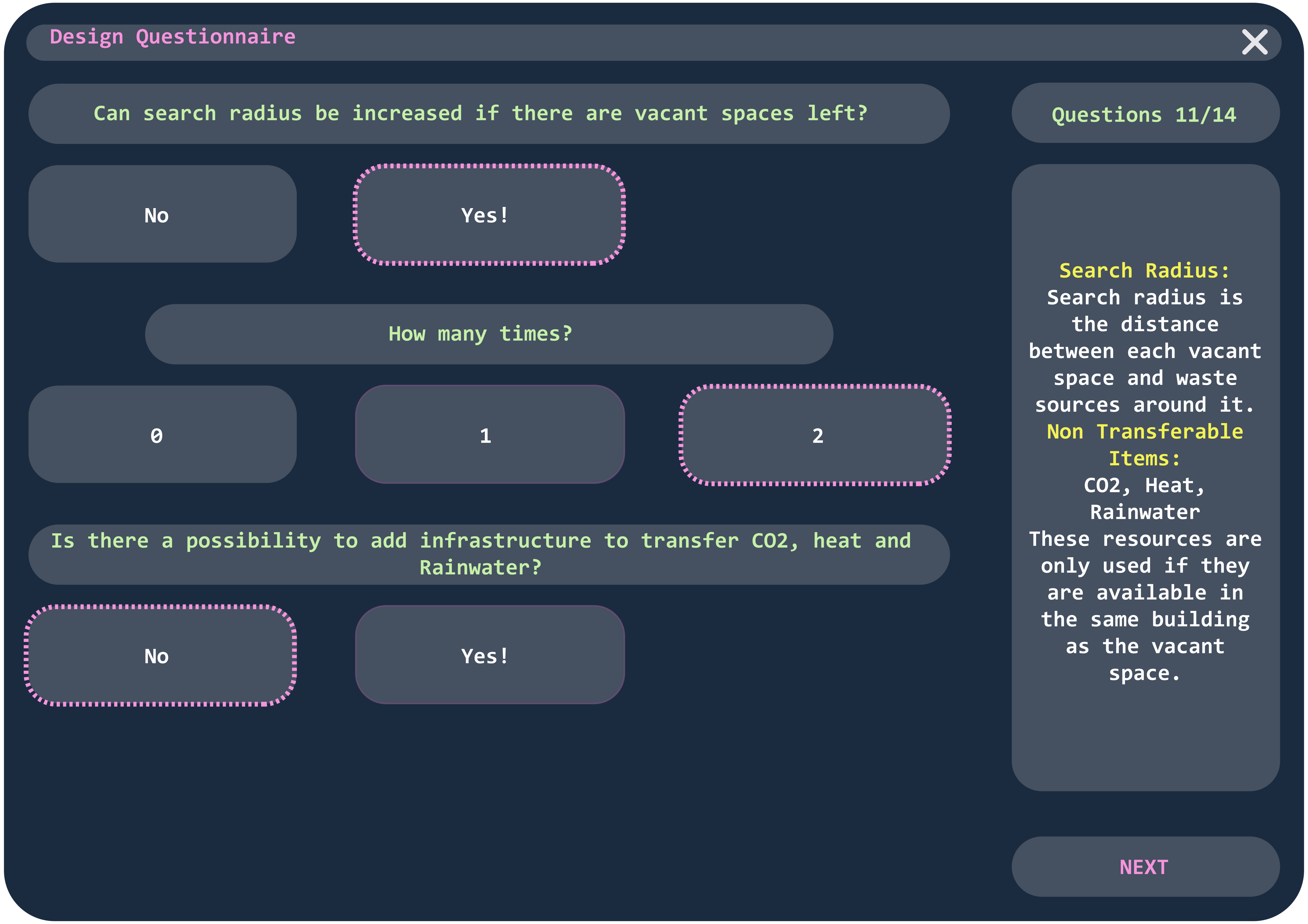
Questions 8/14

Food Production Systems: Systems which only produce food including mushrooms, soft fruits and leafy greens.

Food Producing Supplementary Systems: Systems which produce supplements in addition to food.

Supplementary Systems: Systems which only produce supplementary items but no food items.

NEXT



Design Questionnaire

Can search radius be increased if there are vacant spaces left?

No [✓ Yes!]

How many times?

0 1 [✓ 2]

Is there a possibility to add infrastructure to transfer CO2, heat and Rainwater?

[✓ No] Yes!

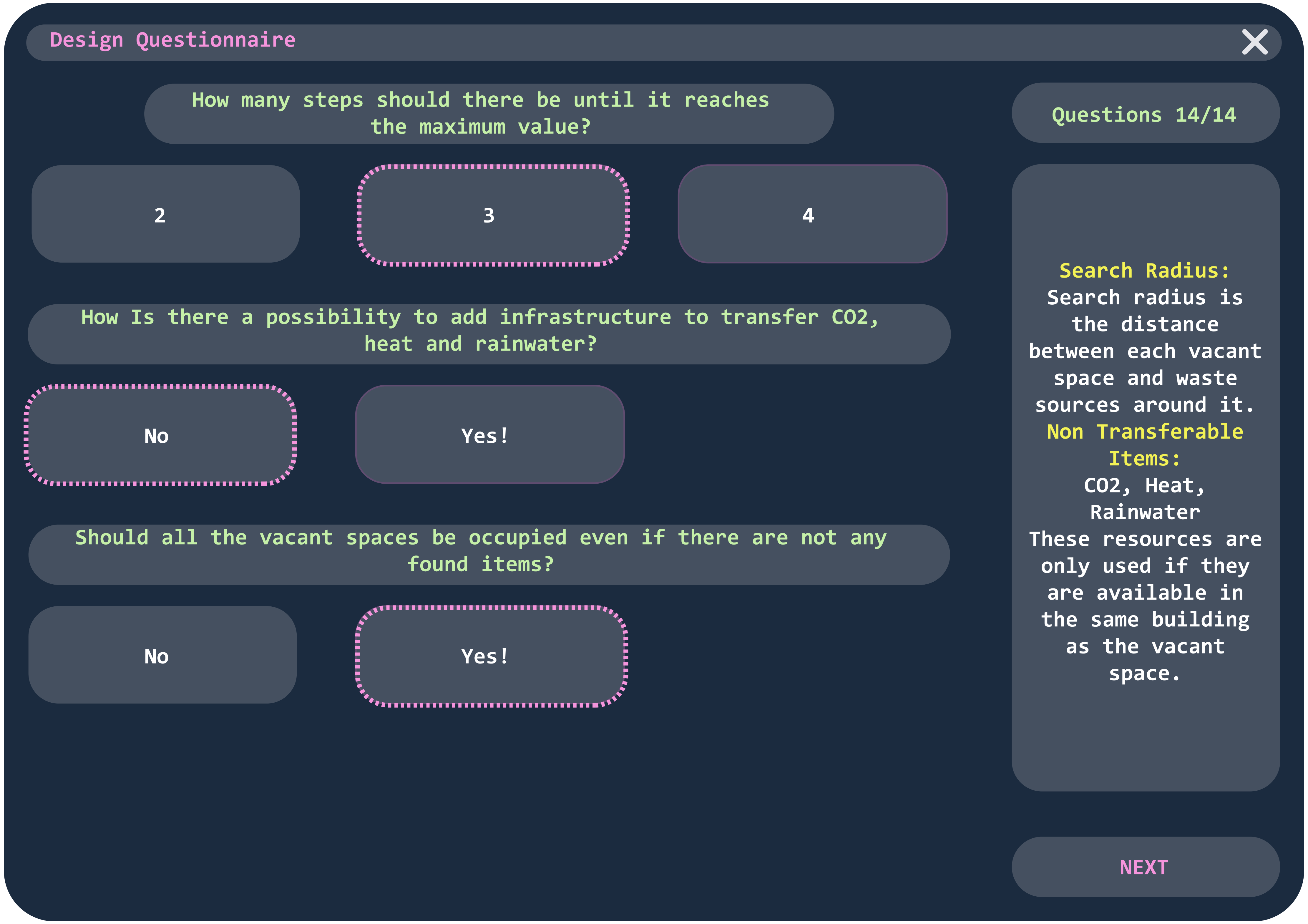
Questions 11/14

Search Radius: Search radius is the distance between each vacant space and waste sources around it.

Non Transferable Items: CO2, Heat, Rainwater

These resources are only used if they are available in the same building as the vacant space.

NEXT



Design Questionnaire

How many steps should there be until it reaches the maximum value?

2 [✓ 3] 4

How Is there a possibility to add infrastructure to transfer CO2, heat and rainwater?

[✓ No] Yes!

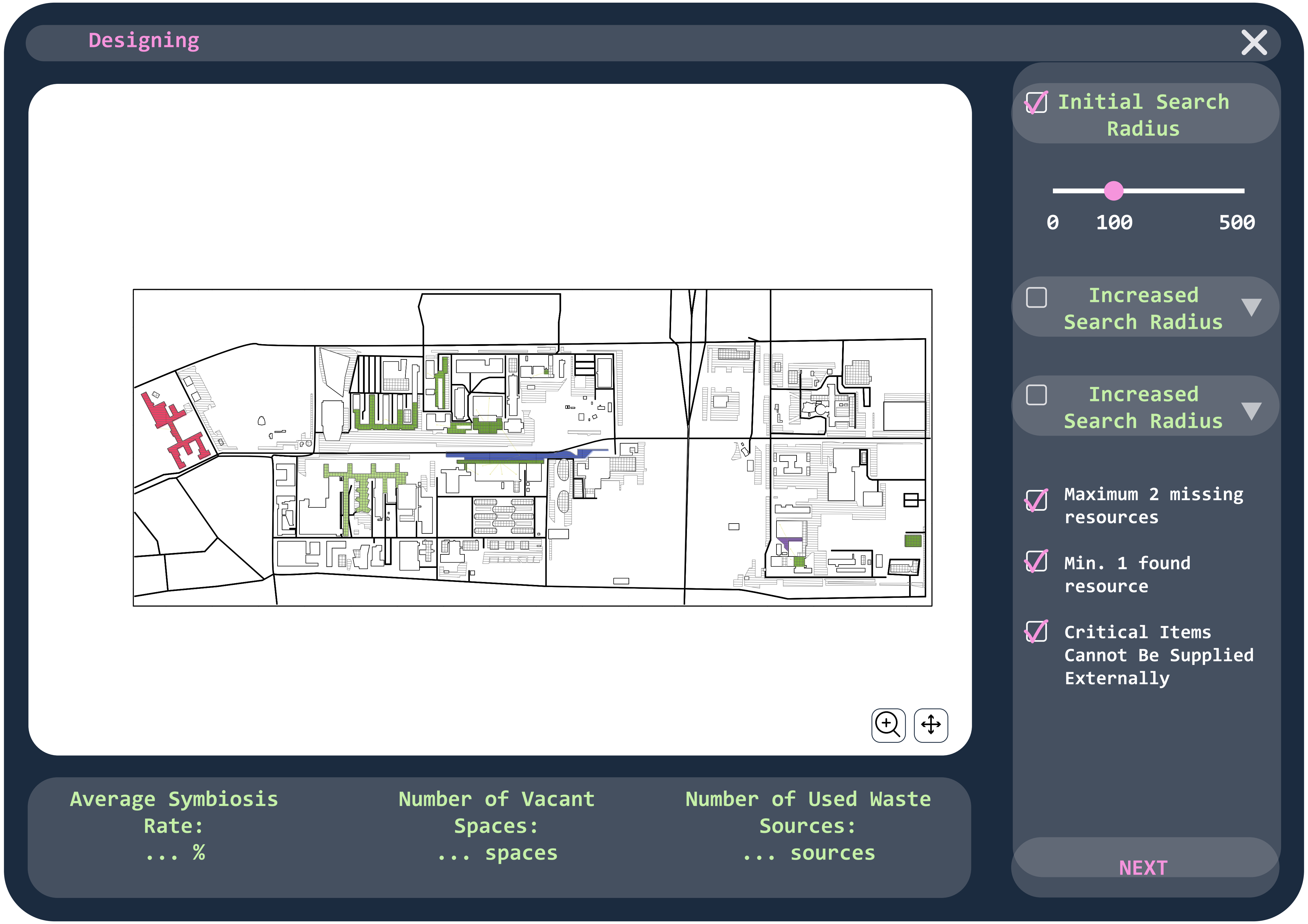
Should all the vacant spaces be occupied even if there are not any found items?

No [✓ Yes!]

Questions 14/14

Search Radius:
Search radius is the distance between each vacant space and waste sources around it.
Non Transferable Items:
CO2, Heat, Rainwater
These resources are only used if they are available in the same building as the vacant space.

NEXT



Designing

☑ Initial Search Radius

0 100 500

□ Increased Search Radius ▼

□ Increased Search Radius ▼

☑ Maximum 2 missing resources

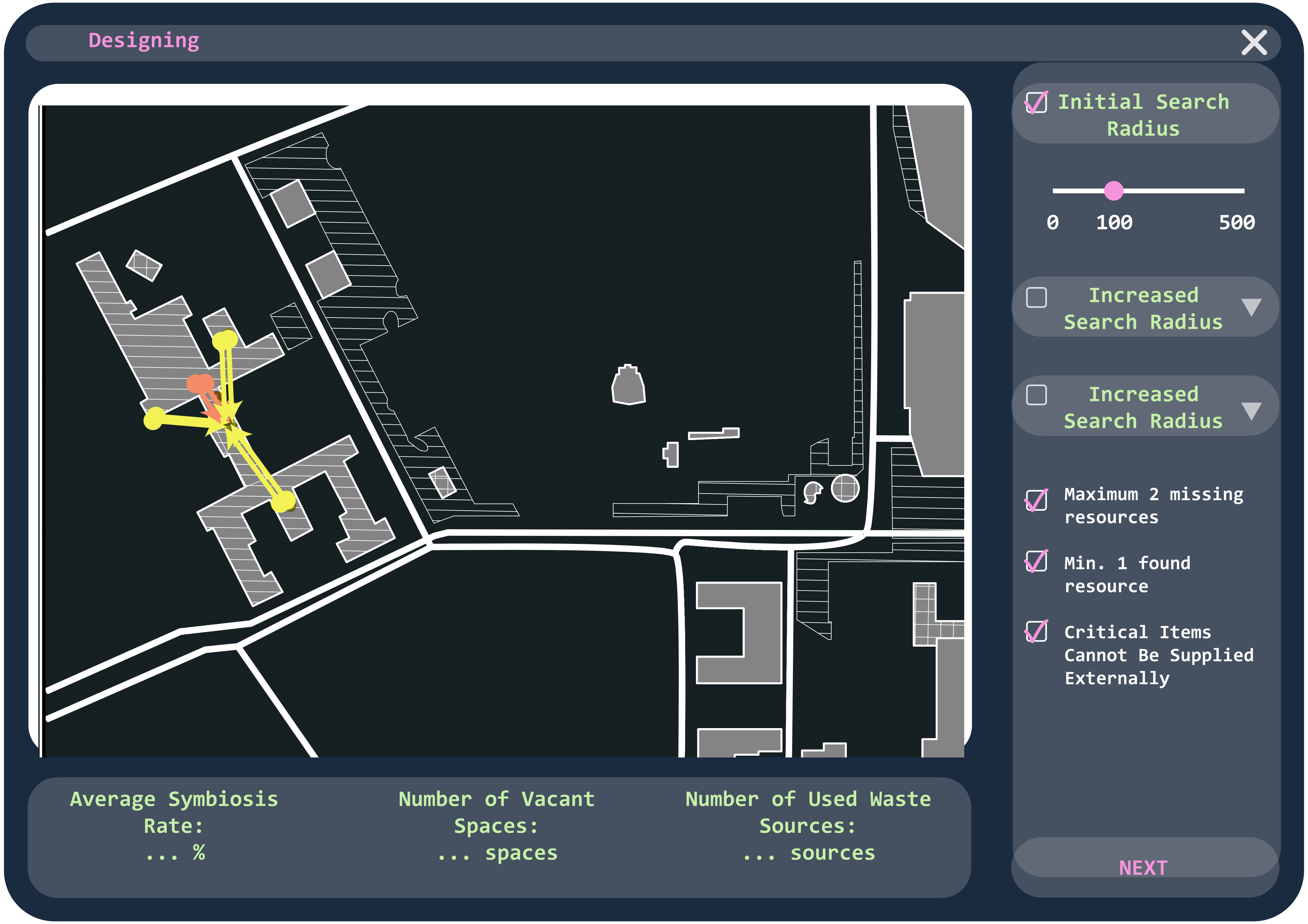
☑ Min. 1 found resource

☑ Critical Items Cannot Be Supplied Externally

Average Symbiosis Rate:
... %

Number of Vacant Spaces:
... spaces

Number of Used Waste Sources:
... sources



☑ Initial Search Radius

☐ Increased Search Radius

☐ Increased Search Radius

☑ Maximum 2 missing resources

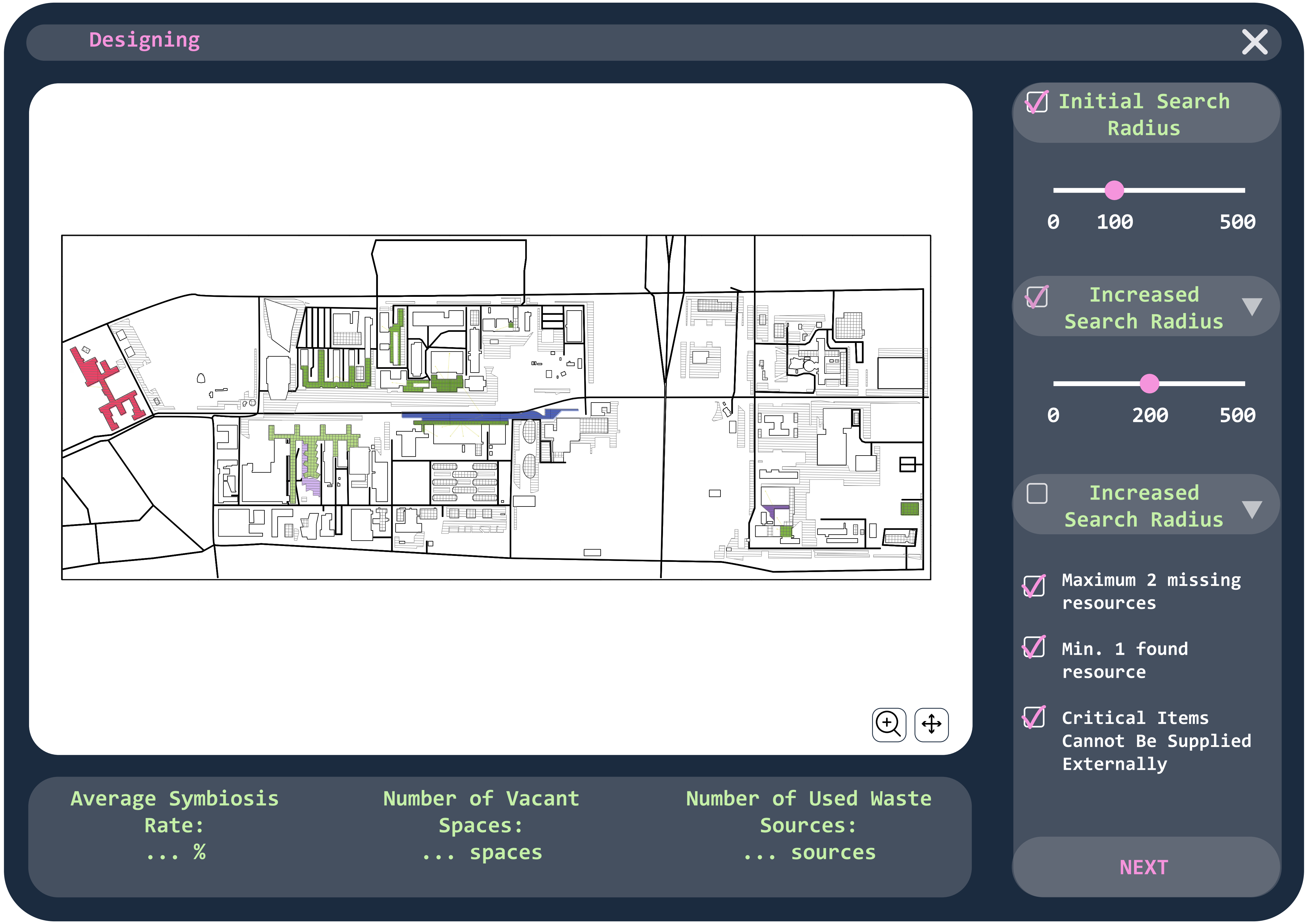
☑ Min. 1 found resource

☑ Critical Items Cannot Be Supplied Externally

Average Symbiosis Rate:
... %

Number of Vacant Spaces:
... spaces

Number of Used Waste Sources:
... sources



☑ Initial Search Radius

0 100 500

☑ Increased Search Radius ▼

0 200 500

□ Increased Search Radius ▼

☑ Maximum 2 missing resources

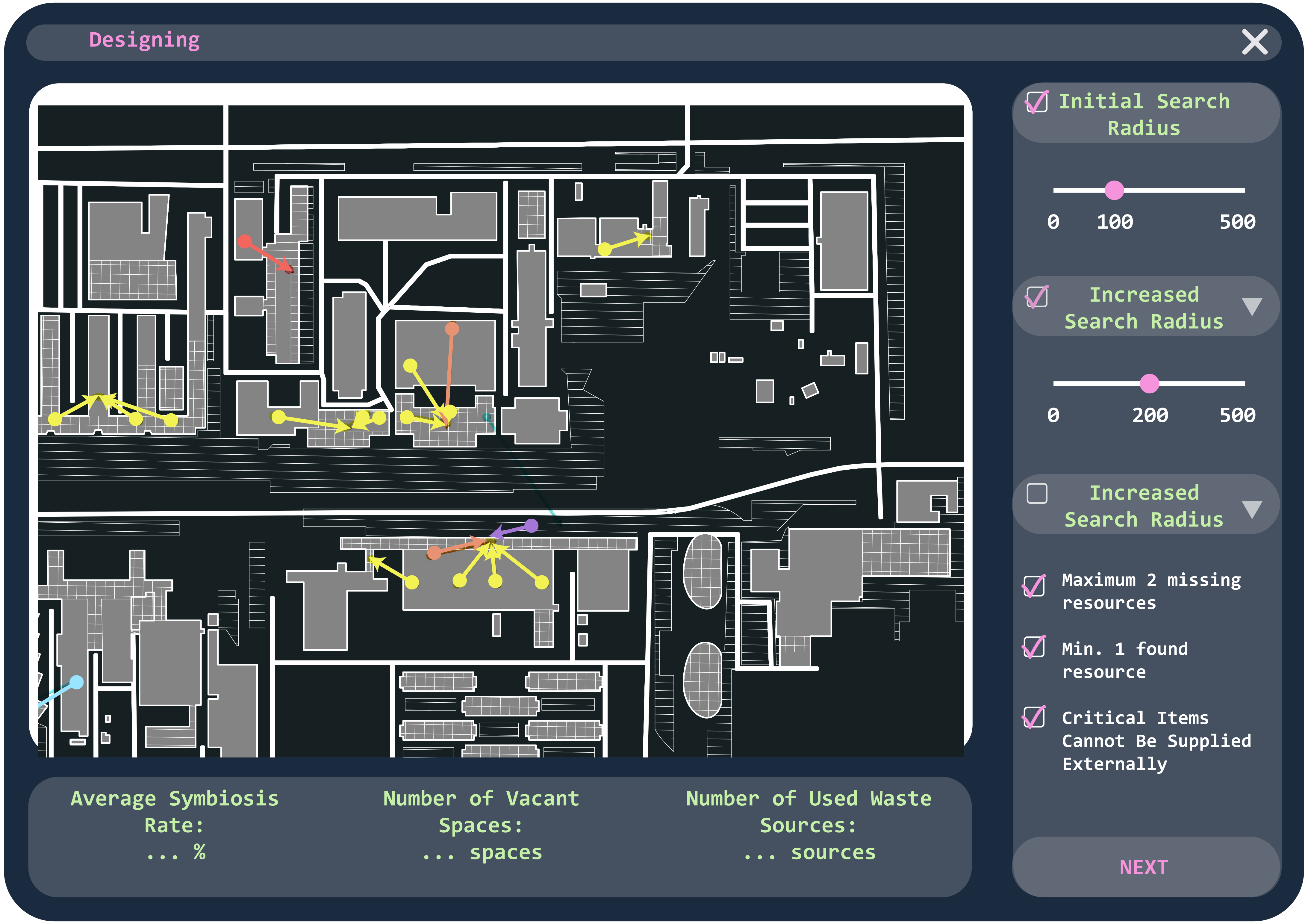
☑ Min. 1 found resource

☑ Critical Items Cannot Be Supplied Externally

Average Symbiosis Rate:
... %

Number of Vacant Spaces:
... spaces

Number of Used Waste Sources:
... sources



Designing

☑ Initial Search Radius

0 100 500

☑ Increased Search Radius

0 200 500

□ Increased Search Radius

☑ Maximum 2 missing resources

☑ Min. 1 found resource

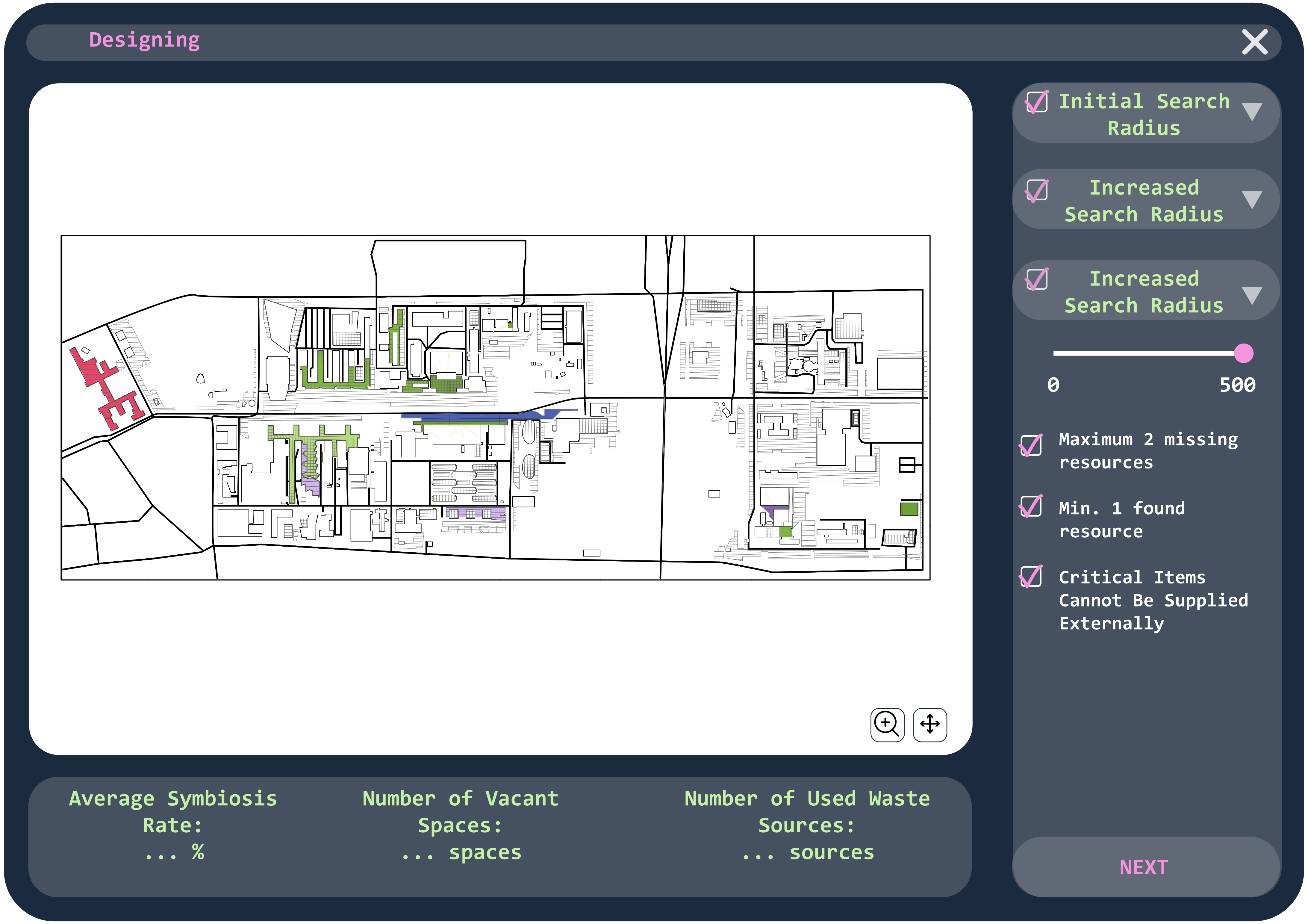
☑ Critical Items Cannot Be Supplied Externally

Average Symbiosis Rate: ... %

Number of Vacant Spaces: ... spaces

Number of Used Waste Sources: ... sources

NEXT



Designing

☑ Initial Search Radius ▼

☑ Increased Search Radius ▼

☑ Increased Search Radius ▼

0 — 500

☑ Maximum 2 missing resources

☑ Min. 1 found resource

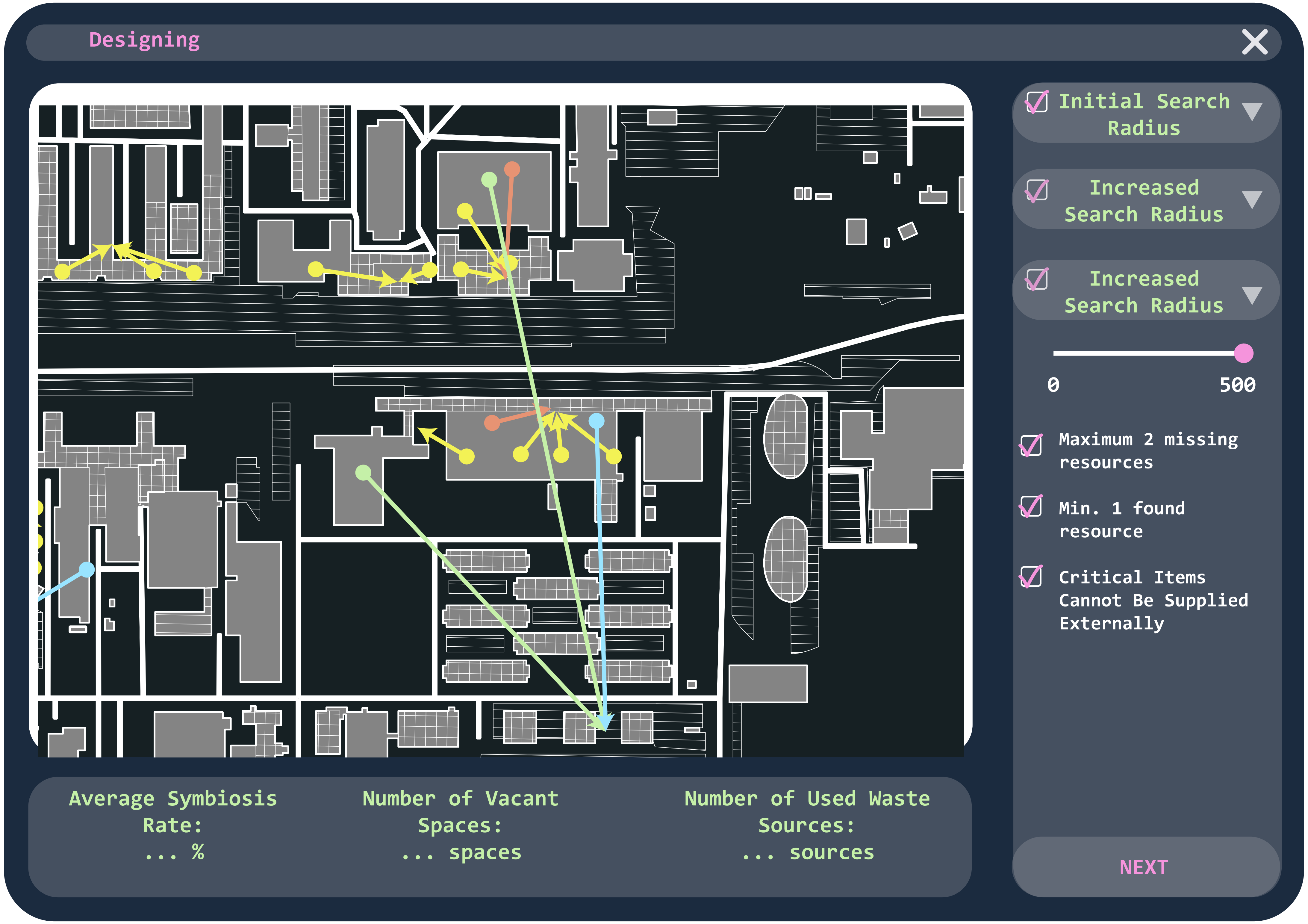
☑ Critical Items Cannot Be Supplied Externally

NEXT

Average Symbiosis Rate:
... %

Number of Vacant Spaces:
... spaces

Number of Used Waste Sources:
... sources



Designing

☑ Initial Search Radius ▼

☑ Increased Search Radius ▼

☑ Increased Search Radius ▼

0 500

☑ Maximum 2 missing resources

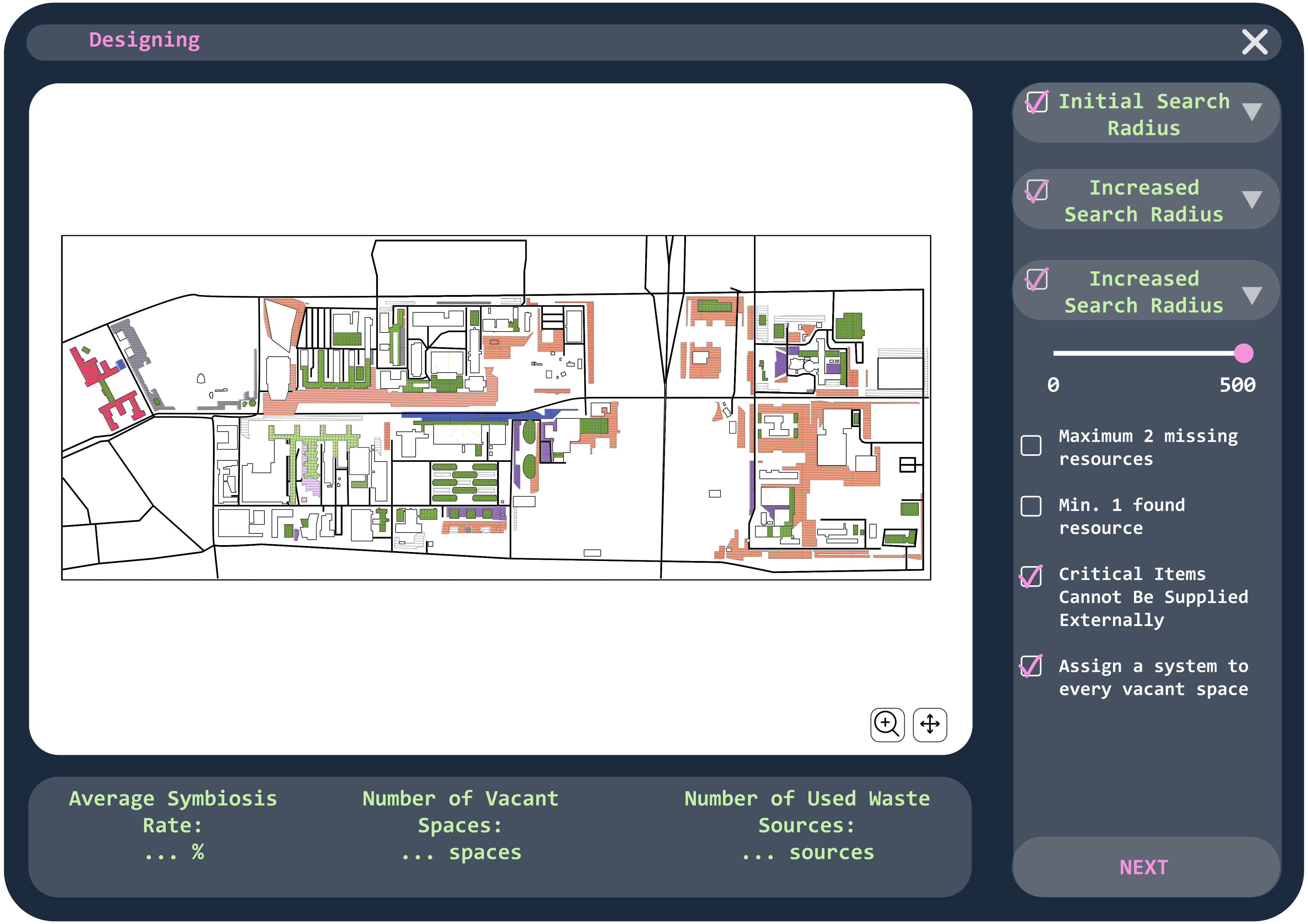
☑ Min. 1 found resource

☑ Critical Items Cannot Be Supplied Externally

Average Symbiosis Rate: ... %

Number of Vacant Spaces: ... spaces

Number of Used Waste Sources: ... sources



☑ Initial Search Radius

☑ Increased Search Radius

☑ Increased Search Radius

□ Maximum 2 missing resources

□ Min. 1 found resource

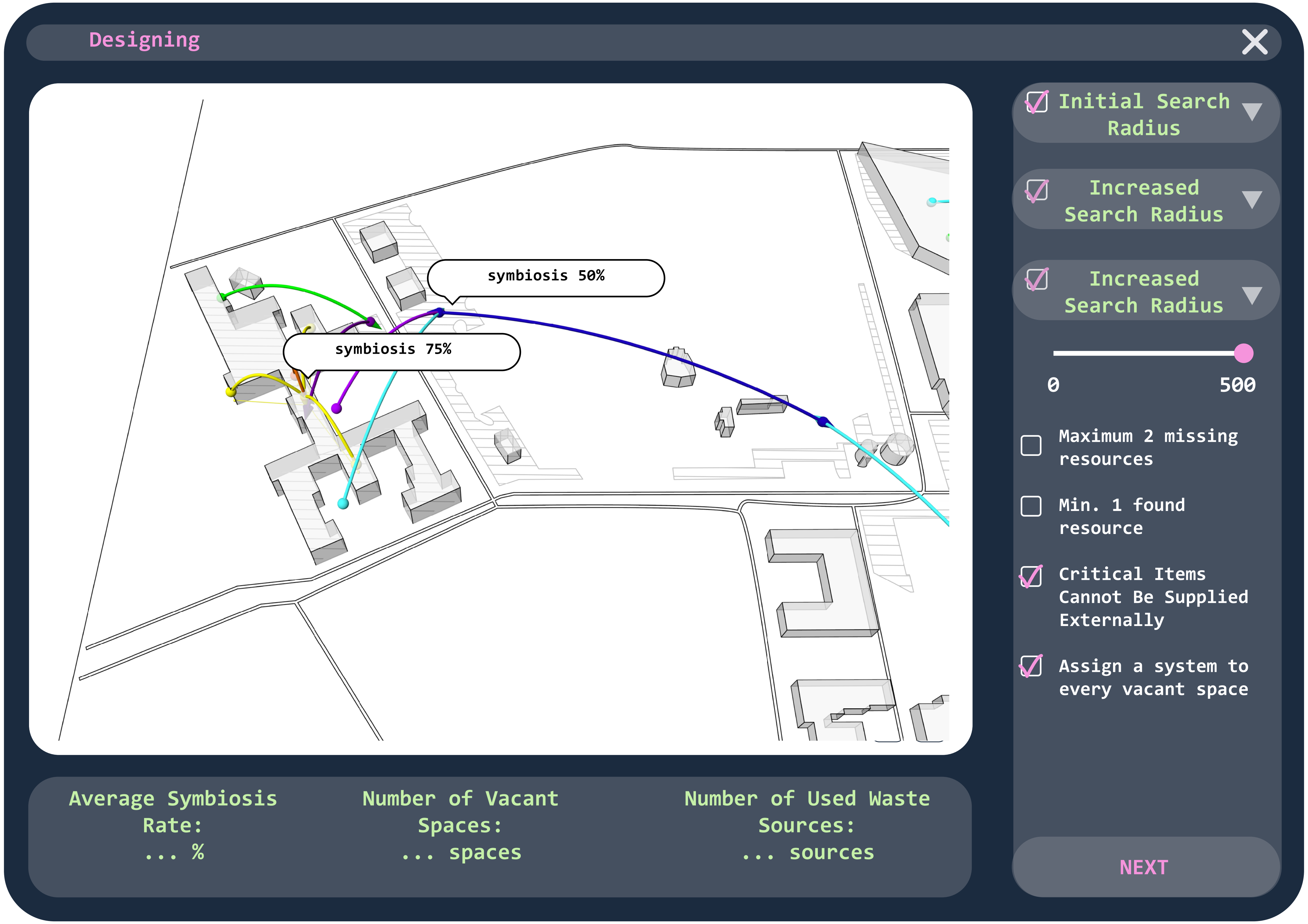
☑ Critical Items Cannot Be Supplied Externally

☑ Assign a system to every vacant space

Average Symbiosis Rate:
... %

Number of Vacant Spaces:
... spaces

Number of Used Waste Sources:
... sources



Designing

☑ Initial Search Radius ▼

☑ Increased Search Radius ▼

☑ Increased Search Radius ▼

0 500

☐ Maximum 2 missing resources

☐ Min. 1 found resource

☑ Critical Items Cannot Be Supplied Externally

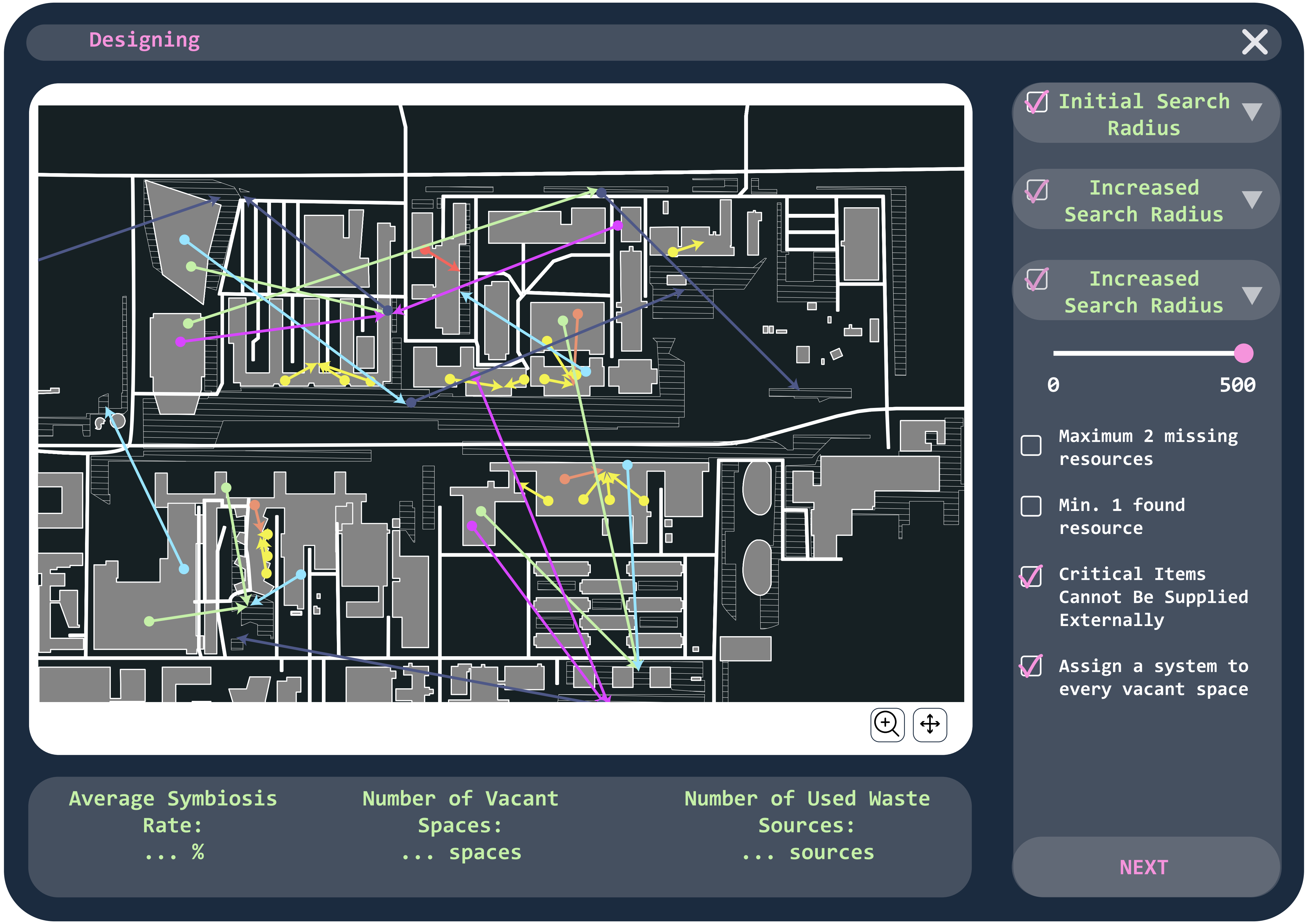
☑ Assign a system to every vacant space

Average Symbiosis Rate:
... %

Number of Vacant Spaces:
... spaces

Number of Used Waste Sources:
... sources

NEXT



☑ Initial Search Radius ▼

☑ Increased Search Radius ▼

☑ Increased Search Radius ▼

0 500

☐ Maximum 2 missing resources

☐ Min. 1 found resource

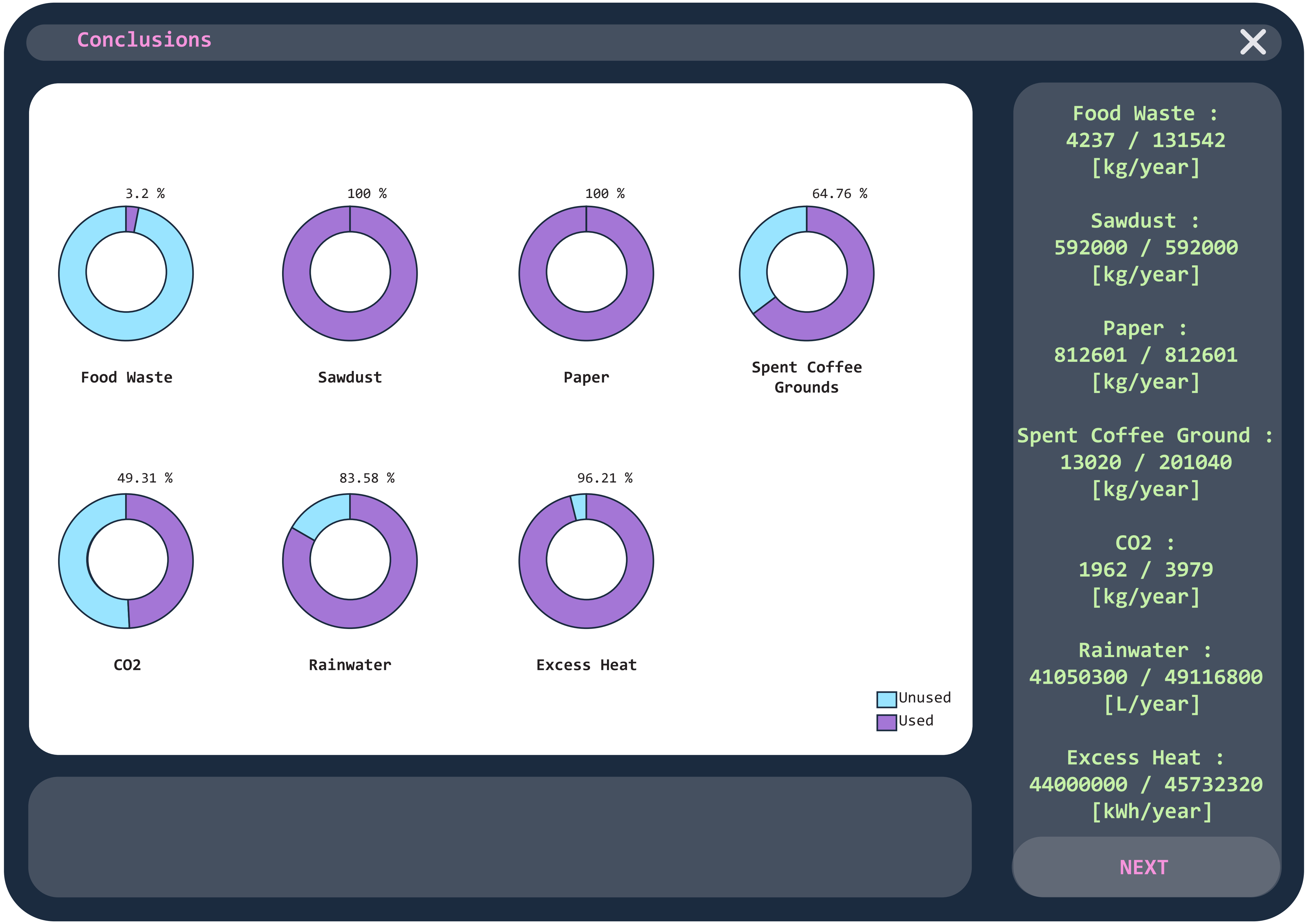
☑ Critical Items Cannot Be Supplied Externally

☑ Assign a system to every vacant space

Average Symbiosis Rate:
... %

Number of Vacant Spaces:
... spaces

Number of Used Waste Sources:
... sources



Food Waste :
4237 / 131542
[kg/year]

Sawdust :
592000 / 592000
[kg/year]

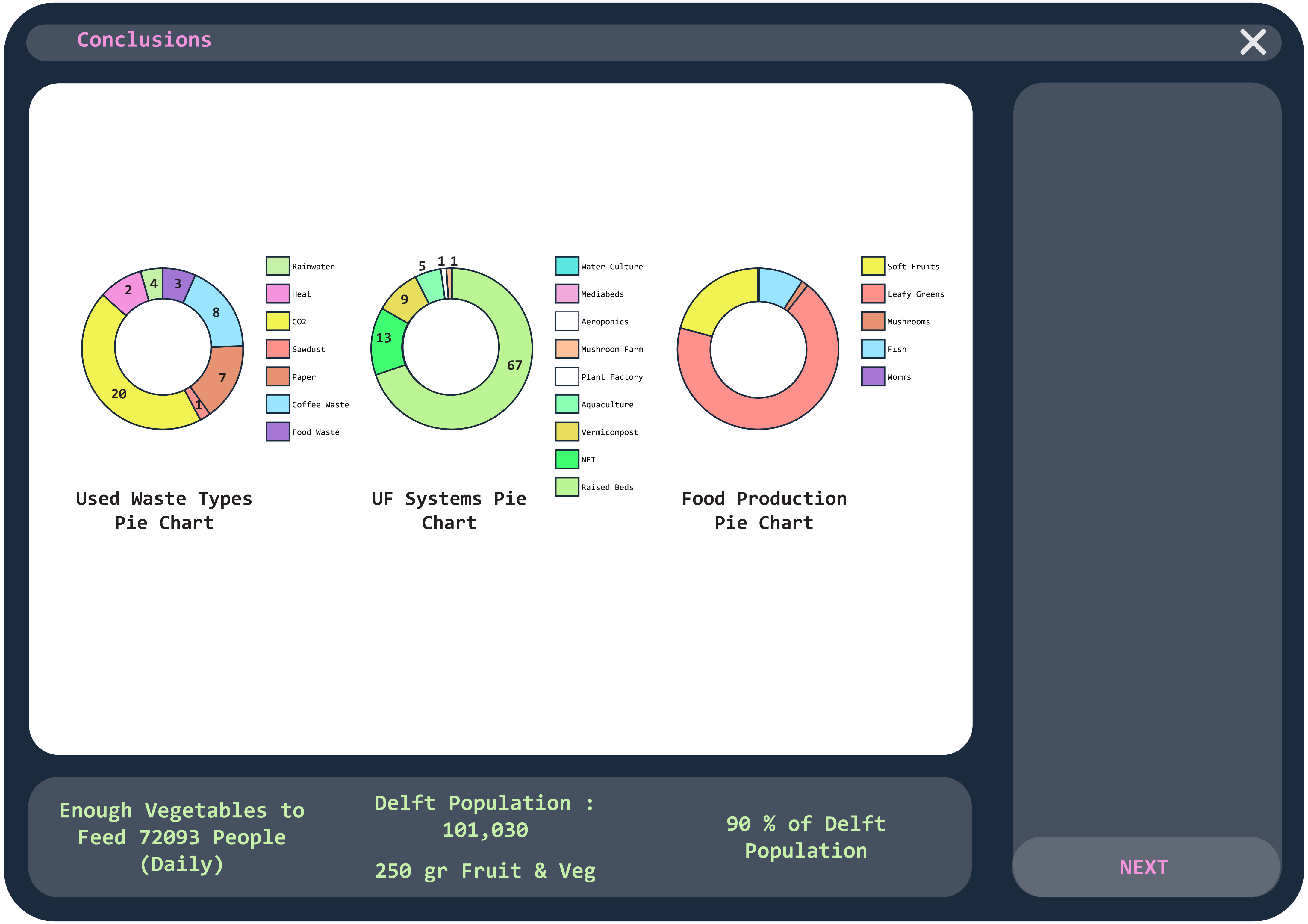
Paper :
812601 / 812601
[kg/year]

Spent Coffee Ground :
13020 / 201040
[kg/year]

CO2 :
1962 / 3979
[kg/year]

Rainwater :
41050300 / 49116800
[L/year]

Excess Heat :
44000000 / 45732320
[kWh/year]

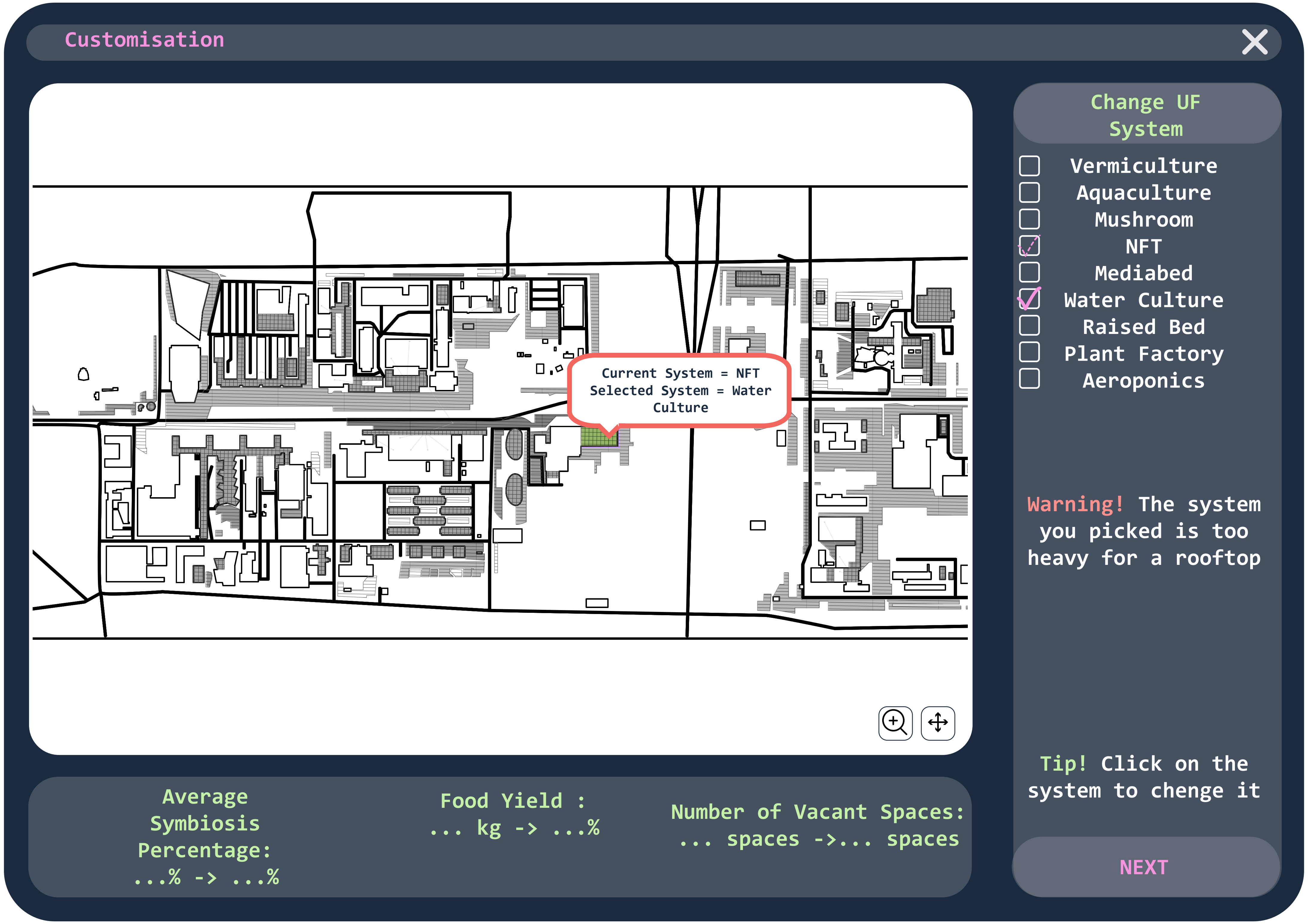


Enough Vegetables to Feed 72093 People (Daily)

Delft Population : 101,030

250 gr Fruit & Veg

90 % of Delft Population



Customisation

Current System = NFT
Selected System = Water Culture

Change UF System

☐ Vermiculture
☐ Aquaculture
☐ Mushroom
☑ NFT
☐ Mediabed
☑ Water Culture
☐ Raised Bed
☐ Plant Factory
☐ Aeroponics

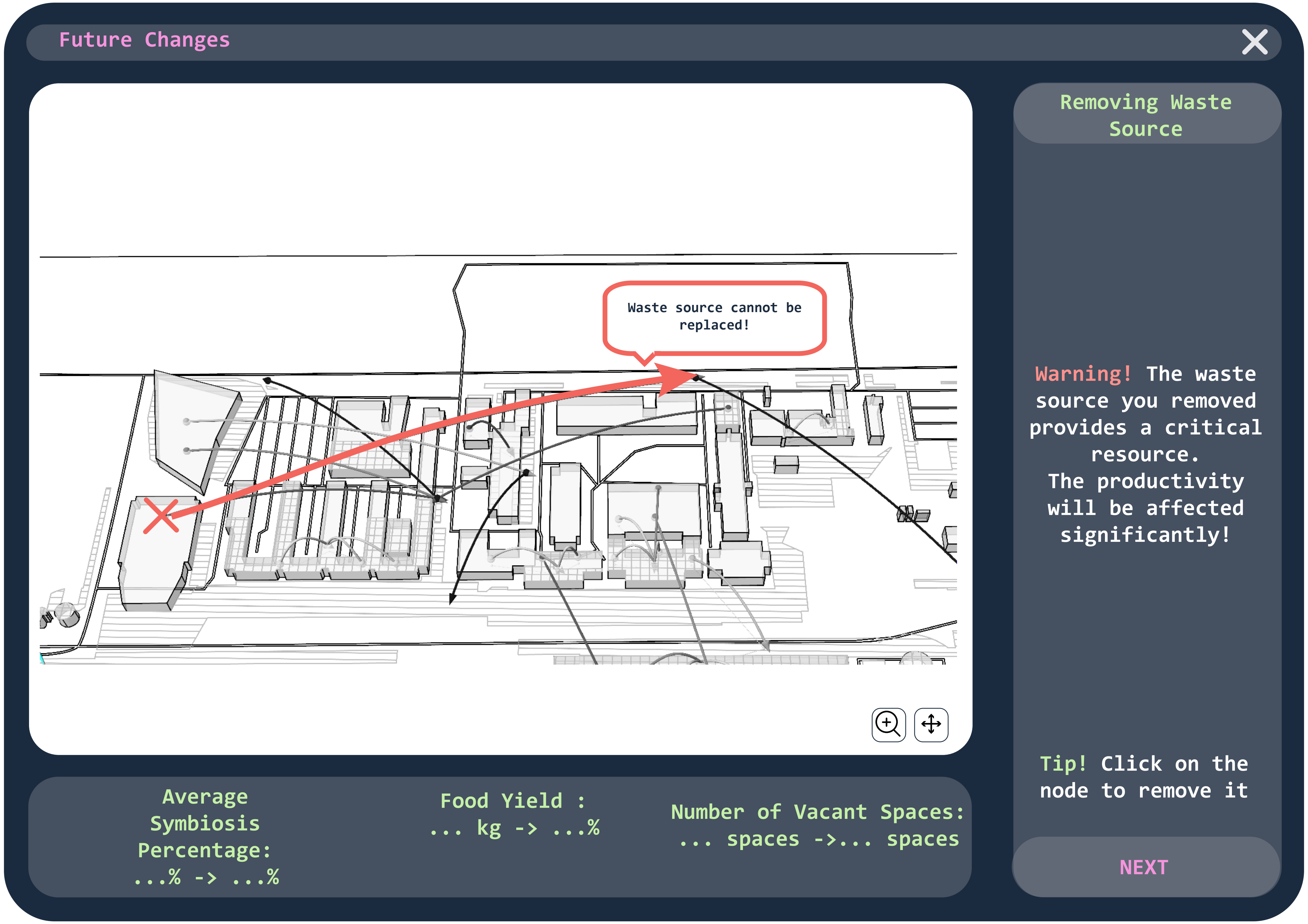
Warning! The system you picked is too heavy for a rooftop

Tip! Click on the system to chenge it

Average Symbiosis Percentage:
...% -> ...%

Food Yield :
... kg -> ...%

Number of Vacant Spaces:
... spaces ->... spaces



Future Changes

Removing Waste Source

Warning! The waste source you removed provides a critical resource.
The productivity will be affected significantly!

Tip! Click on the node to remove it

Average Symbiosis Percentage:
...% -> ...%

Food Yield :
... kg -> ...%

Number of Vacant Spaces:
... spaces ->... spaces

Appendix D Waste Quantities & Yields

APPENDIX D WASTE QUANTITIES & YIELDS

D.1 WASTE DEMAND OF EACH UF SYSTEM (PER M2)

FOOD WASTE (W1)

Worm Weight Per Square Foot (Pound/ Sq Foot)	Worm Weight Per m2 (gr/m2)	Worm Weight Per M2 (kg/m2)	Weight Of 1 Worm (gr)	Area (cm2)	Worm Count
1 pound	4882	4.882	0.25	10000	19528

System	Worm Count	Worm Weight (gr)	Daily Feed (gr)	Total Worm Weight (gr)	Daily Waste (grams)	Annual Waste (kg/m2)	Growth Time (Week)
Vermicompost	19528	0.25	0.125	4882	2441	890.965	7

Worm weight & feed (Fun Facts about Compost Worms, n.d.)
Worm population density (Biernbaum, 2014)

SAWDUST (W2)

System	Area (cm2)	Volume (cm2)	Paper Demand (kg)	use length (Weeks)	Annual Paper Demand (kg)
Mushroom Production	10000	281250	59.0625	9	341.25

Annual Demand = Demand per use x (52/use length)

PAPER WASTE (W3)

System	Area (cm2)	Volume (cm2)	Paper Demand (gr)	use length (Weeks)	Annual Paper Demand (kg)
Vermicompost	10000	(irrelevant)	3125	1	162.5
Mushroom Production	10000	281250	337500	9	1950

Annual Demand = Demand per use x (52/use length)

Vermicompost
example_volume = 40 x 60 x 20 = 48000 cm3 = 0.048 m3
50 pages of newspaper added
newspaper page = 40 - 50 g/m2
newspaper size = 60 x 50 cm
50 pages of newspaper = 0.60 x 0.50 x 50 x 50 = 750 gr

Vermicompost Bin
Sizes = 100 x 100 x 20 = 0.2 m3

direct proportion

in 0.048 m3	750 gr paper	75 gr water
in 0.2 m3	x ?	y ?

x = 3125 gr paper
y = 0.03125 gr water

Mushroom Production:
Substrate + Water = Volume of Bag
x = water
y = substrate
cultivation bag : 100 x 100 x 45 cm
volume of Bag = 100 x 100 x 45 = 450000
sawdust density = 0.21
paper density = 1.2

x+y = 450000
x/y=60/100

x = 168750 cm3 water
y = substrate = 281250 cm3

density = weight/volume

sawdust:
0.21 = sawdust weight / 281250
sawdust weight = 59.0625 kg
paper:
1.2 = paper weight / 281250
paper weight = 337.5 kg

Vermicompost example (Fong & Hewitt, n.d.)
substrate to water ratio (Sayner, 2021)
grow bag dimensions (Shields, 2017)
saw dust density (Density of Sawdust, n.d.)

COFFEE WASTE (W4)

System	Area Of Farm (m2)	Annual Mushroom(kg/m2)	Substrate - Yield Ratio	Annual SCG (kg/m2)
Mushroom Productıon	10000	125	1.14	142

(Annual Yield)x (Ratio) = (Annual SCG Demand)

substrate - yield ratio based on Table 2 (Martínez-Carrera et al., 2000)

CO2 (W5)

System	Area (cm2)	Volume (Cm3)	ppm	CO2 Demand (M3)	Air Exchange Rate (1/h)	(kg/h)	Time (h)	CO2 Demand (kg/year)
Plant Factory	10000	4000000	1000	0.004	0.03333333333	0.2208	18	1450.656
Aeroponics	10000	4000000	1000	0.004	0.03333333333	0.2208	18	1450.656
NFT	10000	4000000	1000	0.004	0.03333333333	0.2208	18	1450.656
Water Culture	10000	4000000	1000	0.004	0.03333333333	0.2208	18	1450.656
Media Bed	10000	4000000	1000	0.004	0.03333333333	0.2208	18	1450.656

(Room Volume) x (ppm/1000000) = (CO2 Volume)
(CO2 Volume) x (Density) / (Air Exchange Rate) = (CO2 Demand) kg/h
(CO2 Demand per hour) x time x 365 = Annual CO2 Demand

CO2 Demand Calculation (*The Effects of Too Much CO2 In a Grow Room*, 2021)
CO2 Demand Calculation (CO2Meter, 2015)
Air Exchange Rate (NSW Government, 2021)

RAINWATER (W6)

System	Area (cm2)	Depth (Cm)	Volume (Cm3)	Water Demand (L)	Notes	Use Period (Weeks)	Annual Water Demand(L)
Plant Factory	10000			600	per year	52 weeks	600
Raised Bed	10000	2.5	25000	25	per week	52	2600
Aeroponics	10000			50.26548246	based on 20% less water use compared to other hydroponics	2	1306.902544
NFT	10000	-	62831.85307	62.83185307	circulated water	2	1633.62818
Water Culture	10000	30	300000	300	circulated water	2	7800
Media Bed	10000	30	300000	120	circulated water	2	3120
Vermicompost	10000	20	200000	0.03125		2	0.8125
Mushrooms	10000	45	450000	168.75	60% hydration	16	548.4375

NFT Water Demand
pipe length = 1 m x 8 rows = 8 m
pipe radius = 5 cm
Volume = (pi) (5) x (5) x (800)

Media Bed Water Demand
volume(m3) / 1000 = volume (L)
40% water
water volume = (volume/1000)*0.4

Water Culture Water Demand
container volume = water demand

Aeroponics
20 % less water use compared to hydroponic systems
NFT WATER DEMAND x 0.8 = water demand

Plant Factory Water Demand (Graamans et al., 2018)
Raised Bed Water Demand (de Peyster, 2014)
Aeroponics Water Demand Compared To Other Systems (Rassmann, 2015)
* evaporation and transpiration are not taken into consideration in these calculations

EXCESS HEAT (W7)

System	Area (cm2)	Heat Demand(mj)
Plant Factory	10000	1000

Plant Factory Heat Demand (Graamans et al., 2018)

D.2 SIMPLIFICATION OF VACANT SPACE SIZES & WASTE QUANTITIES

RANGES FOR VACANT SPACE SIZE

Vacant Space Size	Range 1 (m)	Range 2 (m)	Range 3 (m)
Min (m2)	0	200	1000
Max (m2)	200	1000	...

Waste Demand Ranges

for each waste type and each system:
Minimum Waste Demand = (waste demand per m2) x (range min)
Maximum Waste Demand = (waste demand per m2) x (range max)

Food Waste (W1)	kg/m2	Range 1 (kg) (0 - 200 m^2)		Range 2 (kg) (200 - 1000 m^2)		Range 3 (kg) (1000 m^2- above)	
Vermicompost	918.5225	0	183704.5	183704.5	918522.5	918522.5	above

Sawdust (W2)	kg/m2	Range 1 (kg) (0 - 200 m^2)		Range 2 (kg) (200 - 1000 m^2)		Range 3 (kg) (1000 m^2- above)	
Mushroom Production	59.0625	0	11812.5	11812.5	59062.5	59062.5	above

Paper (W3)	kg/m2	Range 1 (kg) (0 - 200 m^2)		Range 2 (kg) (200 - 1000 m^2)		Range 3 (kg) (1000 m^2- above)	
Vermicompost	162.5	0	32500	32500	162500	162500	above
Mushroom Production	337.5	67500	67500	67500	337500	337500	above

Spent Coffee Grounds (W4)	KG/m2	Range 1 (kg) (0 - 200 m^2)		Range 2 (kg) (200 - 1000 m^2)		Range 3 (kg) (1000 m^2- above)	
Mushroom	141	0	28200	28200	141000	141000	above

CO2 (W5)	kg/m2	Range 1 (kg) (0 - 200 m^2)		Range 2 (kg) (200 - 1000 m^2)		Range 3 (kg) (1000 m^2- above)	
Plant Factory	10	0	2000	2000	10000	10000	above
Aeroponics	96.7104	0	19342.08	19342.08	96710.4	96710.4	above
NFT	96.7104	0	19342.08	19342.08	96710.4	96710.4	above
Water Culture	96.7104	0	19342.08	19342.08	96710.4	96710.4	above
Media Bed	96.7104	0	19342.08	19342.08	96710.4	96710.4	above

Rainwater (W6)	L/m2	Range 1 (kg) (0 - 200 m^2)		Range 2 (kg) (200 - 1000 m^2)		Range 3 (kg) (1000 m^2- above)	
Plant Factory	600	0	120000	120000	600000	600000	above
Raised Bed	2600	0	520000	520000	2600000	2600000	above
Aeroponics	1306.902544	0	261380.5088	261380.5088	1306902.544	1306902.544	above
NFT	1633.62818	0	326725.636	326725.636	1633628.18	1633628.18	above
Water Culture	7800	0	1560000	1560000	7800000	7800000	above
Media Bed	3120	0	624000	624000	3120000	3120000	above
Vermicompost	0.8125	0	162.5	162.5	812.5	812.5	above
Mushrooms	548.4375	0	109687.5	109687.5	548437.5	548437.5	above

Residual Heat (W7)	MJ/m2	Range 1 (kg) (0 - 200 m^2)		Range 2 (kg) (200 - 1000 m^2)		Range 3 (kg) (1000 m^2- above)	
Plant Factory	1000	0	200000	200000	1000000	1000000	above

D.3 YIELD OF EACH UF SYSTEM (PER M2)

LEAFY GREENS & SOFT FRUITS

Different Systems		Crop Characteristics							System Sizes					Yield (Plants)		Yield(Fruits)		
System Name	**Horizontal/ Vertical**	**Crop Type**	**Horizontal Distance**	**Vertical Distance**	**Equipment Size**	**Number of Fruits**	**Growth Time**	**Crop weight(gr)**	**Area (cm2)**	**Height (cm)**	**Horizontal**	**Vertical**	**#Stacked Racks**	**1 Level**	**Stacked**	**Yield(Fruits)**	**Fruit Yield Annual**	**Annual Fruit Yield (kg)**
Plant Factory	Vertical	Lettuce	30 (The Royal Horticultural Society, n.d.)	30	10	1	45	140	10000	300	30	40	7	11	77	77	624.5555556	87.43777778
Raised Bed	Horizontal	Lettuce	30	-	-	1	45	140	10000	300	30	-	-	11	-	11	89.22222222	12.49111111
Raised Bed	Horizontal	Dwarf Bush Cherry Tomato	60	-	-	90	57	15	10000	300	60	-	-	2	-	180	-	2.7
Aeroponics	Vertical	Lettuce	30	30	-	1	45	140	10000	300	30	-	-	11	-	11	89.22222222	12.49111111
NFT	Vertical	Lettuce	30	30	7.5	1	45	140	10000	300	30	37.5	8	11	88	88	713.7777778	99.92888889
Water Culture	Horizontal	Lettuce	30	-	-	1	45	140	10000	300	30	-	-	11	-	11	89.22222222	12.49111111
Media Bed	Horizontal	Lettuce	30	-	-	1	45	140	10000	300	30	-	-	11	-	11	89.22222222	12.49111111
Media Bed	Horizontal	Beefsteak Tomato	60	-	-	20	57	350	10000	300	60	-	-	2	-	40	256.1403509	89.64912281

Butterhead Lettuce Horizontal Distance (The Royal Horticultural Society, n.d.)
Butterhead Lettuce Vertical Distance (Viscon Group, 2021)
Butterhead Lettuce Growth Time (Mcintosh, 2021)
Butterhead Lettuce Weight (Baras, 2018)
Dwarf Bush Cherry Tomato Horizontal Distance (LeBoeuf, 2013)
Dwarf Bush Cherry Tomato Growth Time (All About Growing Dwarf Tomatoes Guide and Q&A, n.d.)
Dwarf Bush Cherry TomatoWeight ("Fresh Tomato Weights (Ultimate Guide With Charts & Calculator)," 2021)
Beefsteak Tomato (Gillette, 2022)
Beefsteak Tomato Weight ("Fresh Tomato Weights (Ultimate Guide With Charts & Calculator)," 2021)
NFT Channel Depth (Cropking, n.d)
Plant Factory Tray Depth (Pipp Horticulture, n.d.)

System Name	Horizontal/Vertical	Crop Type	Fruit Yield Annual	Annual Fruit Yield (kg)
Plant Factory	Vertical	Lettuce	624.5555556	87.43777778
Raised Bed	Horizontal	Lettuce	89.22222222	12.49111111
Raised Bed	Horizontal	Dwarf Bush Cherry Tomato	-	2.7
Aeroponics	Vertical	Lettuce	89.22222222	12.49111111
NFT	Vertical	Lettuce	713.7777778	99.92888889
Water Culture	Horizontal	Lettuce	89.22222222	12.49111111
Media Bed	Horizontal	Lettuce	89.22222222	12.49111111
Media Bed	Horizontal	Beefsteak Tomato	256.1403509	89.64912281

FISH

Product	weight range min. (gr)	weight range max. (gr)	tank density kg/m2	tank volume L	tank volume m3	number of fish	total kg of fish	max	average kg of fish per m2
small fish	0	100	20	750	0.75	150	15	15	15
medium fish	100	250	25	750	0.75	75	7.5	18.75	13.125
big fish	250	450	30	750	0.75	50	12.5	22.5	17.5

Values and formulas retrieved from *"Building Integrated Technical Food Systems.* by Jenkins, A. (2018)."
https://pure.qub.ac.uk/en/studentTheses/building-integrated-technical-food-systems

FISH FOOD DEMAND

Product	Daily Food (kg/m2)	Daily Worm Count	Annual Worm Count/ m2	Worm Count (per 6 weeks)	Available Worm Count (Fish Food) (per 6 weeks)
small fish	0.15	600	219000	25200	19504.21407
medium fish	0.0875	350	127750	14700	19504.21407
big fish	0.05833333333	236	86140	9912	19504.21407

Small fish (0g-100g) should be fed 3% of their body weight daily (Jenkins, 2018)
Medium fish (100g-250g) should be fed 2% of their body weight daily (Jenkins, 2018)
Large fish (250g-450g) should be fed 1% of their body weight daily (Jenkins, 2018)
1/3 of food can be worms (Jenkins, 2018)
1 worm weights 0.25 grams (Compost Community,n.d.)

WORMS

worm weight per square foot (pound/ sq foot)	worm weight per m2 (gr/m2)	worm weight per m2 (kg/m2)	weight of 1 worm (gr)	area (cm2)	worm count
1 pound	4882	4.882	0.25	10000	19528

Worm Weight per square foot (Fun Facts about Compost Worms, n.d.)

Appendix E TU Delft Analysis

APPENDIX E TU DELFT ANALYSIS

E.0 ANALYSIS FRAMEWORK

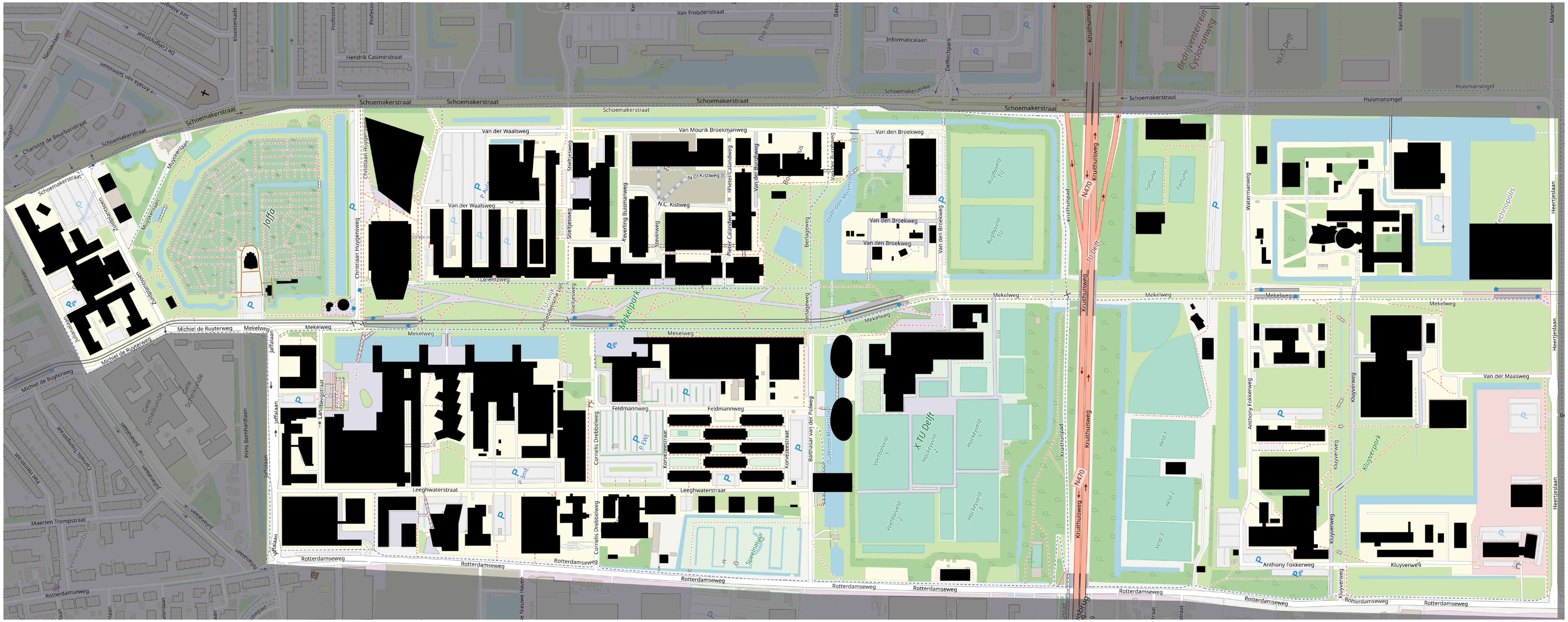
VACANT SPACE CHARACTERISTICS

Location	Data	Interpretation 1	Interpretation 2
Location A	Basement	stuctural capacity : high (3)	solar exposure : low (1)
Location B	Ground Floor	stuctural capacity : high (3)	
Location C	Ground Floor Outdoor	stuctural capacity : high (3)	
Location D	Intermediate	stuctural capacity : medium (2)	
Location E	Rooftop	stuctural capacity : low (1)	solar exposure : high (3)
Location F	Facade	stuctural capacity : low (1)	
	North	solar exposure : low (1)	
	North East	solar exposure : low (1)	
	East	solar exposure : adequate (2)	
	South East	solar exposure : high (3)	
	South	solar exposure : high (3)	
	South West	solar exposure : high (3)	
	West	solar exposure : adequate (2)	
	North West	solar exposure : low (1)	

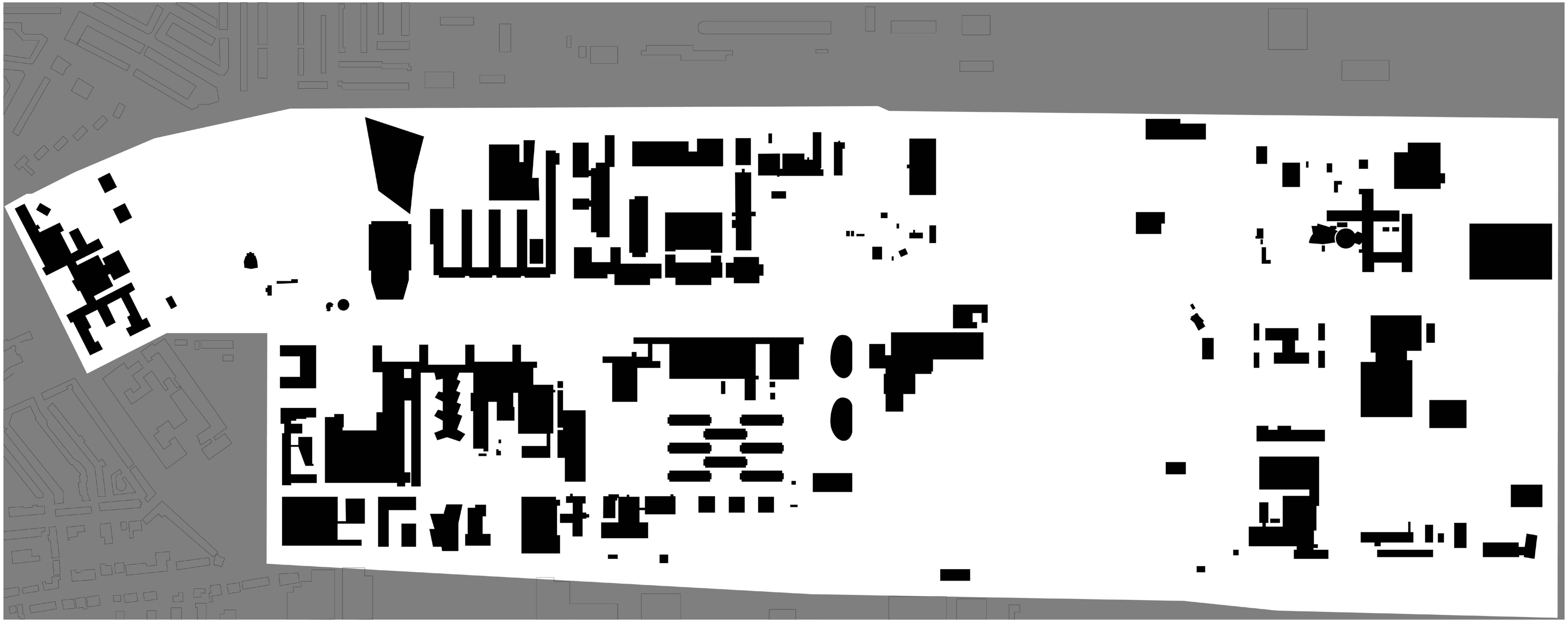
VACANT SPACE CHARACTERISTICS

Location	Data	Interpretation 1	Quantity Information
Location G	Cafeteria	Organic Waste : Food	serving ... people
Location H	Cafe/Restaurant	Organic Waste : Food	serving ... people
Location I	Agricultural Activity	Organic Waste : Food	
Location J	Wood Workshop	Other Waste : Wood Chips	
Location K	School & Offices	Other Waste : Paper	
Location L	Paper Waste Bins	Other Waste : Paper	
Location M	Espresso Bar	Organic Waste : Coffee	serving ... people
Location N	Conference Hall	CO2	serving ... people
Location O	Classroom	CO2	serving ... people
Location P	Meeting Room	CO2	serving ... people
Location R	Metal/ Sloped Roof	Rainwater	surface area
Location S	Sloped Roof	Rainwater	surface area
Location T	Supermarket	Excess Heat Source	
Location U	Datacenter	Excess Heat Source	

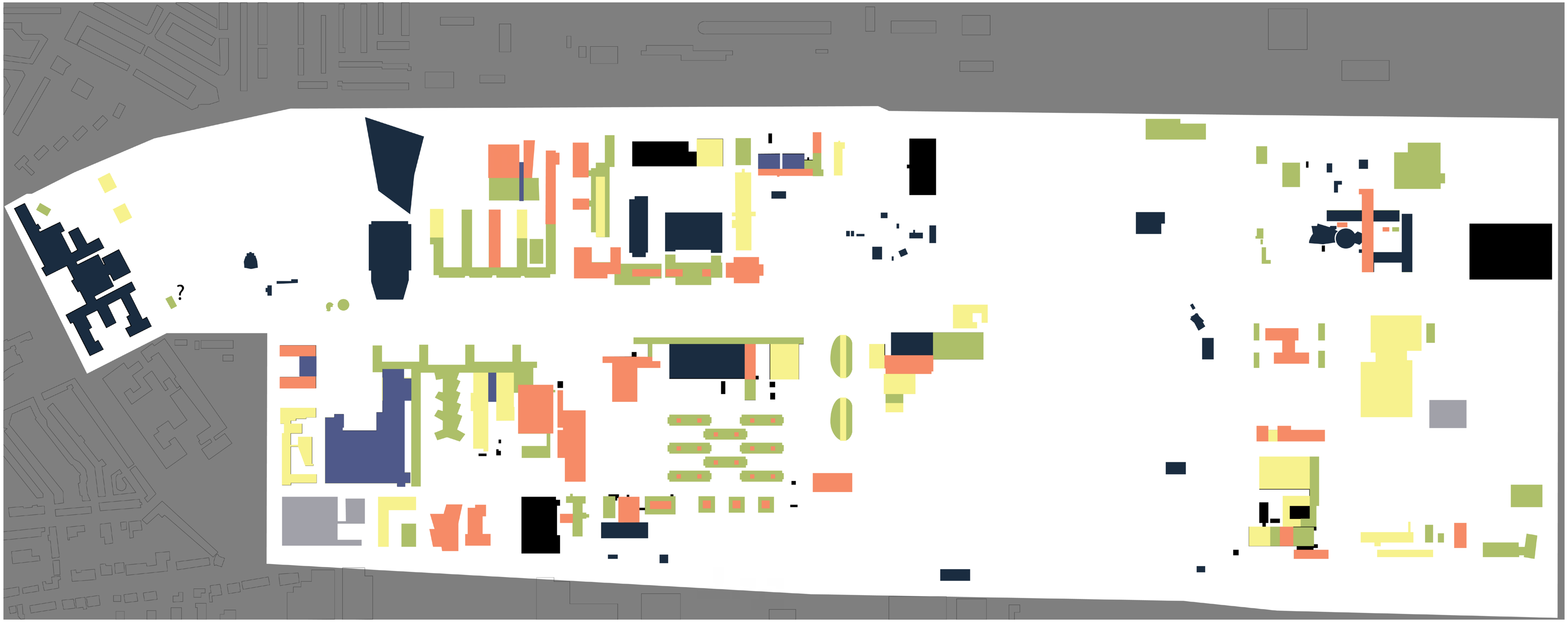
E.1 Aerial View



E.2 Building Footprints



E.3 Analysis of Rooftops’ Availability



🟩 Available

🟨 PV Panels

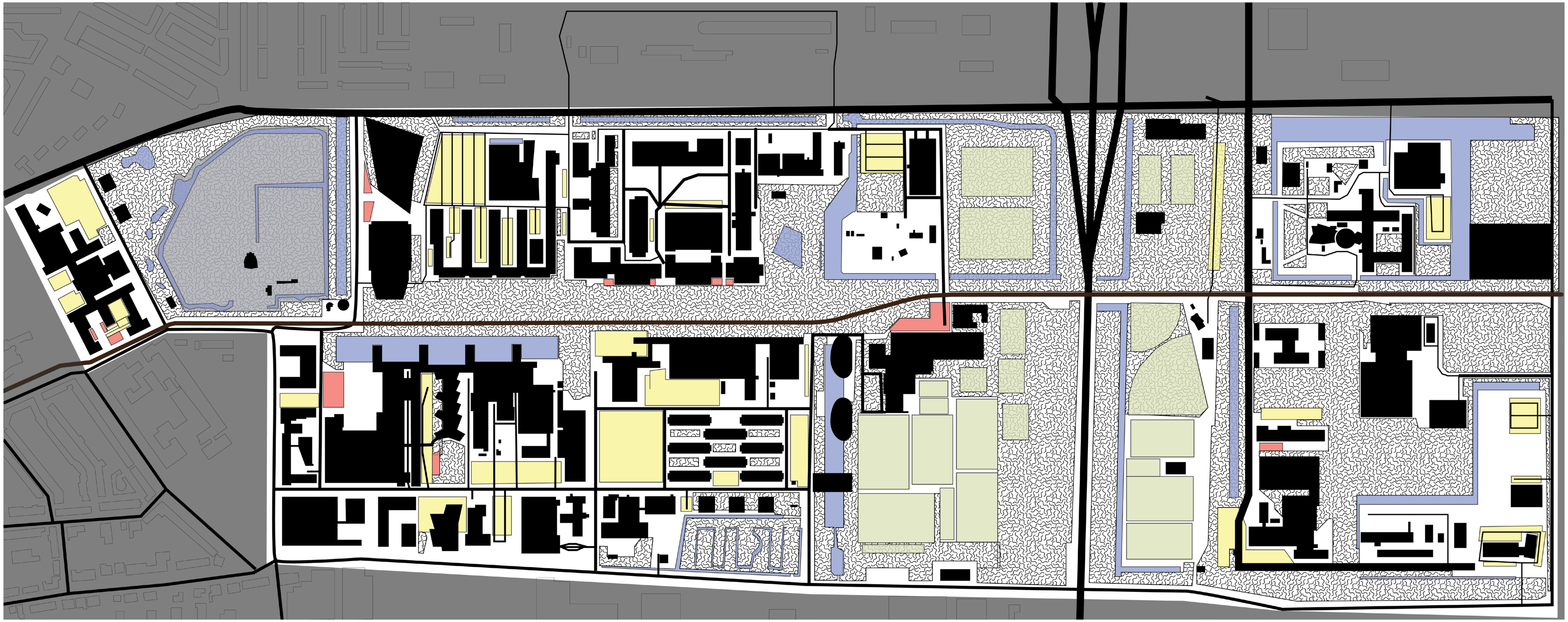
🟦 Sky Light

🟧 Mechanical Equipment

▨ Parking Lot

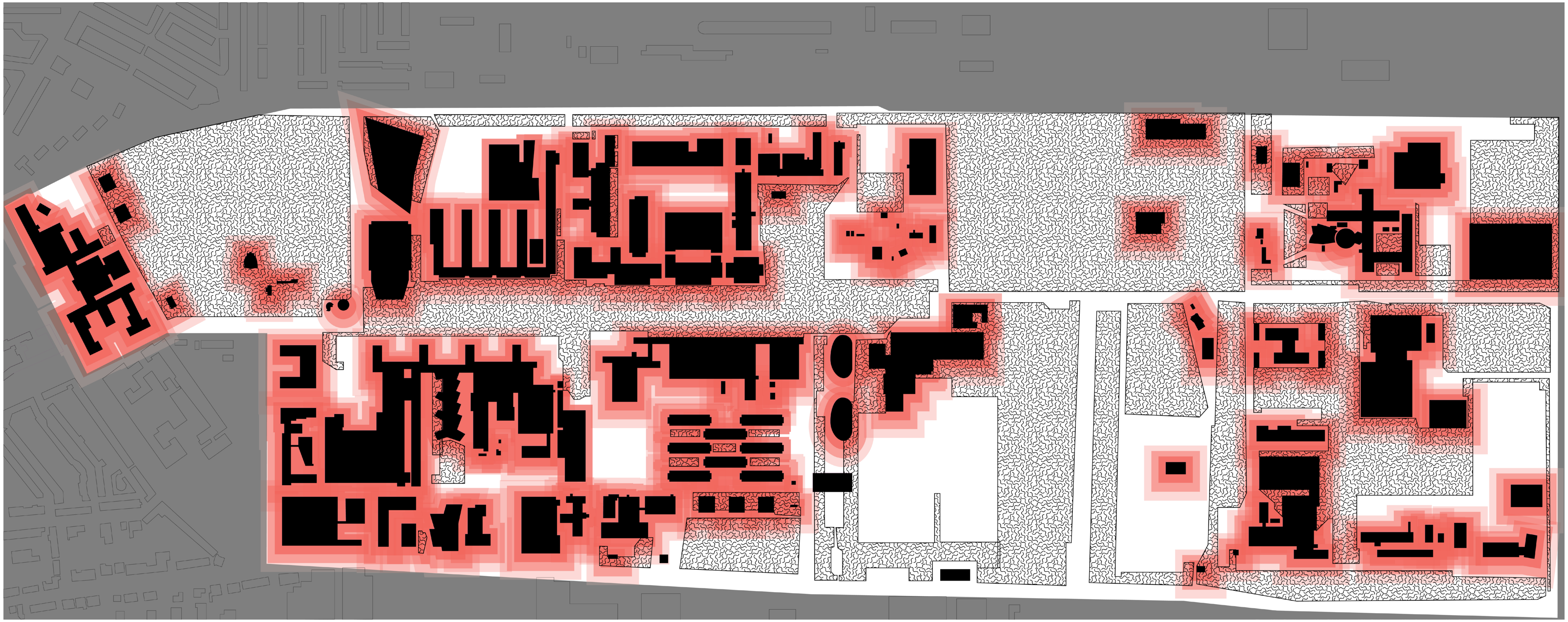
⬛ Sloped Roof

E.4 Analysis of Ground Floor Availability



🟩 Sports Fields
🟨 Bike Storage
🟦 Canals
🟧 Parking Lot
⬜ Cemetery
▨ Greenery

E.5 Radiant Map Showing Vicinity To Buildings



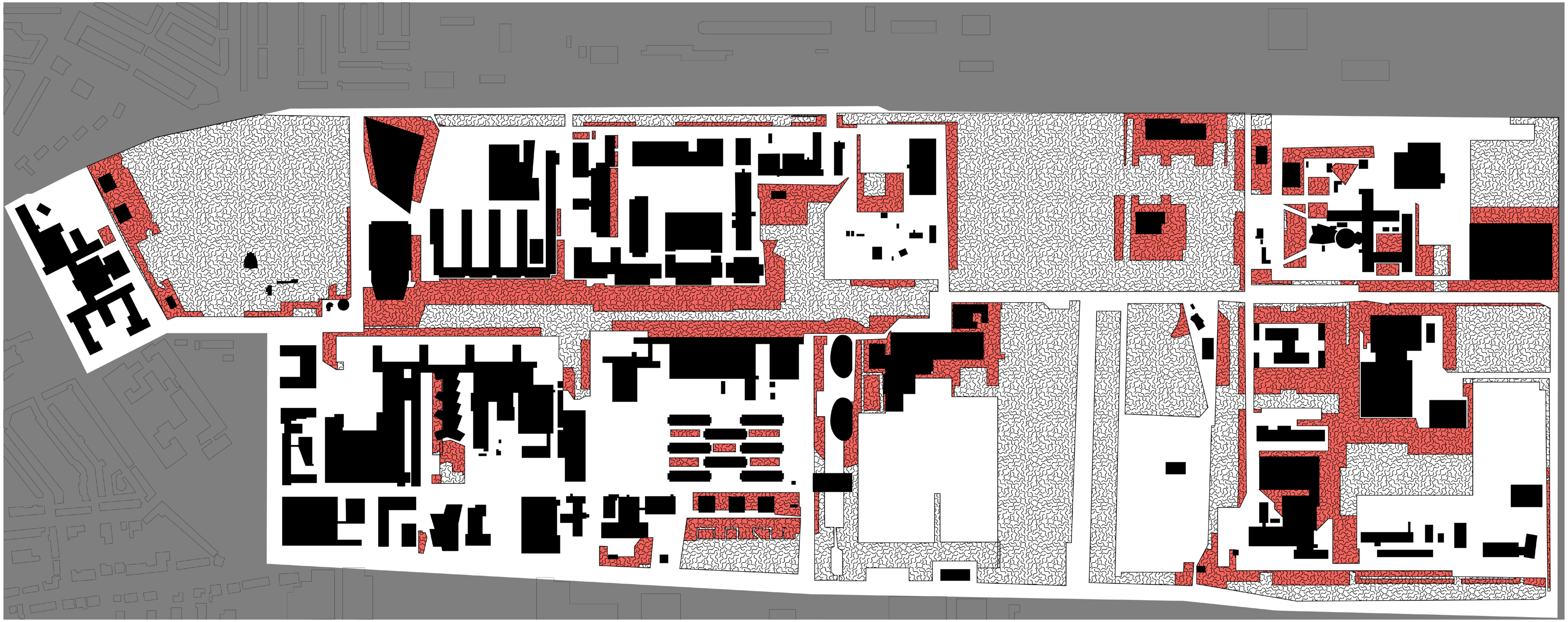
🟥 10 m

🟥 20 m

🟥 30 m

▨ Greenery

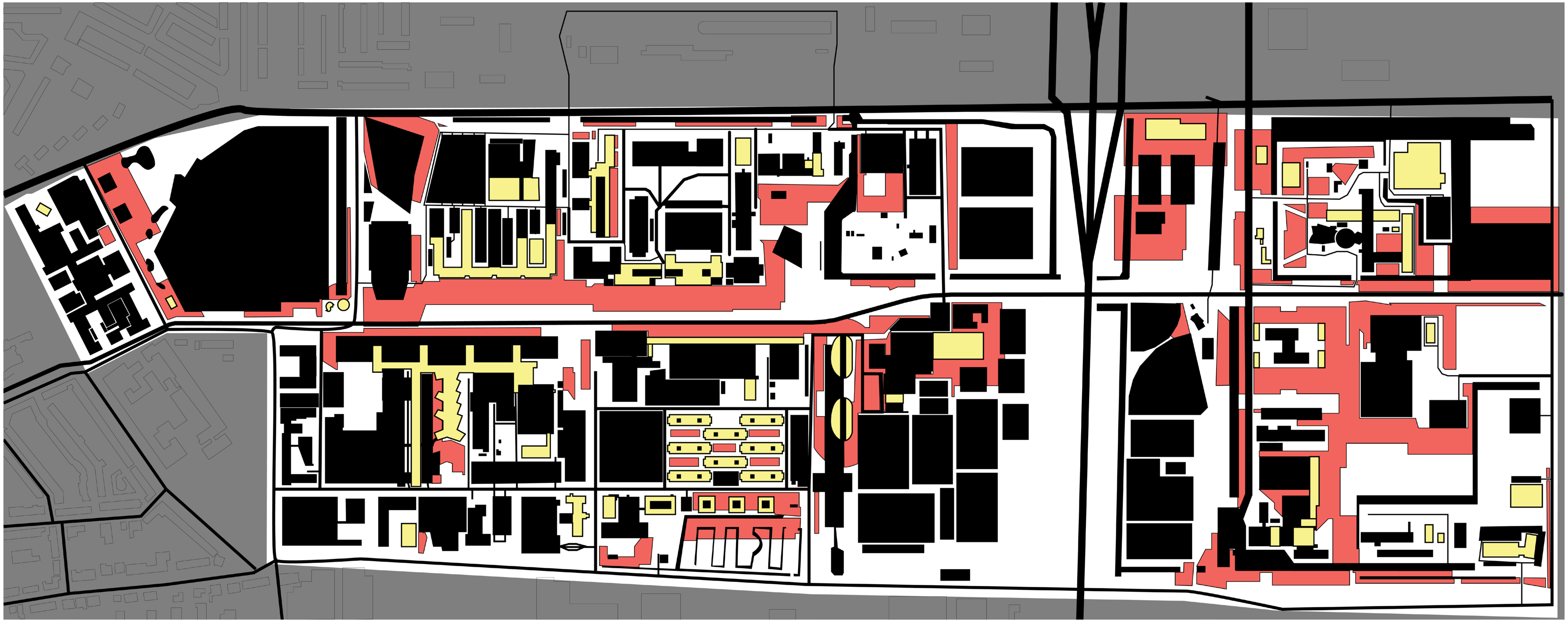
E.6 Overlapping Greenery And Radiant Map



🟥 Overlapping

▨ Greenery

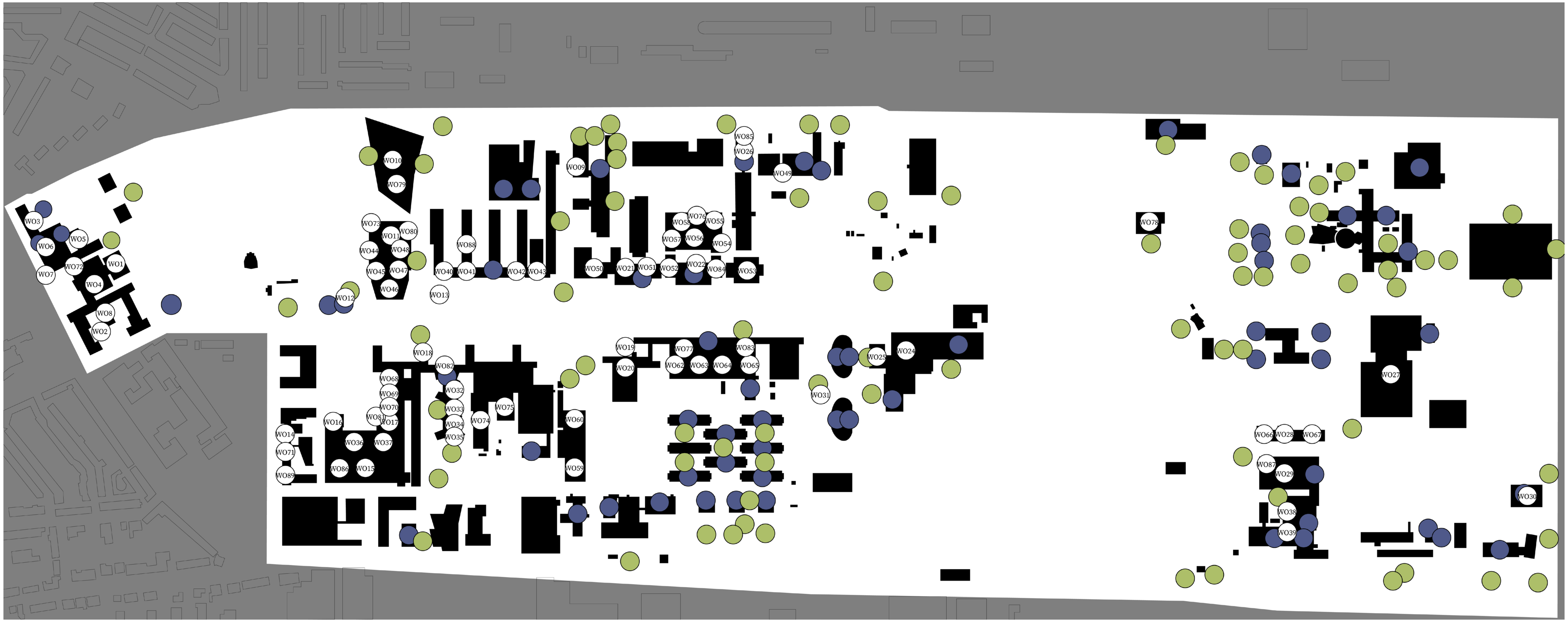
E.7 Vacant Spaces



🟥 Available Ground Floor Area

🟨 Available Rooftop

E.8 Nodes

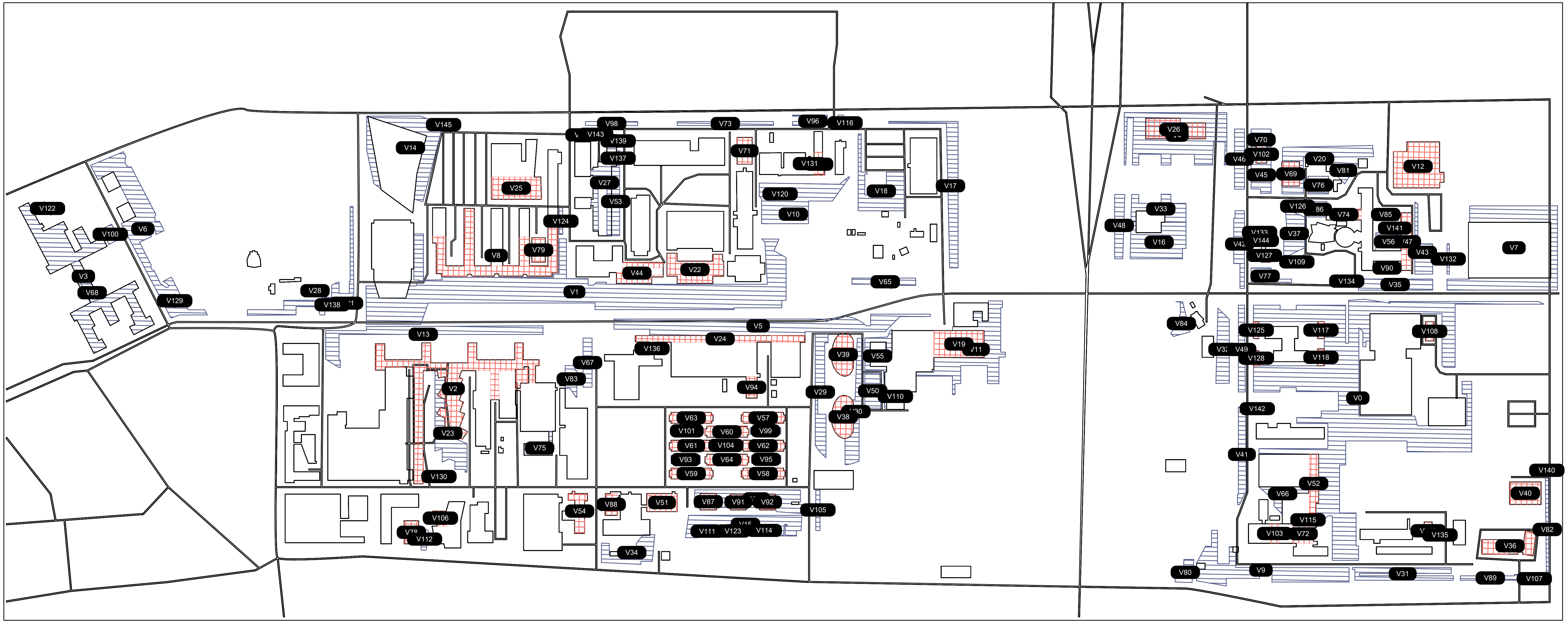


🟢 Available Ground Floor Area Nodes

🔵 Available Rooftop Nodes

(WO1) Waste Output Nodes

E.9 Sorted Vacant Space Nodes



▦ Vacant Rooftop

▤ Vacant Ground Floor

E.10 Vacant Space Characteristics

Identifier	Coordinates	Size	Building	Location	Orientation	Tag	Node Type
V0	{2020.580643, 387.391529, 0}	31174.53666	outside	outside	S	V0	vacant space
V1	{935.001269, 534.594454, 0}	23824.09908	outside	outside	S	V1	vacant space
V2	{766.997201, 400.341737, 12.0}	9665.369642	3me	roof	S	V2	vacant space
V3	{254.491041, 556.132177, 0}	8429.633469	BK	basement	S	V3	vacant space
V4	{1770.13761, 751.608275, 0}	8272.572478	outside	outside	S	V4	vacant space
V5	{1189.499865, 487.345394, 0}	8061.2872	outside	outside	E	V5	vacant space
V6	{336.903473, 621.882305, 0}	6439.317191	outside	outside	S	V6	vacant space
V7	{2237.009725, 595.715597, 0}	6015.400281	outside	outside	E	V7	vacant space
V8	{826.49752, 584.987928, 12}	5436.112064	AS	roof	S	V8	vacant space
V9	{1886.407729, 148.985643, 0}	4880.694962	outside	outside	S	V9	vacant space
V10	{1238.274556, 660.86452, 0}	4720.707857	outside	outside	S	V10	vacant space
V11	{1491.382251, 454.896508, 0}	4436.905613	outside	outside	S	V11	vacant space
V12	{2104.099002, 708.543263, 12}	3912.406602	hollandptc	roof	S	V12	vacant space
V13	{725.481226, 475.783652, 0}	3860.805254	outside	outside	E	V13	vacant space
V14	{707.801875, 734.33, 0}	3795.550842	outside	outside	S	V14	vacant space
V15	{1172.021866, 212.417878, 0}	3414.20724	outside	outside	SW	V15	vacant space
V16	{1745.277986, 603.305799, 0}	3321.599943	outside	outside	SW	V16	vacant space
V17	{1456.101959, 681.594694, 0}	3035.067605	outside	outside	S	V17	vacant space
V18	{1360.266072, 674.2871, 0}	2707.049732	outside	outside	S	V18	vacant space
V19	{1467.87926, 462.26593, 12}	2570.605398	X	roof	S	V19	vacant space
V20	{1967.622502, 719.121831, 0}	2542.354255	outside	outside	E	V20	vacant space
V21	{1187.069112, 247.773475, 0}	2541.191524	outside	outside	NE	V21	vacant space
V22	{1102.010609, 565.395898, 12}	2440.648879	CEG	roof	S	V22	vacant space
V23	{759.513592, 338.631234, 0}	2245.407526	outside	outside	NW	V23	vacant space
V24	{1136.901105, 469.397271, 12}	2167.980456	EEMCS2	roof	S	V24	vacant space
V25	{854.301619, 678.271942, 12}	2144.104305	TNO	roof	S	V25	vacant space
V26	{1765.662254, 759.761065, 12}	2139.318657	V26	roof	S	V26	vacant space
V27	{976.861624, 685.9273, 12}	2111.030067	V27	roof	S	V27	vacant space
V28	{575.482159, 536.19563, 0}	2093.170986	outside	outside	S	V28	vacant space
V29	{1275.73311, 395.652487, 0}	1954.974543	outside	outside	N	V29	vacant space
V30	{1325.122739, 368.657999, 0}	1942.882355	outside	outside	S	V30	vacant space
V31	{2082.810089, 143.808595, 0}	1921.86682	outside	outside	SE	V31	vacant space
V32	{1833.647815, 454.916946, 0}	1665.10129	outside	outside	S	V32	vacant space
V33	{1747.573228, 649.592317, 0}	1658.479829	outside	outside	SE	V33	vacant space
V34	{1014.438887, 173.231638, 0}	1610.590794	outside	outside	W	V34	vacant space
V35	{2072.329024, 544.690135, 0}	1597.51193	outside	outside	SW	V35	vacant space
V36	{2231.559493, 182.612876, 12}	1578.782096	Radex	roof	S	V36	vacant space
V37	{1932.550756, 615.437238, 0}	1572.935303	outside	outside	N	V37	vacant space
V38	{1307.476123, 361.299511, 12}	1501.150494	V38	roof	S	V38	vacant space

Identifier	Coordinates	Size	Building	Location	Orientation	Tag	Node Type
V39	{1307.476123, 447.714593, 12}	1501.150494	V39	roof	S	V39	vacant space
V40	{2252.746342, 255.457019, 12}	1354.283896	datacenter	roof	S	V40	vacant space
V41	{1860.04359, 309.245812, 0}	1340.761573	outside	outside	N	V41	vacant space
V42	{1856.896197, 600.868463, 0}	1288.877422	outside	outside	S	V42	vacant space
V43	{2110.090435, 589.321934, 0}	1275.764993	outside	outside	S	V43	vacant space
V44	{1021.549269, 560.580277, 12}	1233.523908	CEG	roof	S	V44	vacant space
V45	{1886.971833, 696.66426, 0}	1231.775195	outside	outside	W	V45	vacant space
V46	{1856.612474, 718.585648, 0}	1153.400093	outside	outside	S	V46	vacant space
V47	{2087.886501, 604.053783, 12}	1148.748086	AS(reactor)	roof	S	V47	vacant space
V48	{1689.948378, 626.657299, 0}	1130.80954	outside	outside	S	V48	vacant space
V49	{1859.611528, 454.743545, 0}	1091.599357	outside	outside	S	V49	vacant space
V50	{1348.321473, 397.432156, 0}	1060.91461	outside	outside	NE	V50	vacant space
V51	{1056.358892, 242.452808, 12}	1055.834638	TNO2	roof	S	V51	vacant space
V52	{1959.42501, 268.942099, 12}	1044.105087	AE	roof	S	V52	vacant space
V53	{992.176817, 659.846045, 0}	1034.014055	outside	outside	S	V53	vacant space
V54	{942.191497, 230.737664, 12}	950.174407	windtunnellab	roof	S	V54	vacant space
V55	{1355.143487, 445.71046, 0}	937.010293	outside	outside	N	V55	vacant space
V56	{2063.13655, 604.053783, 0}	886.901096	outside	outside	N	V56	vacant space
V57	{1196.644963, 359.892335, 12}	886.221007	V57	roof	S	V57	vacant space
V58	{1146.306081, 340.595763, 12}	886.221007	V58	roof	S	V58	vacant space

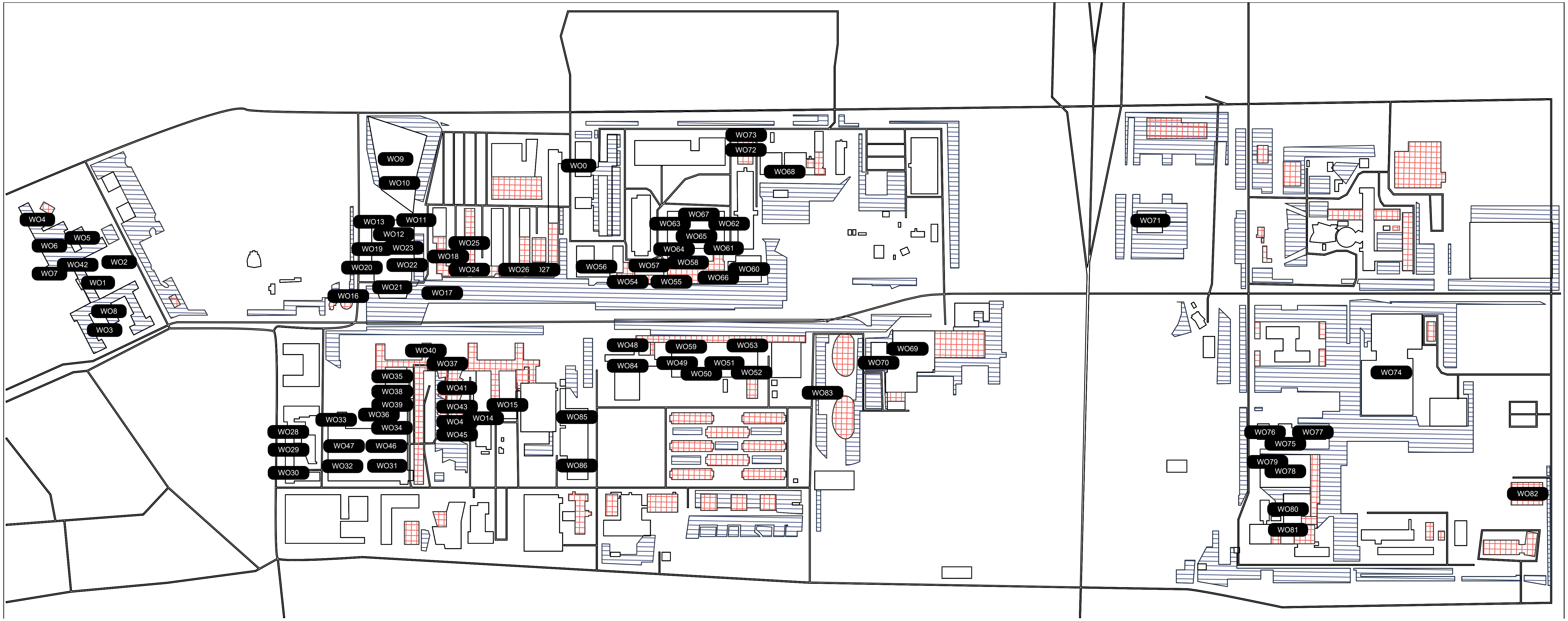
Identifier	Coordinates	Size	Building	Location	Orientation	Tag	Node Type
V59	{1196.644963, 321.299191, 12.0}	886.221007	V59	roof	S	V59	vacant space
V60	{1196.644963, 282.706047, 12}	886.221007	V60	roof	S	V60	vacant space
V61	{1096.806179, 359.892335, 12}	886.221007	V61	roof	S	V61	vacant space
V62	{1146.306081, 302.002619, 12}	886.221007	V62	roof	S	V62	vacant space
V63	{1096.806179, 321.299191, 12}	886.221007	V63	roof	S	V63	vacant space
V64	{1096.806179, 282.706047, 12}	886.221007	V64	roof	S	V64	vacant space
V65	{1365.642899, 548.590606, 0}	884.459846	outside	outside	S	V65	vacant space
V66	{1916.474869, 254.754857, 0}	871.979286	outside	outside	N	V66	vacant space
V67	{953.203195, 436.507943, 0}	849.661333	outside	outside	S	V67	vacant space
V68	{257.114541, 548.516607, 9}	817.90353	BK	intermediate	S	V68	vacant space
V69	{1927.641057, 698.019698, 12}	816.51212	V69	roof	S	V69	vacant space
V70	{1886.441857, 745.460984, 0}	784.696363	outside	outside	E	V70	vacant space
V71	{1170.88866, 730.078854, 12}	782.431852	EEMCS	roof	S	V71	vacant space
V72	{1944.840175, 199.245266, 12}	741.899806	AE	roof	S	V72	vacant space
V73	{1144.065167, 767.895521, 0}	735.651059	outside	outside	SE	V73	vacant space
V74	{2001.02806, 641.807945, 12}	734.14002	AS(reactor)	roof	S	V74	vacant space
V75	{886.807134, 318.03995, 12}	731.337236	ProcessEnergy	roof	S	V75	vacant space
V76	{1965.81471, 682.498542, 0}	694.035302	outside	outside	S	V76	vacant space
V77	{1892.075078, 556.457122, 0}	669.748444	outside	outside	W	V77	vacant space
V78	{709.0206, 201.34272, 12}	641.94746	V78	roof	S	V78	vacant space

Identifier	Coordinates	Size	Building	Location	Orientation	Tag	Node Type
V79	{885.322889, 592.55806, 12}	636.567695	AS	roof	S	V79	vacant space
V80	{1781.529949, 145.808698, 0}	619.178938	outside	outside	S	V80	vacant space
V81	{2000.503935, 702.969141, 0}	603.53602	outside	outside	SW	V81	vacant space
V82	{2283.175338, 205.215859, 0}	566.432602	outside	outside	S	V82	vacant space
V83	{930.622033, 413.865393, 0}	560.237079	outside	outside	E	V83	vacant space
V84	{1776.14714, 490.752615, 0}	559.772953	outside	outside	S	V84	vacant space
V85	{2059.209388, 641.807945, 12}	549.393913	AS(reactor)	roof	S	V85	vacant space
V86	{1964.556238, 648.939287, 0}	545.514563	outside	outside	N	V86	vacant space
V87	{1120.540968, 243.71128, 12}	513.135634	V87	roof	S	V87	vacant space
V88	{985.884456, 239.935863, 12}	506.800626	TNO2	roof	S	V88	vacant space
V89	{2204.207461, 138.013049, 0}	501.818741	outside	outside	S	V89	vacant space
V90	{2061.039097, 567.558093, 0}	494.8345	outside	outside	W	V90	vacant space
V91	{1162.070546, 243.291789, 12}	475.829477	V91	roof	S	V91	vacant space
V92	{1202.341653, 243.291789, 12}	475.829477	V92	roof	S	V92	vacant space
V93	{1089.079166, 302.020486, 0}	473.013918	outside	outside	E	V93	vacant space
V94	{1180.528137, 402.278761, 12.0}	443.450548	EEMCS2	roof	S	V94	vacant space
V95	{1200.66369, 302.439976, 0}	439.227209	outside	outside	E	V95	vacant space
V96	{1265.429833, 771.166212, 0}	436.388609	outside	outside	E	V96	vacant space
V97	{1273.333716, 711.489837, 12}	402.637612	Bouwcampus	roof	S	V97	vacant space
V98	{986.590205, 767.909158, 0}	388.062522	outside	outside	E	V98	vacant space
V99	{1199.824709, 341.872101, 0}	387.139367	outside	outside	E	V99	vacant space
V100	{291.760006, 614.43794, 0}	373.210485	outside	outside	S	V100	vacant space
V101	{1090.757129, 341.872101, 0}	371.653793	outside	outside	E	V101	vacant space
V102	{1886.95046, 725.286593, 12}	367.430454	V102	roof	S	V102	vacant space
V103	{1905.40805, 199.664757, 12}	360.391556	AE	roof	S	V103	vacant space
V104	{1144.871428, 321.736548, 0}	338.570974	outside	outside	E	V104	vacant space
V105	{1272.371848, 232.305328, 0}	338.339612	outside	outside	E	V105	vacant space
V106	{749.263449, 220.917178, 12}	329.297226	Inholland	roof	S	V106	vacant space
V107	{2265.186467, 137.043157, 0}	319.677165	outside	outside	S	V107	vacant space
V108	{2120.187284, 479.884539, 12}	315.342612	V108	roof	S	V108	vacant space
V109	{1936.138471, 576.098149, 0}	307.896866	outside	outside	S	V109	vacant space
V110	{1379.786215, 389.69404, 12}	306.192045	X	roof	S	V110	vacant space
V111	{1119.240678, 202.74666, 0}	285.118209	outside	outside	W	V111	vacant space
V112	{727.310526, 191.879847, 0}	273.904622	outside	outside	NW	V112	vacant space
V113	{2118.089831, 203.440173, 12}	265.366439	catalysislab	roof	S	V113	vacant space
V114	{1198.128771, 203.891088, 0}	248.228987	outside	outside	W	V114	vacant space
V115	{1951.552026, 218.541838, 12}	237.914738	AE	roof	S	V115	vacant space
V116	{1310.048145, 768.8042, 0}	221.053856	outside	outside	S	V116	vacant space
V117	{1969.590126, 481.562501, 12}	216.798046	V117	roof	S	V117	vacant space
V118	{1969.590126, 443.808339, 12.0}	216.798046	V118	roof	S	V118	vacant space

Identifier	Coordinates	Size	Building	Location	Orientation	Tag	Node Type
V119	{946.871822, 752.133997, 0}	209.055258	outside	outside	E	V119	vacant space
V120	{1219.960262, 670.333312, 0}	202.72025	outside	outside	S	V120	vacant space
V121	{618.830102, 518.897005, 12}	199.632282	donercompany	roof	S	V121	vacant space
V122	{204.792622, 650.197759, 12}	192.161373	bouwpub	roof	S	V122	vacant space
V123	{1154.60924, 203.113098, 0}	191.032928	outside	outside	W	V123	vacant space
V124	{916.249002, 632.57915, 0}	185.826896	outside	outside	S	V124	vacant space
V125	{1879.819118, 481.562501, 12}	177.380219	V125	roof	S	V125	vacant space
V126	{1937.378834, 653.267983, 0}	177.133105	outside	outside	S	V126	vacant space
V127	{1891.150962, 585.077942, 12}	166.899954	V127	roof	S	V127	vacant space
V128	{1879.819118, 442.549867, 12}	158.375196	V128	roof	S	V128	vacant space
V129	{381.049617, 522.281747, 12}	149.579037	boathouse	roof	S	V129	vacant space
V130	{747.219077, 278.590871, 0}	147.355996	outside	outside	S	V130	vacant space
V131	{1261.070349, 712.282381, 12}	137.962393	Bouwcampus	roof	S	V131	vacant space
V132	{2145.817746, 580.142814, 0}	137.293697	outside	outside	S	V132	vacant space
V133	{1884.449721, 616.882019, 12}	133.566698	V133	roof	S	V133	vacant space
V134	{2005.255424, 549.436989, 0}	107.033708	outside	outside	S	V134	vacant space
V135	{2134.449968, 197.567304, 12}	105.583464	catalysislab	roof	S	V135	vacant space
V136	{1042.515699, 455.973569, 12.0}	103.471795	EEMCS2	roof	S	V136	vacant space
V137	{995.532742, 719.833214, 0}	84.466771	outside	outside	S	V137	vacant space
V138	{599.151462, 516.677845, 12}	83.121305	stud	roof	S	V138	vacant space

Identifier	Coordinates	Size	Building	Location	Orientation	Tag	Node Type
V139	{995.532742, 743.744183, 0}	73.908425	outside	outside	S	V139	vacant space
V140	{2282.543163, 287.235901, 0}	56.943728	outside	outside	S	V140	vacant space
V141	{2071.945855, 622.930864, 12}	54.199511	roof	outside	S	V141	vacant space
V142	{1881.287336, 372.668171, 0}	47.626941	outside	outside	S	V142	vacant space
V143	{964.490431, 752.972978, 0}	45.752834	outside	outside	E	V143	vacant space
V144	{1886.95046, 605.731746, 12.0}	22.524472	V144	roof	S	V144	vacant space
V145	{753.644985, 766.359989, 0}	17.542418	outside	outside	S	V145	vacant space

E.11 Waste Source Nodes



▦ Vacant Rooftop

▤ Vacant Ground Floor

E.12 Waste Source Data

Identifier	Coordinates	Source	Waste Type	Quantity	Tag	Node Type	Waste
WO0	{940.159971, 709.345779, 0}	V27	W7	1732320	WO0	waste	heat (units MJ)
WO1	{274.847735, 547.422372, 0}	BK	W2	3360	WO1	waste	sawdust (units kg)
WO2	{304.212084, 575.947739, 0}	BK	W1	938	WO2	waste	food (units kg)
WO3	{284.076531, 481.981824, 0}	BK	W3	50412.011	WO3	waste	paper (units kg)
WO4	{190.949597, 634.676436, 0}	BK	W4	40142.31698	WO4	waste	coffee (units kg)
WO5	{253.034219, 609.506994, 0}	BK	W5	170.2	WO5	waste	co2 (units kg)
WO6	{207.729225, 598.600236, 0}	BK	W5	92	WO6	waste	co2
WO7	{207.729225, 560.007093, 0}	BK	W5	72.68	WO7	waste	co2
WO8	{289.9494, 507.990247, 0}	BK	W5	55.2	WO8	waste	co2
WO9	{686.787594, 718.574574, 0}	Lib	W3	477438.75	WO9	waste	paper
WO10	{692.240973, 685.434809, 0}	Lib	W4	2864.4	WO10	waste	coffee
WO11	{708.181619, 620.833243, 0}	Aula	W4	14857.68302	WO11	waste	coffee
WO12	{684.27065, 614.540883, 0}	Aula	W1	11474	WO12	waste	food
WO13	{657.003755, 631.740001, 0}	Aula	W6	5.26E+06	WO13	waste	rainwater (units L)
WO14	{808.020403, 359.910033, 0}	3me	W3	84960.3727	WO14	waste	paper
WO15	{841.579659, 378.367624, 0}	3me	None	0	WO15	waste	None
WO16	{621.347046, 528.964782, 0}	donercompany	None	0	WO16	waste	None
WO17	{751.38916, 533.159689, 0}	foodtrucks	None	0	WO17	waste	None
WO18	{757.68152, 565.460472, 0}	AS	W5	48.3	WO18	waste	co2
WO19	{654.486811, 593.985839, 0}	Aula	W5	151.8	WO19	waste	co2
WO20	{663.715606, 565.460472, 0}	Aula	W5	151.8	WO20	waste	co2
WO21	{682.173196, 541.130012, 0}	Aula	W5	107.64	WO21	waste	co2
WO22	{694.757917, 567.138434, 0}	Aula	W5	107.64	WO22	waste	co2
WO23	{697.274861, 595.663801, 0}	Aula	W5	460	WO23	waste	co2
WO24	{788.723832, 565.460472, 0}	AS	W5	60.72	WO24	waste	co2
WO25	{788.723832, 602.375652, 0}	AS	None	0	WO25	waste	None
WO26	{857.520305, 565.460472, 0}	AS	W5	69.92	WO26	waste	co2
WO27	{886.045672, 565.460472, 0}	AS	W5	92	WO27	waste	co2
WO28	{538.287889, 341.032952, 0}	TPM	W4	5558.820744	WO28	waste	coffee
WO29	{538.70738, 316.283001, 0}	TPM	W6	2.81E+06	WO29	waste	rainwater
WO30	{538.70738, 284.401709, 0}	TPM	W4	5558.820744	WO30	waste	coffee
WO31	{649.033432, 294.049995, 0}	ID	W4	8359.677183	WO31	waste	coffee
WO32	{613.376723, 293.630504, 0}	ID	None	0	WO32	waste	None
WO33	{604.567418, 357.81258, 0}	ID	W4	8359.677183	WO33	waste	coffee
WO34	{681.753705, 357.81258, 0}	ID	W3	37659.26004	WO34	waste	paper
WO35	{682.173196, 417.799749, 0}	ID	W5	57.96	WO35	waste	co2
WO36	{663.715606, 364.943922, 0}	ID	W1	1017.817839	WO36	waste	food
WO37	{758.520502, 435.418358, 0}	3me	W6	8.70E+06	WO37	waste	rainwater
WO38	{682.173196, 396.825214, 0}	ID	W5	47.84	WO38	waste	co2

Identifier	Coordinates	Source	Waste Type	Quantity	Tag	Node Type	Waste
WO39	{682.173196, 378.367624, 0}	ID	W5	58.88	WO39	waste	co2
WO40	{728.736663, 453.456457, 0}	3me	W4	18859.67192	WO40	waste	coffee
WO41	{771.944204, 401.859102, 0}	3me	W5	115.46	WO41	waste	co2
WO42	{246.741859, 572.172323, 0}	BK	W6	9.77E+06	WO42	waste	rainwater
WO43	{771.944204, 375.850679, 0}	3me	W5	77.28	WO43	waste	co2
WO44	{771.944204, 354.876145, 0}	3me	W5	62.1	WO44	waste	co2
WO45	{771.944204, 337.257536, 0}	3me	W5	62.1	WO45	waste	co2
WO46	{673.783382, 329.706703, 0}	ID	None	0	WO46	waste	None
WO47	{633.512276, 329.706703, 0}	ID	None	0	WO47	waste	None
WO48	{1008.117463, 461.00729, 0}	EEMCS2	W4	10324.13698	WO48	waste	coffee
WO49	{1076.494445, 436.257339, 0}	EEMCS2	W5	151.34	WO49	waste	co2
WO50	{1110.053701, 436.257339, 0}	EEMCS2	W5	90.62	WO50	waste	co2
WO51	{1141.934993, 436.257339, 0}	EEMCS2	W5	64.4	WO51	waste	co2
WO52	{1179.689155, 436.257339, 0}	EEMCS2	W5	64.4	WO52	waste	co2
WO53	{1173.816286, 460.587799, 0}	EEMCS2	W3	46508.89631	WO53	waste	paper
WO54	{1008.117463, 570.074869, 0}	CEG	W1	1761.467199	WO54	waste	food
WO55	{1068.524122, 570.074869, 0}	CEG	W5	164.68	WO55	waste	co2
WO56	{964.909922, 569.655379, 0}	CEG	W5	155.48	WO56	waste	co2
WO57	{1037.901302, 571.333341, 0}	CEG	W5	124.2	WO57	waste	co2
WO58	{1105.858794, 575.528248, 0}	CEG	W5	124.2	WO58	waste	co2

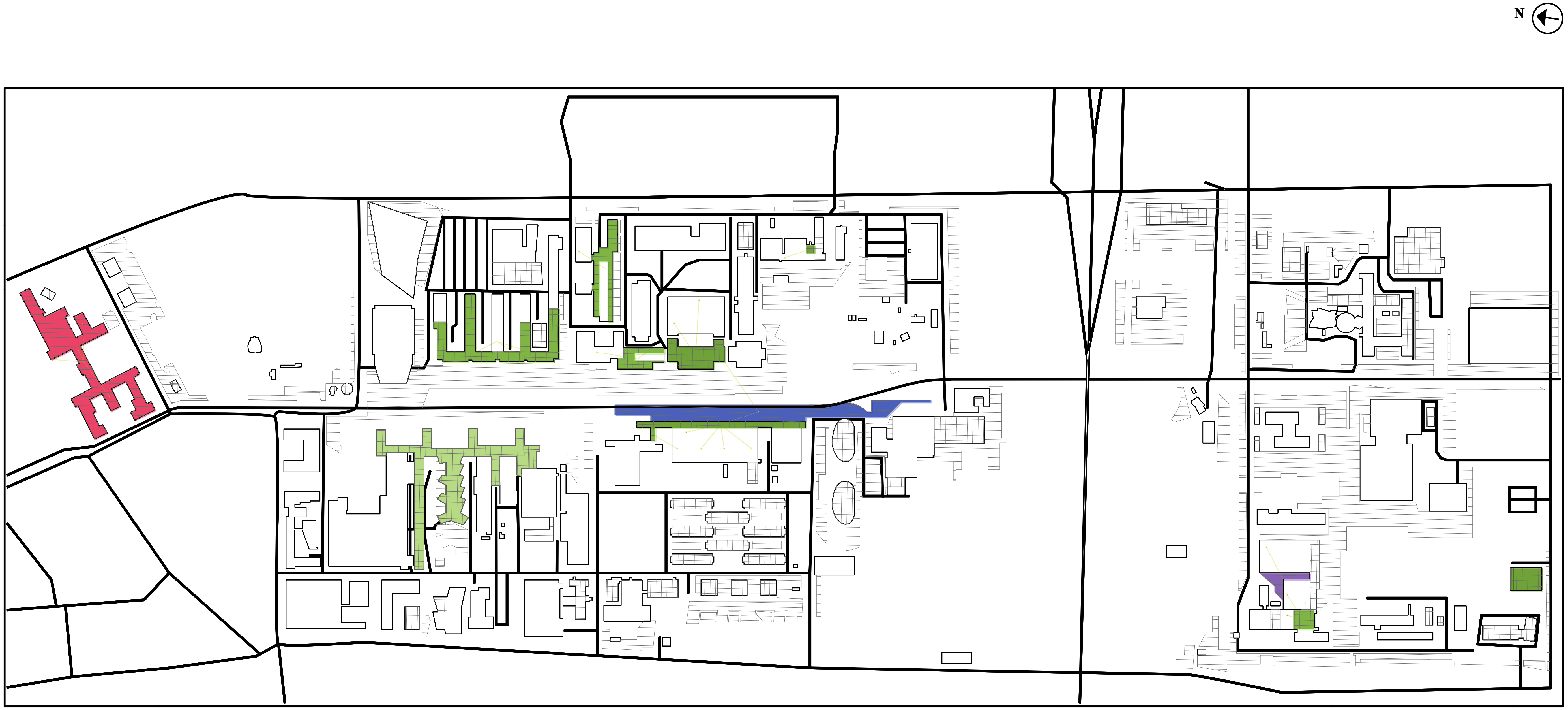
Identifier	Coordinates	Source	Waste Type	Quantity	Tag	Node Type	Waste
WO59	{1089.079166, 458.909836, 0}	EEMCS2	W6	1.18E+07	WO59	waste	rainwater
WO60	{1176.33323, 566.299453, 0}	CEG	W5	92.92	WO60	waste	co2
WO61	{1141.096012, 604.053615, 0}	CEG	W5	50.14	WO61	waste	co2
WO62	{1131.028235, 635.095926, 0}	CEG	W5	50.14	WO62	waste	co2
WO63	{1085.723241, 633.417964, 0}	CEG	W5	55.2	WO63	waste	co2
WO64	{1072.299539, 609.926485, 0}	CEG	W5	57.5	WO64	waste	co2
WO65	{1103.34185, 611.604448, 0}	CEG	W4	14467.51726	WO65	waste	coffee
WO66	{1133.545179, 568.816397, 0}	CEG	W3	65174.28635	WO66	waste	paper
WO67	{1106.697775, 641.807777, 0}	CEG	W6	1.07E+07	WO67	waste	rainwater
WO68	{1225.413641, 700.955965, 0}	Bouwcampus	W5	51.52	WO68	waste	co2
WO69	{1395.726861, 455.973401, 0}	X	None	0	WO69	waste	None
WO70	{1355.455755, 447.583588, 0}	X	None	0	WO70	waste	None
WO71	{1731.738904, 633.417964, 0}	WO71	None	0	WO71	waste	None
WO72	{1171.718832, 731.159295, 0}	EEMCS	W1	1256.997198	WO72	waste	food
WO73	{1172.138323, 751.714338, 0}	EEMCS	W4	10324.13698	WO73	waste	coffee
WO74	{2065.234004, 423.253128, 0}	AS2	W4	10028.18125	WO74	waste	coffee
WO75	{1918.412262, 341.032952, 0}	fellowship	W1	1220.963626	WO75	waste	food
WO76	{1890.306386, 340.613461, 0}	fellowship	W5	52.44	WO76	waste	co2
WO77	{1956.585915, 340.613461, 0}	fellowship	W5	57.96	WO77	waste	co2
WO78	{1918.412262, 286.499162, 0}	AE	W4	11199.13654	WO78	waste	coffee

Identifier	Coordinates	Source	Waste Type	Quantity	Tag	Node Type	Waste
WO79	{1893.662311, 299.503374, 0}	AE	W3	50450.6557	WO79	waste	paper
WO80	{1922.187678, 234.062826, 0}	AE	W5	133.4	WO80	waste	co2
WO81	{1922.187678, 205.537459, 0}	AE	W5	64.4	WO81	waste	co2
WO82	{2254.004814, 255.456851, 0}	datacenter	W7	8625000	WO82	waste	heat
WO83	{1277.849977, 395.147251, 0}	compostbin	None	0	WO83	waste	None
WO84	{1008.117463, 432.481923, 0}	EEMCS2	W1	1256.997198	WO84	waste	food
WO85	{938.062517, 361.587996, 0}	education	W5	147.2	WO85	waste	co2
WO86	{938.062517, 294.469485, 0}	education	W5	123.28	WO86	waste	co2

Appendix F TU Delft Decisions

APPENDIX F TU DELFT DECISIONS

F.1 Stage 1 Assigned Systems



⬜ UF1 Vermiculture

🟦 UF2 Aquaculture

🟪 UF3 Mushroom Farm

🟩 UF4 NFT

🟩 UF5 Media Beds

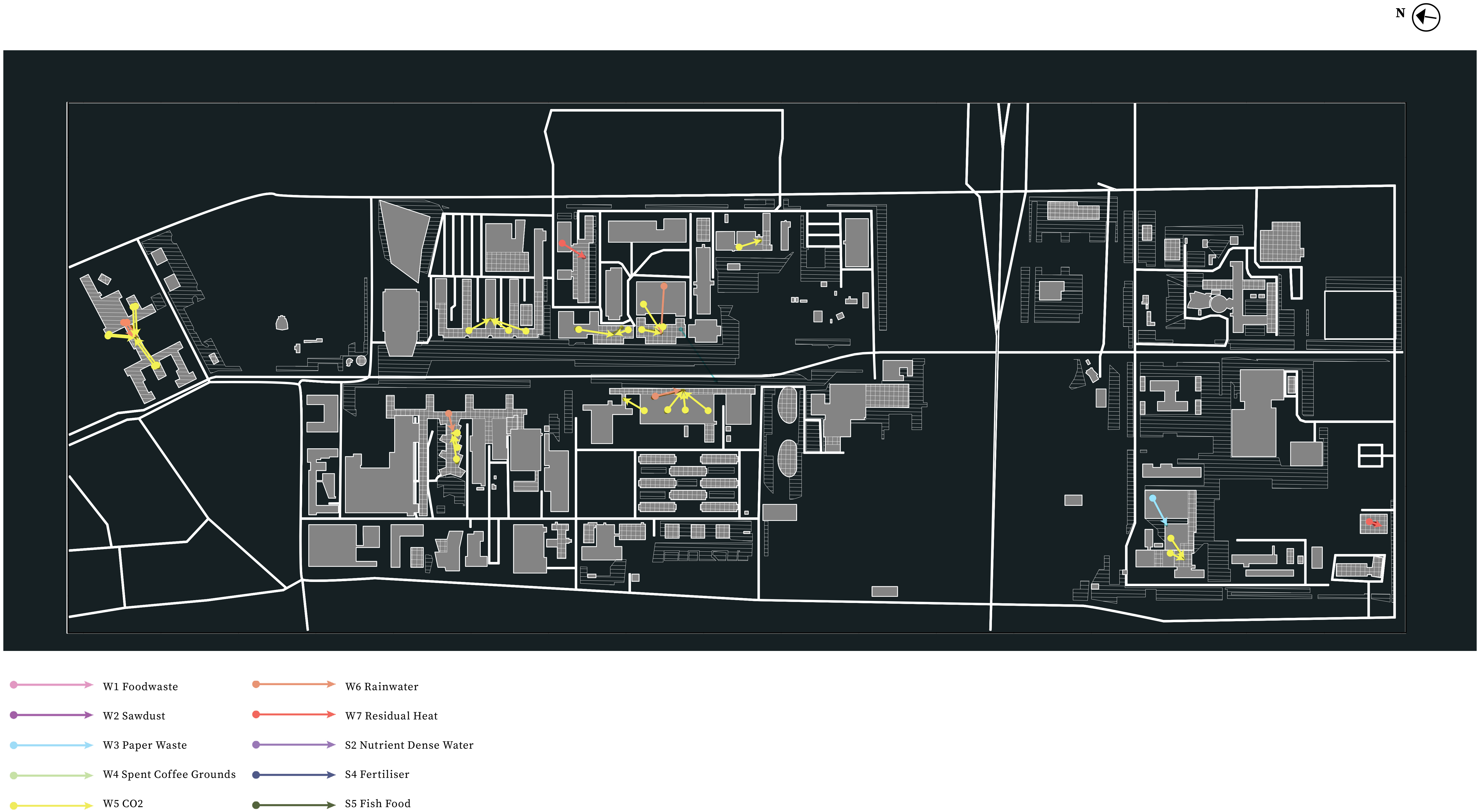
🟧 UF6 Raised Beds

🟩 UF7 Deep Water Culture

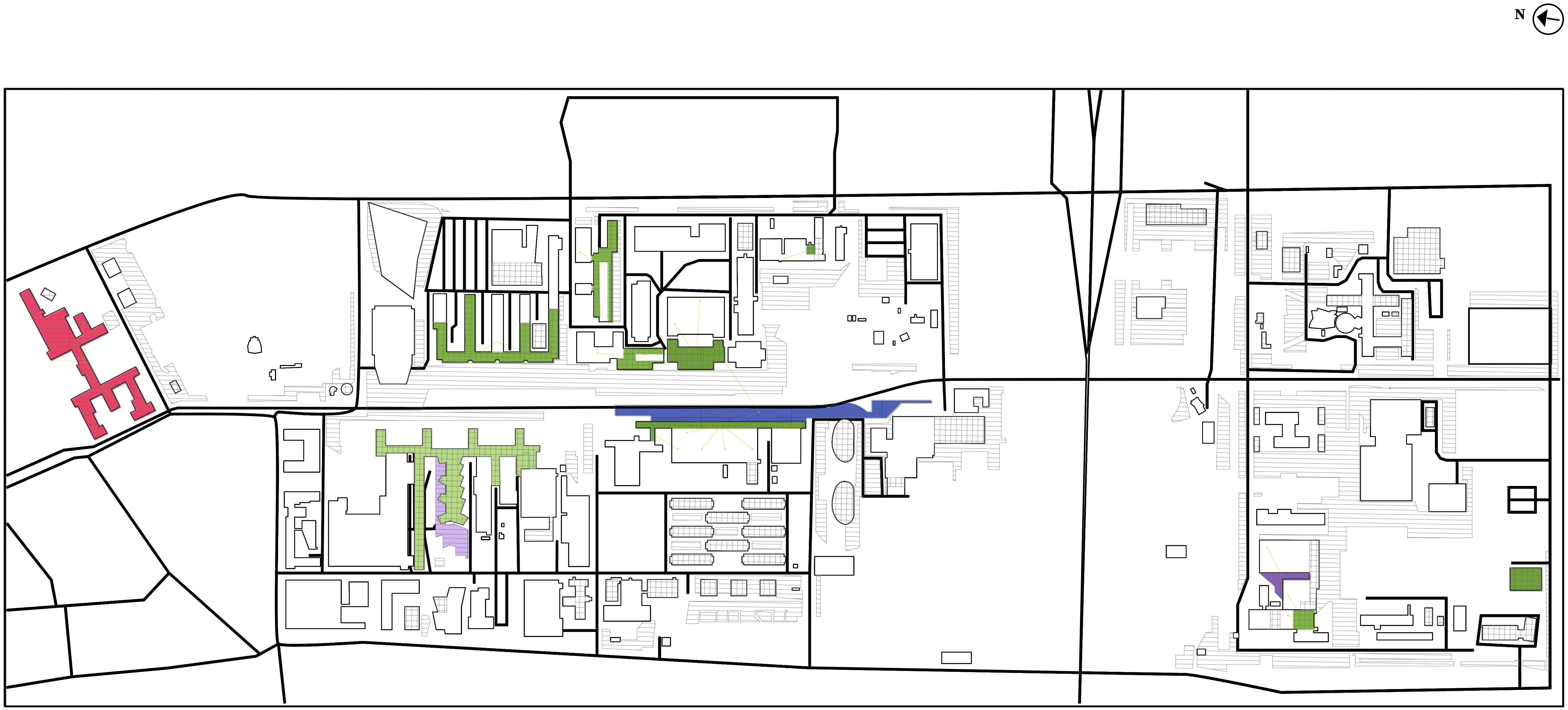
🟥 UF8 Plant Factory

🟨 UF9 Aeroponics

F.2 Stage 1 Connections



F.3 Stage 2 Assigned Systems



⬜ UF1 Vermiculture

🟦 UF2 Aquaculture

🟪 UF3 Mushroom Farm

🟩 UF4 NFT

🟩 UF5 Media Beds

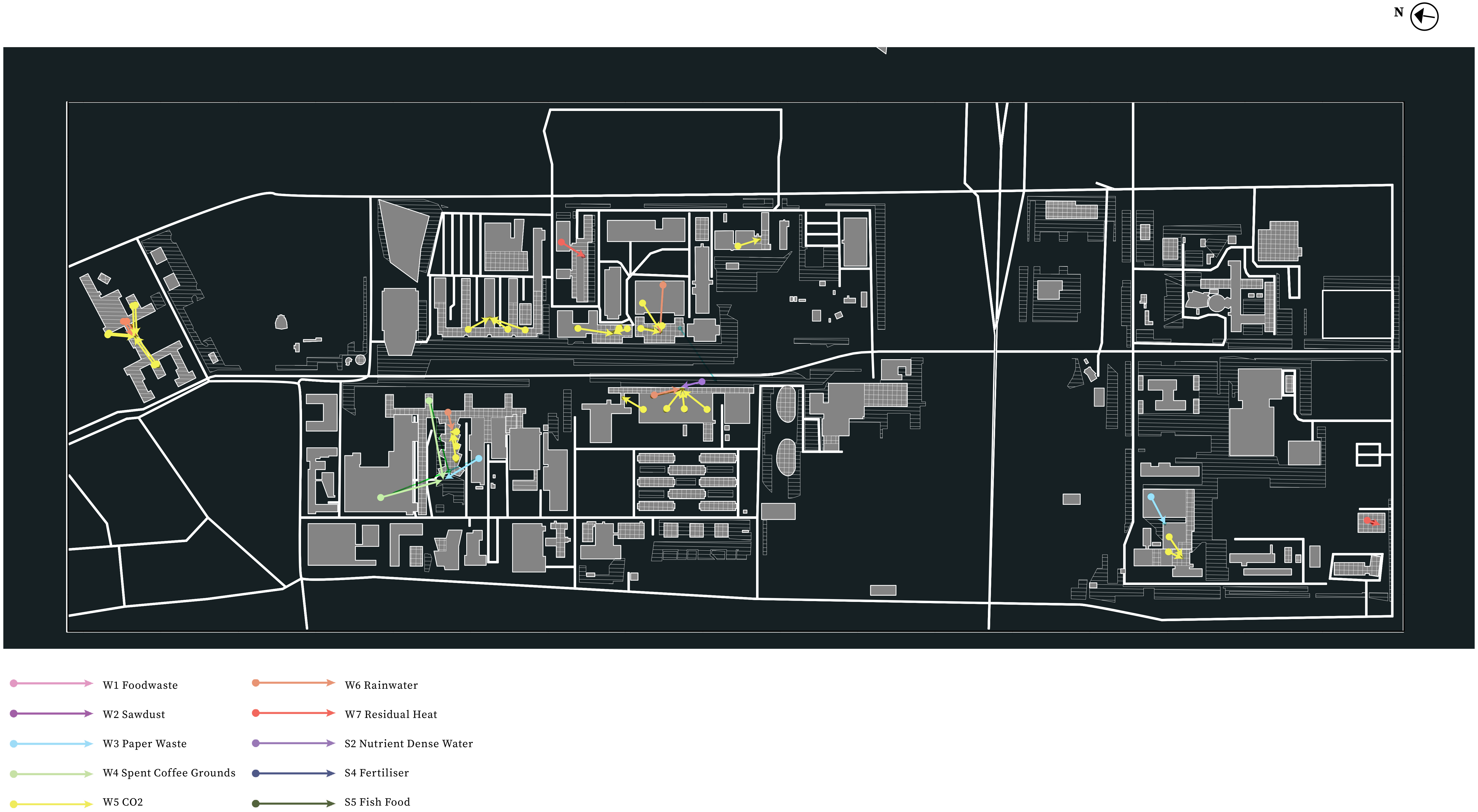
🟧 UF6 Raised Beds

🟩 UF7 Deep Water Culture

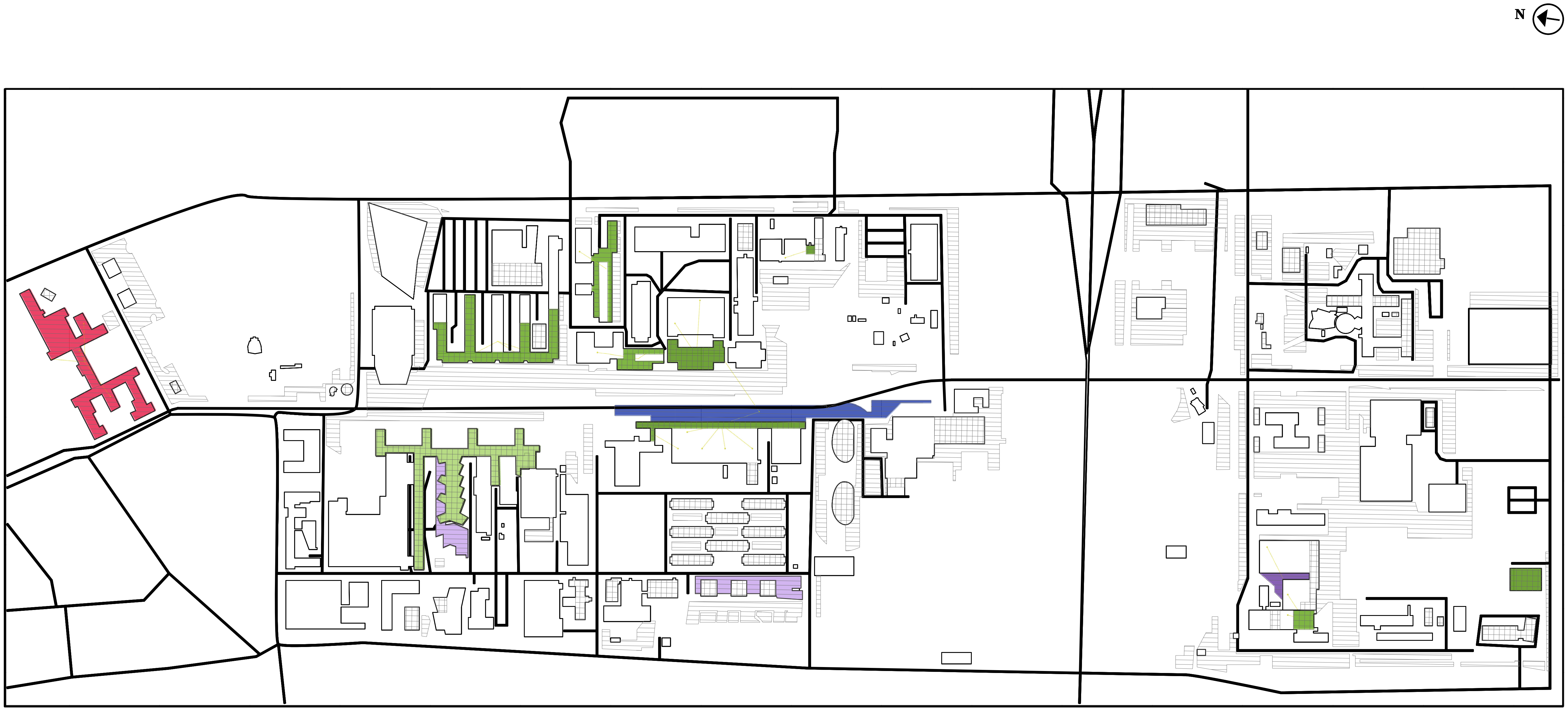
🟥 UF8 Plant Factory

🟨 UF9 Aeroponics

F.4 Stage 2 Connections

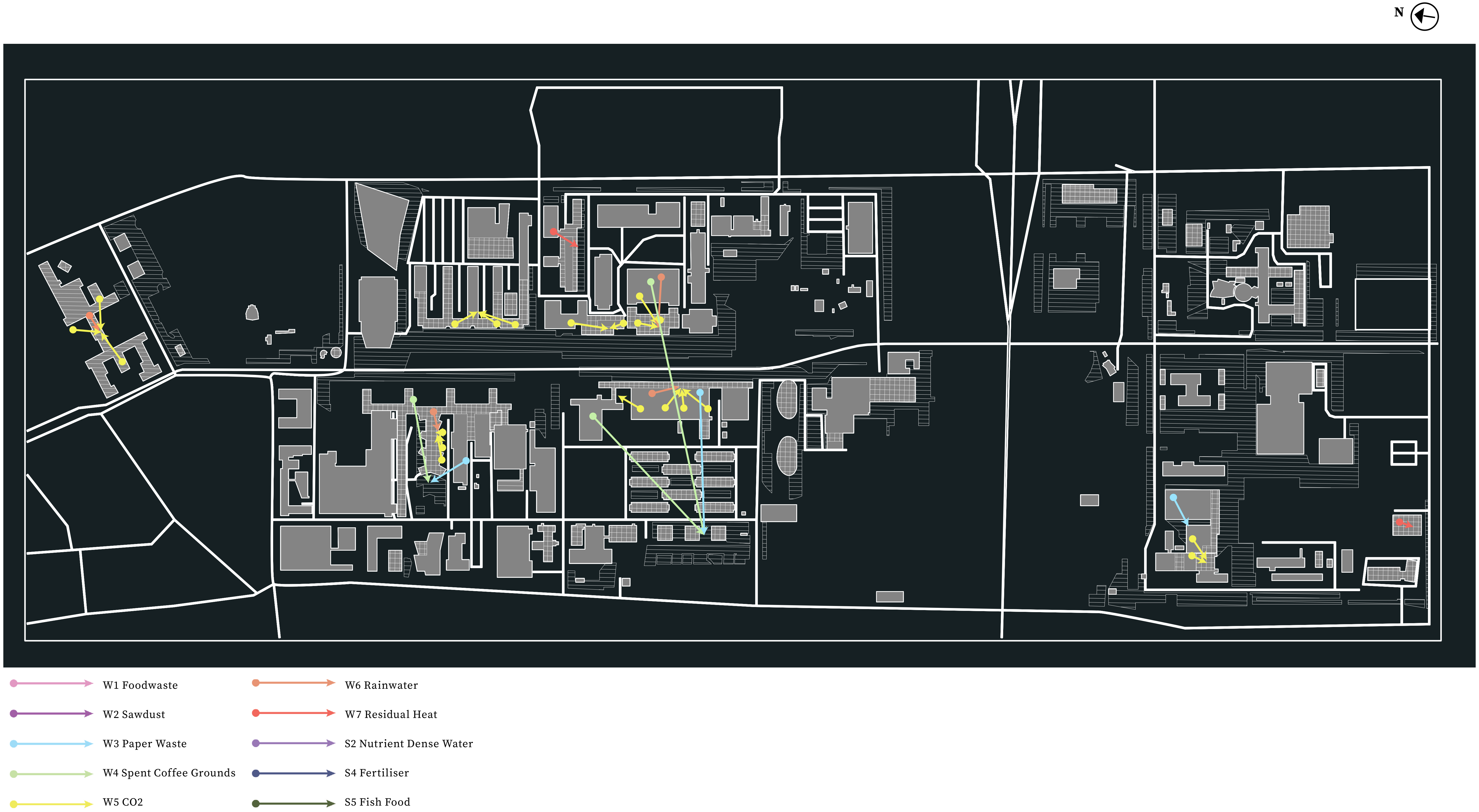


F.5 Stage 3 Assigned Systems

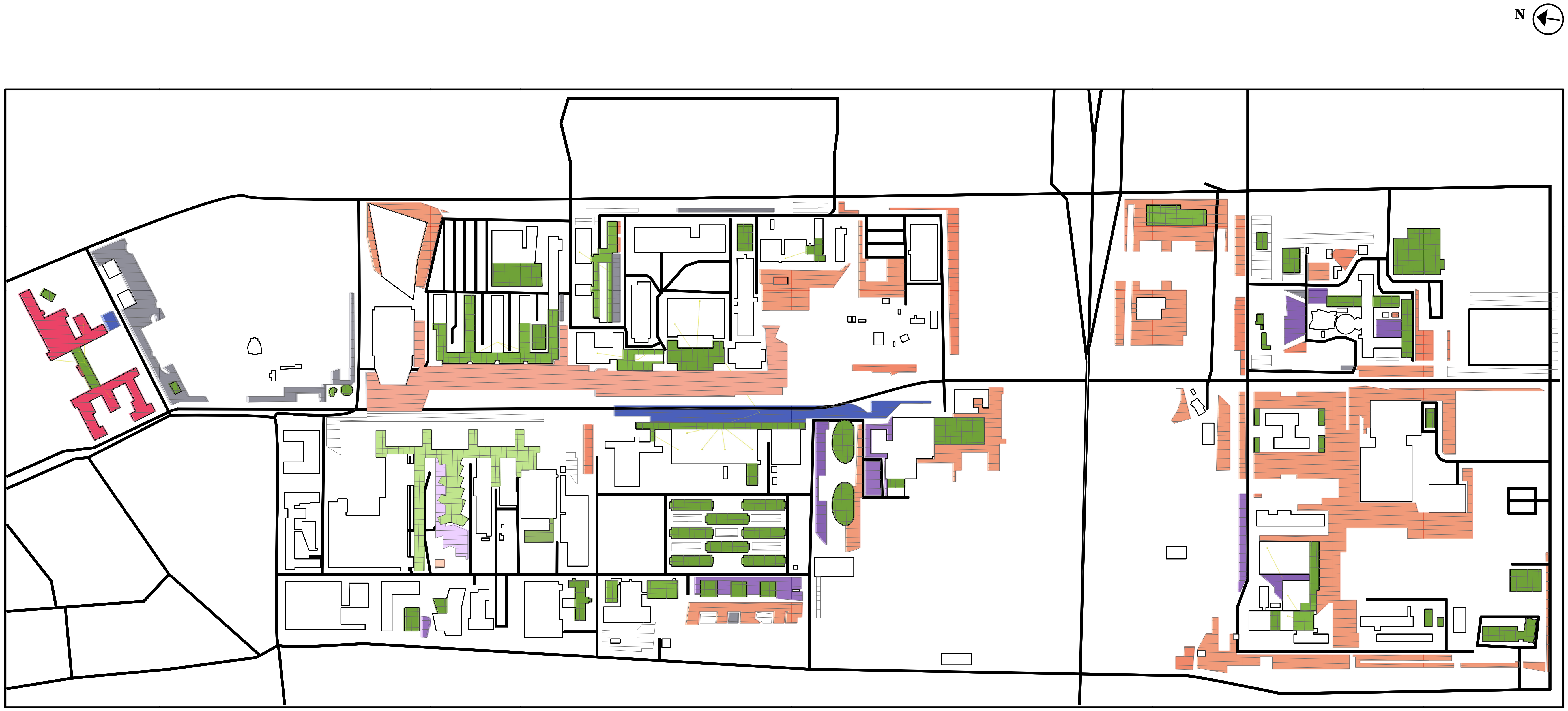


- ⬜ UF1 Vermiculture
- 🟦 UF2 Aquaculture
- 🟪 UF3 Mushroom Farm
- 🟩 UF4 NFT
- 🟩 UF5 Media Beds
- 🟧 UF6 Raised Beds
- 🟩 UF7 Deep Water Culture
- 🟥 UF8 Plant Factory
- 🟨 UF9 Aeroponics

F.6 Stage 3 Connections



F.7 Stage 4 Assigned Systems



■ UF1 Vermiculture

■ UF2 Aquaculture

■ UF3 Mushroom Farm

■ UF4 NFT

■ UF5 Media Beds

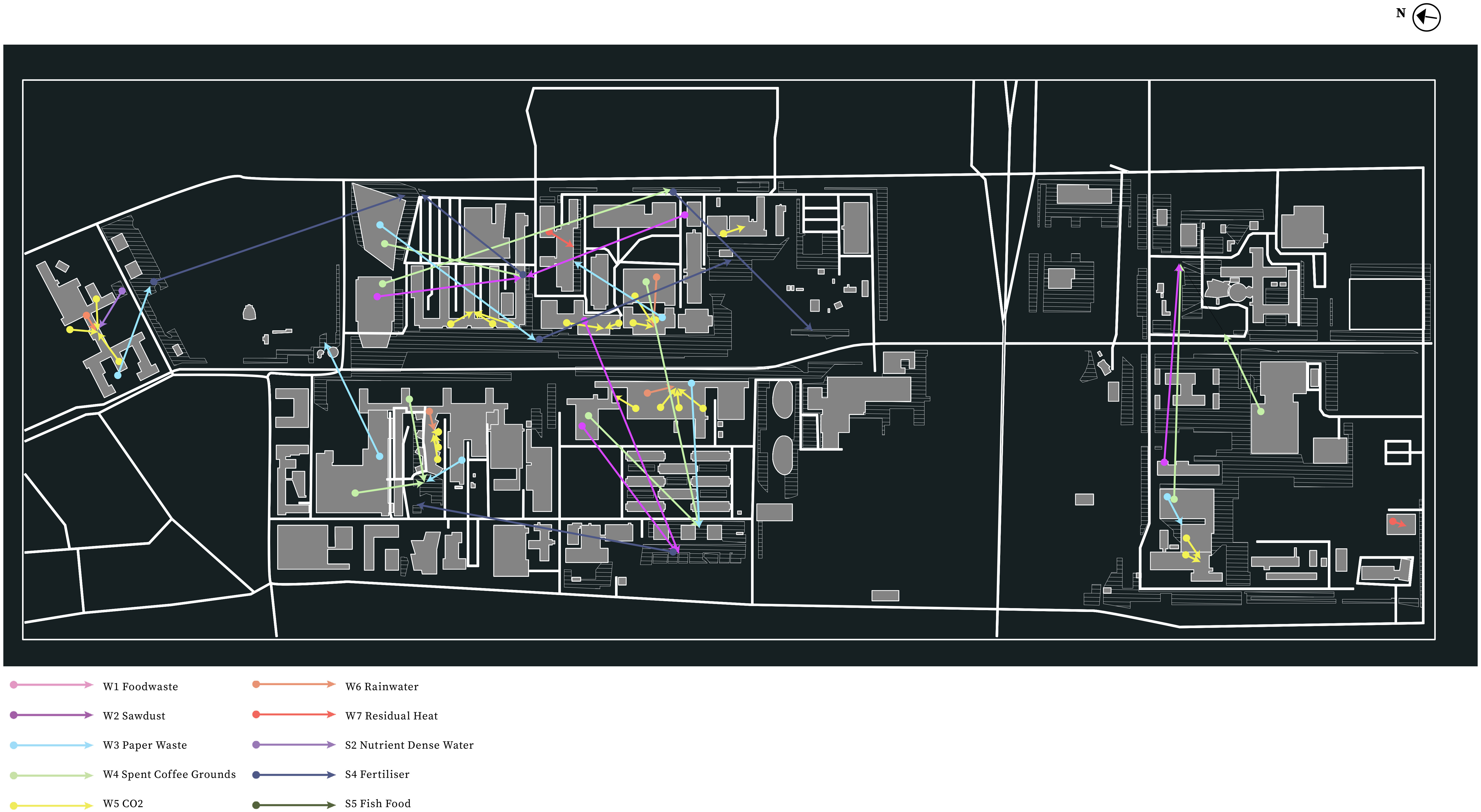
■ UF6 Raised Beds

■ UF7 Deep Water Culture

■ UF8 Plant Factory

■ UF9 Aeroponics

F.8 Stage 4 Connections



F.9 Statistics

TU DELFT YIELDS

	Size (m2)	System	Symbiosis Rate	Which stage did the space get occupied?	Produce	Produce 2	Yield (kg/ year)	Yield 2 (kg/ year)
V0	31174	UF6	0	4	small veg	big veg	389675	84169.8
V1	23824	UF1	1	4	worm		116022.88	
V10	4720	UF6	0	4	small veg	big veg	59000	12744
V100	373	UF2	1	4	fish		5595	
V102	367	UF4	0	4	small veg		36663.3	
V103	360	UF4	0	4	small veg		35964	
V106	329	UF4	0	4	small veg		32867.1	
V107	319	UF6	0	4	small veg	big veg	3987.5	861.3
V108	315	UF4	0	4	small veg		31468.5	
V109	307	UF6	0	4	small veg	big veg	3837.5	828.9
V11	4436	UF6	0	4	small veg	big veg	55450	11977.2
V110	306	UF4	0	4	small veg		30569.4	
V112	273	UF3	0.75	4	mushroom		34125	
V113	265	UF4	0	4	small veg		26473.5	
V115	237	UF4	0	4	small veg		23676.3	
V116	221	UF6	0	4	small veg	big veg	2762.5	596.7
V117	216	UF4	0	4	small veg		21578.4	
V118	216	UF4	0	4	small veg		21578.4	
V12	3912	UF4	0	4	small veg		390808.8	
V120	202	UF6	0	4	small veg	big veg	2525	545.4
V121	199	UF4	0	4	small veg		19880.1	
V122	192	UF4	0	4	small veg		19180.8	
V123	191	UF1	1	4	worm		930.17	
V124	185	UF1	1	4	worm		900.95	
V125	177	UF4	0	4	small veg		17682.3	
V126	177	UF1	1	4	worm		861.99	
V127	166	UF4	0	4	small veg		16583.4	
V128	158	UF4	0	4	small veg		15784.2	
V129	149	UF4	0	4	small veg		14885.1	
V130	147	UF6	0	4	small veg	big veg	1837.5	396.9
V131	137	UF4	0.333333333	1	small veg		13686.3	
V132	137	UF6	0	4	small veg	big veg	1712.5	369.9
V133	133	UF4	0	4	small veg		13286.7	
V134	107	UF1	1	4	worm		521.09	
V135	105	UF4	0	4	small veg		10489.5	
V136	103	UF4	0.5	1	small veg		10289.7	
V137	84	UF6	0	4	small veg	big veg	1050	226.8
V138	83	UF4	0	4	small veg		8291.7	
V139	73	UF6	0	4	small veg	big veg	912.5	197.1
V14	3795	UF6	0	4	small veg	big veg	47437.5	10246.5
V140	56	UF6	0	4	small veg	big veg	700	151.2
V141	54	UF6	0	4	small veg	big veg	675	145.8
V142	47	UF6	0	4	small veg	big veg	587.5	126.9
V144	22	UF4	0	4	small veg		2197.8	
V145	17	UF6	0	4	small veg	big veg	212.5	45.9
V15	3414	UF6	0	4	small veg	big veg	42675	9217.8
V16	3321	UF6	0	4	small veg	big veg	41512.5	8966.7
V17	3035	UF6	0	4	small veg	big veg	37937.5	8194.5
V18	2707	UF6	0	4	small veg	big veg	33837.5	7308.9
V19	2570	UF4	0	4	small veg		256743	
V2	9665	UF4	0.666666667	1	small veg		965533.5	
V21	2541	UF3	0.5	3	mushroom		317625	
V22	2440	UF4	0.666666667	1	small veg		243756	
V23	2245	UF3	0.5	2	mushroom		280625	
V24	2167	UF4	0.666666667	1	small veg		216483.3	
V25	2144	UF4	0	4	small veg		214185.6	
V26	2139	UF4	0	4	small veg		213686.1	
V27	2111	UF4	0.333333333	1	small veg		210888.9	
V28	2093	UF1	1	4	worm		10192.91	
V29	1954	UF3	0.5	4	mushroom		244250	
V3	8429	UF8	1	1	small veg		736694.6	
V30	1942	UF6	0	4	small veg	big veg	24275	5243.4
V31	1921	UF6	0	4	small veg	big veg	24012.5	5186.7
V32	1665	UF6	0	4	small veg	big veg	20812.5	4495.5
V33	1658	UF6	0	4	small veg	big veg	20725	4476.6
V35	1597	UF6	0	4	small veg	big veg	19962.5	4311.9
V36	1578	UF4	0	4	small veg		157642.2	
V37	1572	UF3	0.4	4	mushroom		196500	
V38	1501	UF4	0	4	small veg		149949.9	
V39	1501	UF4	0	4	small veg		149949.9	
V4	8272	UF6	0	4	small veg	big veg	103400	22334.4
V40	1354	UF4	0.333333333	1	small veg		135264.6	
V41	1340	UF3	0.4	4	mushroom		167500	
V42	1288	UF6	0	4	small veg	big veg	16100	3477.6
V43	1275	UF6	0	4	small veg	big veg	15937.5	3442.5
V44	1233	UF4	0.333333333	1	small veg		123176.7	
V46	1153	UF6	0	4	small veg	big veg	14412.5	3113.1
V47	1148	UF4	0	4	small veg		114685.2	
V48	1130	UF6	0	4	small veg	big veg	14125	3051
V49	1091	UF6	0	4	small veg	big veg	13637.5	2945.7
V5	8061	UF2	1	1	fish		120915	
V50	1060	UF3	0.5	4	mushroom		132500	
V51	1055	UF4	0	4	small veg		105394.5	
V52	1044	UF4	0	4	small veg		104295.6	
V53	1034	UF1	1	4	worm		5035.58	
V54	950	UF4	0	4	small veg		94905	
V55	937	UF3	0.5	4	mushroom		117125	
V56	886	UF3	0.4	4	mushroom		110750	
V57	886	UF4	0	4	small veg		88511.4	
V58	886	UF4	0	4	small veg		88511.4	
V59	886	UF4	0	4	small veg		88511.4	
V6	6439	UF1	1	4	worm		31357.93	
V60	886	UF4	0	4	small veg		88511.4	

	Size (m2)	System	Symbiosis Rate	Which stage did the space get occupied?	Produce	Produce 2	Yield (kg/ year)	Yield 2 (kg/ year)
V61	886	UF4	0	4	small veg		88511.4	
V62	886	UF4	0	4	small veg		88511.4	
V63	886	UF4	0	4	small veg		88511.4	
V64	886	UF4	0	4	small veg		88511.4	
V65	884	UF6	0	4	small veg	big veg	11050	2386.8
V66	871	UF3	0.333333333	1	mushroom		108875	
V67	849	UF6	0	4	small veg	big veg	10612.5	2292.3
V68	817	UF4	0.333333333	4	small veg		81618.3	
V69	816	UF4	0	4	small veg		81518.4	
V71	782	UF4	0	4	small veg		78121.8	
V72	741	UF4	0.333333333	1	small veg		74025.9	
V73	735	UF1	1	4	worm		3579.45	
V74	734	UF4	0	4	small veg		73326.6	
V75	731	UF4	0	4	small veg		73026.9	
V76	694	UF6	0	4	small veg	big veg	8675	1873.8
V78	641	UF4	0	4	small veg		64035.9	
V79	636	UF4	0.333333333	4	small veg		63536.4	
V8	5436	UF4	0.333333333	1	small veg		543056.4	
V80	619	UF6	0	4	small veg	big veg	7737.5	1671.3
V81	603	UF6	0	4	small veg	big veg	7537.5	1628.1
V82	566	UF6	0	4	small veg	big veg	7075	1528.2
V84	559	UF6	0	4	small veg	big veg	6987.5	1509.3
V85	549	UF4	0	4	small veg		54845.1	
V86	545	UF3	0.4	4	mushroom		68125	
V87	513	UF4	0	4	small veg		51248.7	
V88	506	UF4	0	4	small veg		50549.4	
V89	501	UF6	0	4	small veg	big veg	6262.5	1352.7
V9	4880	UF6	0	4	small veg	big veg	61000	13176
V91	475	UF4	0	4	small veg		47452.5	
V92	475	UF4	0	4	small veg		47452.5	
V94	443	UF4	0	4	small veg		44255.7	
V97	402	UF4	0	4	small veg		40159.8	

	Area (m2)	Area (hectares)		
TOTAL	221817	22.1817		
	#spaces	Area (hectares)	Area (m2)	Percentage %
small veg	103	16.4374	164374	74.1034276
big veg	40	9.1413	91413	41.21099826
worm	9	3.4785	34785	15.68184585
mushroom	11	1.4224	14224	6.412493181
fish	2	0.8434	8434	3.802233373

Produce	Number	Annual Production (kg)	Daily Production (kg)
worm	677611800	169402.95	464.1176712
fish	506040	126510	346.6027397
mushroom	11853333	1778000	4871.232877
small veg	59472170	8326103.9	22811.24356
big veg	76177500	1142662.5	3130.582192

Daily Production of Leafy Greens	22811.24356
Recommended Daily Fruit & Vegetable Consumption (kg)	0.25 (RIVM, 2017)
People #	91244.97425
Delf Population	103581 (Statistiek, n.d.)

Stage	Symbiosis Rate % of Decisions
Stage 1 (100 m)	52.56410256
Stage 2 (200 m)	50
Stage 3 (500 m)	50
Stage 4 (500 m + occupy all)	13.1969697
Total	17.88

TU DELFT WASTE USE

	Used Quantity	Left Quantity	Used Percentage
Food Waste (W1)	16967	1955	89.66811119
Sawdust (W2)	0	3360	0
Paper Waste (W3)	766093	46508	94.27664992
Coffee Waste (W4)	106308	54590	66.07167274
CO2 (W5)	2013	1966	50.59060065
Rainwater (W6)	41050300	8066500	83.5769024
Residual Heat (W7)	10357320	0	100

Appendix G Ward 7 Analysis & Decisions

APPENDIX G WARD 7 ANALYSIS & DECISIONS

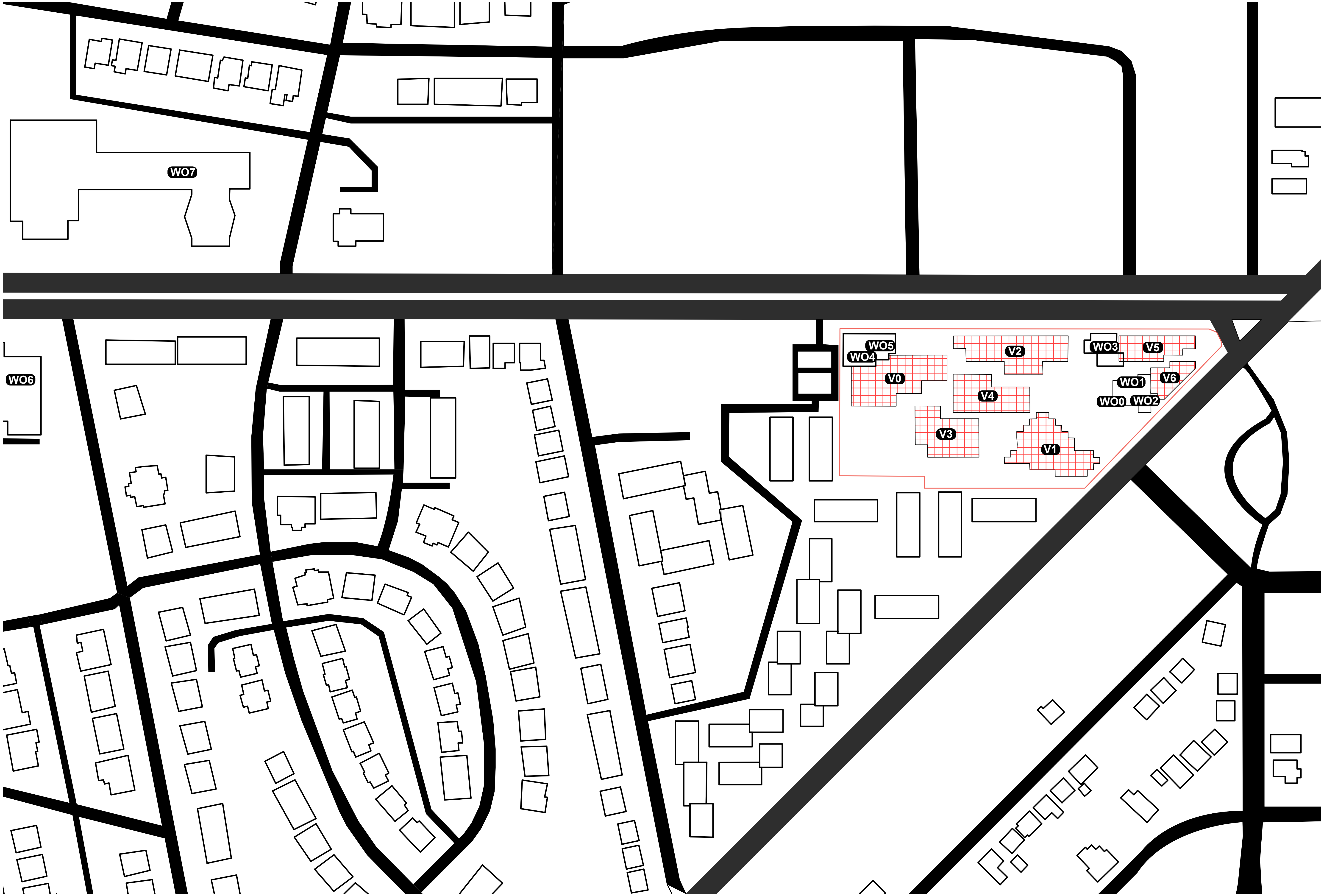
WARD 7 VACANT SPACE NODES

Identifier	Coordinates	Size	Building	Location	Orientation	Tag	Node Type
V0	{-803.904692, 34.790488, 0}	916.723679	V0	outside	S	V0	vacant space
V1	{-724.997122, -1.092597, 0}	833.367616	V1	outside	S	V1	vacant space
V2	{-743.096854, 48.585595, 0}	831.022338	V2	outside	S	V2	vacant space
V3	{-778.018835, 6.722332, 0}	668.658545	V3	outside	S	V3	vacant space
V4	{-757.014994, 25.984151, 0}	625.125626	V4	outside	S	V4	vacant space
V5	{-673.392951, 50.552091, 0}	427.509633	V5	outside	E	V5	vacant space
V6	{-664.923643, 35.208772, 0}	267.734853	V6	outside	NE	V6	vacant space

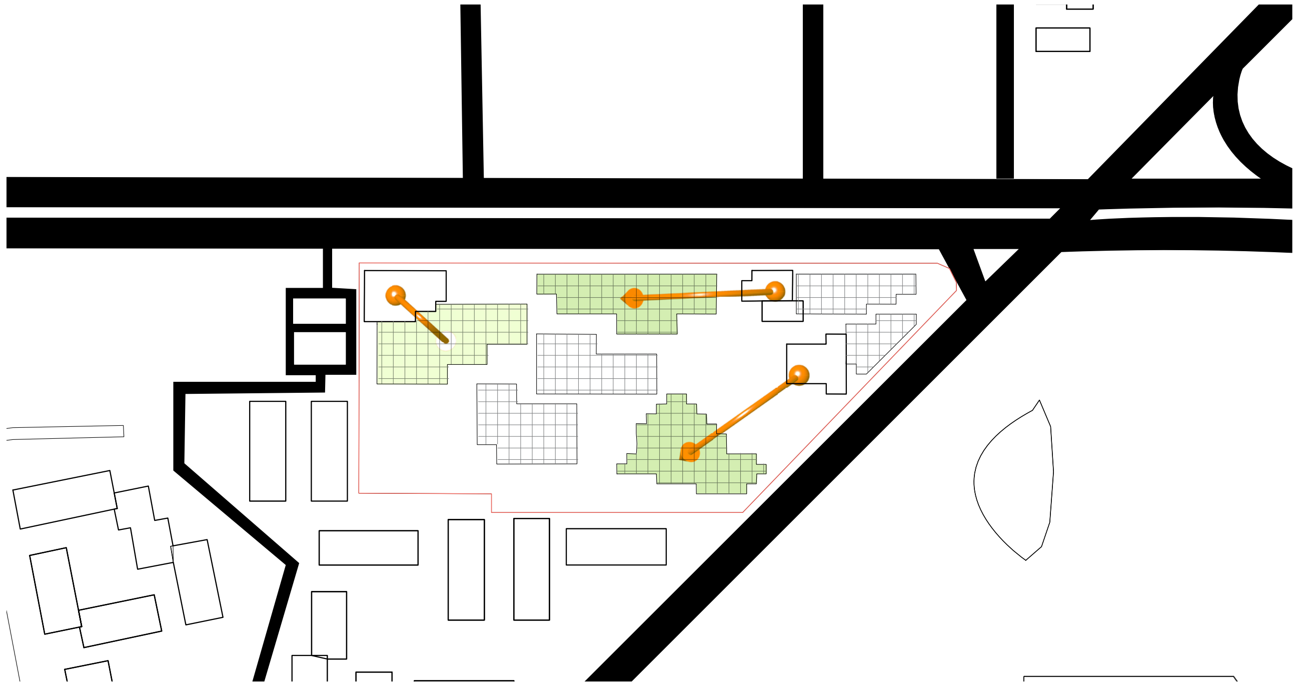
WARD 7 WASTE SOURCE NODES

Identifier	Coordinates	Source	Waste Type	Quantity	Tag	Node Type	Waste
WO0	{-689.802413, 23.741236, 0}	cafe	W6	3	WO0	waste	rainwater
WO1	{-683.915012, 28.756529, 0}	cafe	W1	3	WO1	waste	food
WO2	{-677.172337, 23.553906, 0}	cafe	W4	3	WO2	waste	coffee
WO3	{-697.587895, 50.744049, 0}	community	W6	3	WO3	waste	rainwater
WO4	{-820.161616, 49.441107, 0}	makers	W6	3	WO4	waste	rainwater
WO5	{-811.057505, 51.355828, 0}	makers	W2	3	WO5	waste	sawdust
WO6	{-1266.789423, 45.775932, 0}	school1	W3	3	WO6	waste	paper
WO7	{-1201.863458, 141.943154, 0}	school1	W3	3	WO7	waste	paper

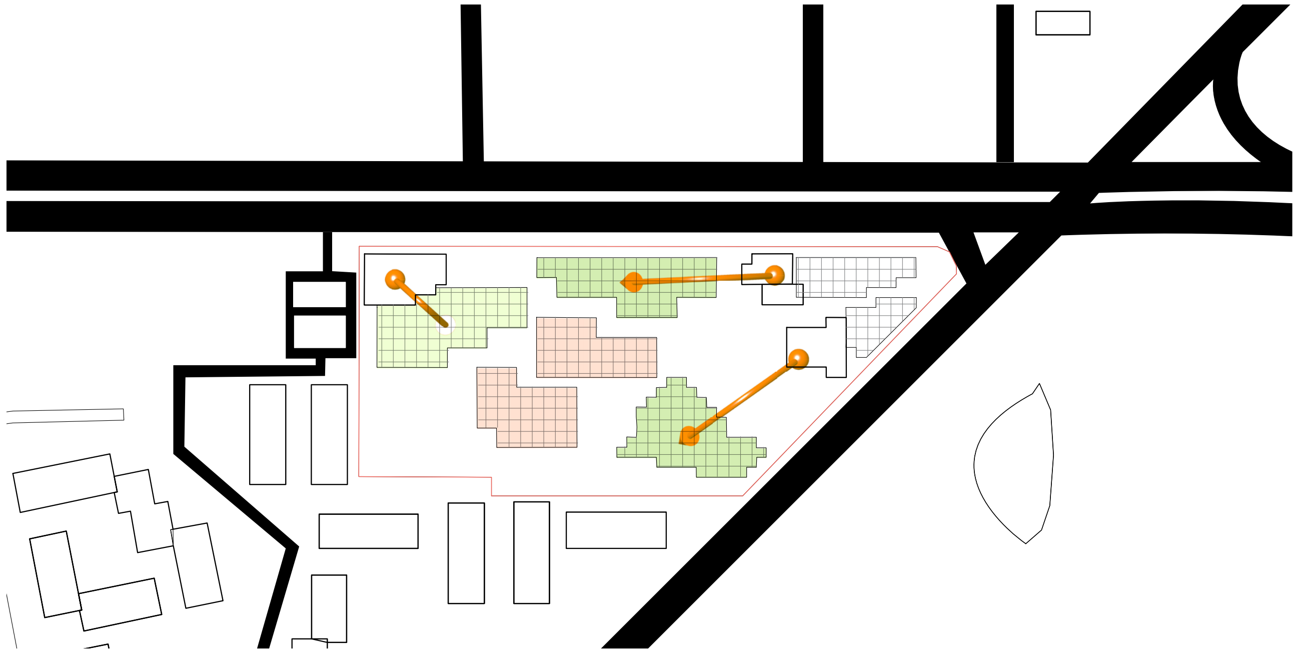
G.1 Vacant Space & Waste Output Nodes



G.2 Stage 1



G.3 Stage 2



G.4 UGC Yields

	size	system	produce	produce2	yield	yield2
V0	916	UF4	small veg		91508.4	
V1	833	UF4	small veg		83216.7	
V2	831	UF4	small veg		83016.9	
V3	668	UF6	small veg	big veg	8350	1803.6
V4	625	UF6	small veg	big veg	7812.5	1687.5

	Number	Annual Produce (kg)	DailyProduce (kg)
worm	0	0	0
fish	0	0	0
mushroom	0	0	0
small veg	1956460	273904.5	750.4232877
big veg	1077500	16162.5	44.28082192

Daily Production of Leafy Greens (kg)	750.4232877
Recommended Daily Fruit & Vegetable Consumption (kg)	0.25 (RIVM, 2017)
People #	3002
Ward 7 Population	77456 (DC Health Matters, n.d.)

Appendix H Technical Details

see Foodcycle’s Github for the scripts

APPENDIX H TECHNICAL DETAILS

H.1 SCRIPT INCLUDING DATA PROCESSING STAGE 1 AND DATA EXPORT

- IMPORT TOOLS -

In [ ]:

```
#Import Tools
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
import json
import requests
import openpyxl
```

In [ ]:

```
# Gives adjacency dictionary (not mine)
edge_dict={}
def create_edge_dict(graph):
    for i, n in G.adjacency():
        # print("i is",i)
        # print("n is",n)
        edge_dict[i] = list(n)
    return edge_dict

# Gives duplicate items in a list (not mine)
def list_duplicates(seq):
  seen = set()
  seen_add = seen.add
  # adds all elements it doesn't know yet to seen and all other to seen_twice
  seen_twice = set( x for x in seq if x in seen or seen_add(x) )
  # turn the set into a list (as requested)
  return list( seen_twice )


#(https://www.geeksforgeeks.org/python-merge-two-lists-into-list-of-tuples/)
def merge(list1, list2):

    merged_list = []
    for i in range(max((len(list1), len(list2)))):

        while True:
            try:
                tup = (list1[i], list2[i])
            except IndexError:
                if len(list1) > len(list2):
                    list2.append('')
                    tup = (list1[i], list2[i])
                elif len(list1) < len(list2):
                    list1.append('')
                    tup = (list1[i], list2[i])
                continue

            merged_list.append(tup)
            break
    return merged_list


# function to check whether the list is empty or not
def is_list_empty(list):
    # checking the length
    if len(list) == 0:
        # returning true as length is 0
        return True
    # returning false as length is greater than 0
    return False
```

Stage 0 | Preperation of Data

- IMPORT DATA & PREPARE FOR PROCESSING -

In [ ]:

```
# Retrieve data regarding vacant spaces from excel file
from openpyxl import load_workbook
workbook = load_workbook(filename="Node_Information_TU.xlsx")
workbook.sheetnames
# worksheet for vacant spaces
sheet1 = workbook.worksheets[0]
# worksheet for waste sources
sheet2 = workbook.worksheets[1]

# Create dictionary based on excel worksheet
vacant_spaces={}
for value in sheet1.iter_rows(min_row=2, values_only=True):
    for index,item in enumerate(value):
        vacant_spaces[value[0]]={}
        vacant_spaces[value[0]]["location"]=value[1]
        vacant_spaces[value[0]]["tag"]=value[6]
        vacant_spaces[value[0]]["building"]=value[3]
        vacant_spaces[value[0]]["size_real"]=int(value[2])
        vacant_spaces[value[0]]["loc_building"]=value[4]
        vacant_spaces[value[0]]["orientation"]=value[5]
        vacant_spaces[value[0]]["node_type"]=value[7]

# Interpret Data
#Rules:
#structure (roof:1,ground:3,outside:3,basement:3,intermediate:2)
#solar (roof:3,
#        basement:1
#        ground & S,SW,SE:3,
#        ground $ E,W:2
#        ground $ N,NE,NW:1
#        outside & S,SW,SE:3,
#        outside $ E,W:2
#        outside $ N,NE,NW:1
#        intermediate & S,SW,SE:3,
#        intermediate & E,W:2
#        intermediate & N,NE,NW:1
#size (0-100 size:1)
#     (101-500 size:2)
#     (500-... size:3)

# Simplify vacant space characteristics
for space in vacant_spaces:
    if vacant_spaces[space]["loc_building"]=="roof":
        vacant_spaces[space]["structure"]=1
        vacant_spaces[space]["solar"]=3
    if vacant_spaces[space]["loc_building"]=="basement":
        vacant_spaces[space]["structure"]=3
        vacant_spaces[space]["solar"]=1
```

```
    if (vacant_spaces[space]["loc_building"]=="ground") and (((vacant_spaces[space]
["orientation"])=="S") or ((vacant_spaces[space]["orientation"])=="SE") or
((vacant_spaces[space]["orientation"])=="SW")):
        vacant_spaces[space]["solar"]=3
        vacant_spaces[space]["structure"]=3
    if (vacant_spaces[space]["loc_building"]=="ground") and (((vacant_spaces[space]
["orientation"])=="N") or ((vacant_spaces[space]["orientation"])=="NE") or
((vacant_spaces[space]["orientation"])=="NW")):
        vacant_spaces[space]["solar"]=1
        vacant_spaces[space]["structure"]=3
    if (vacant_spaces[space]["loc_building"]=="ground") and (((vacant_spaces[space]
["orientation"])=="E") or ((vacant_spaces[space]["orientation"])=="W")):
        vacant_spaces[space]["solar"]=2
        vacant_spaces[space]["structure"]=3
    if (vacant_spaces[space]["loc_building"]=="intermediate") and (((vacant_spaces[space]
["orientation"])=="S") or ((vacant_spaces[space]["orientation"])=="SE") or
((vacant_spaces[space]["orientation"])=="SW")):
        vacant_spaces[space]["solar"]=3
        vacant_spaces[space]["structure"]=2
    if (vacant_spaces[space]["loc_building"]=="intermediate") and (((vacant_spaces[space]
["orientation"])=="N") or ((vacant_spaces[space]["orientation"])=="NE") or
((vacant_spaces[space]["orientation"])=="NW")):
        vacant_spaces[space]["solar"]=1
        vacant_spaces[space]["structure"]=2
    if (vacant_spaces[space]["loc_building"]=="intermediate") and (((vacant_spaces[space]
["orientation"])=="E") or ((vacant_spaces[space]["orientation"])=="W")):
        vacant_spaces[space]["solar"]=2
        vacant_spaces[space]["structure"]=2
    if (vacant_spaces[space]["loc_building"]=="outside") and (((vacant_spaces[space]
["orientation"])=="S") or ((vacant_spaces[space]["orientation"])=="SE") or
((vacant_spaces[space]["orientation"])=="SW")):
        vacant_spaces[space]["solar"]=3
        vacant_spaces[space]["structure"]=3
    if (vacant_spaces[space]["loc_building"]=="outside") and (((vacant_spaces[space]
["orientation"])=="N") or ((vacant_spaces[space]["orientation"])=="NE") or
((vacant_spaces[space]["orientation"])=="NW")):
        vacant_spaces[space]["solar"]=1
        vacant_spaces[space]["structure"]=3
    if (vacant_spaces[space]["loc_building"]=="outside") and (((vacant_spaces[space]
["orientation"])=="E") or ((vacant_spaces[space]["orientation"])=="W")):
        vacant_spaces[space]["solar"]=2
        vacant_spaces[space]["structure"]=3

# Simplify vacant spaces' sizes based on predefined ranges
for space in vacant_spaces:
    if vacant_spaces[space]["size_real"]<=200:
        vacant_spaces[space]["size"]=1
    if (vacant_spaces[space]["size_real"]>200) and (vacant_spaces[space]
["size_real"]<=1000) :
        vacant_spaces[space]["size"]=2
    if (vacant_spaces[space]["size_real"]>1000):
        vacant_spaces[space]["size"]=3

# Remove orientation,loc_building,size_real since these will not be used anymore
for space in vacant_spaces:
    vacant_spaces[space].pop("orientation")
    vacant_spaces[space].pop("size_real")
    vacant_spaces[space].pop("loc_building")

# Retrieve waste sources
# Create dictionary based on excel worksheet
wastes={}
for value in sheet2.iter_rows(min_row=2, values_only=True):
    for index,item in enumerate(value):
            wastes[value[0]]={}
            wastes[value[0]]["location"]=value[1]
            wastes[value[0]]["tag"]=value[5]
            wastes[value[0]]["type"]=value[3]
```

In [ ]:

```
# Retrieve connections between vacant spaces within radius=x
url1="https://raw.githubusercontent.com/erengozdeanil/UF-
DecisionMaker/main/Edges_vacant100.txt"
resp1 = requests.get(url1)
edges1 = json.loads(resp1.text)
#converts nested lists into a list of tuples
nearby_space100  = [tuple(i) for i in edges1]

# Retrieve connections between vacant spaces and waste sources within radius=x with
identifiers
url1="https://raw.githubusercontent.com/erengozdeanil/UF-
DecisionMaker/main/Edges_try1.txt"
resp1 = requests.get(url1)
edges1 = json.loads(resp1.text)
#converts nested lists into a list of tuples
nearby_waste100 = [tuple(i) for i in edges1]
```

In [ ]:

```
# In the excel sheet there were unnecessary/empty waste output points
# Remove empty points
for waste in wastes:
    for couple in nearby_waste100:
        if wastes[waste]["type"]=="None":
            if waste in couple:
                print(couple, "removed")
                nearby_waste100.remove(couple)
                (couple,"removed")
```

In [ ]:

```
# Retrieve node information regarding waste sources
workbook = load_workbook(filename="Node_Information_TU.xlsx")
workbook.sheetnames
sheet1 = workbook.worksheets[0]
sheet2 = workbook.worksheets[1]
wastes={}
for value in sheet2.iter_rows(min_row=2, values_only=True):
    for index,item in enumerate(value):
        if value[4]!=0:
            wastes[value[0]]={}
            wastes[value[0]]["location"]=value[1]
            wastes[value[0]]["source"]=value[2]
            wastes[value[0]]["tag"]=value[5]
```

```
            wastes[value[0]]["type"]=value[3]
            wastes[value[0]]["quantity"]=int(value[4])
            wastes[value[0]]["node_type"]="waste"


# Assign ranges to quantities
for waste in wastes:
    if wastes[waste]["type"]=="W1":
        if wastes[waste]["quantity"]<= 183705:
            wastes[waste]["size"]=1
        if (wastes[waste]["quantity"]>183705) and (wastes[waste]["quantity"]<=918523):
            wastes[waste]["size"]=2
        if wastes[waste]["quantity"]>918523:
            wastes[waste]["size"]=3
    if wastes[waste]["type"]=="W2":
        if wastes[waste]["quantity"]<= 67500:
            wastes[waste]["size"]=1
        if (wastes[waste]["quantity"]>67500) and (wastes[waste]["quantity"]<=337500):
            wastes[waste]["size"]=2
        if wastes[waste]["quantity"]>337500:
            wastes[waste]["size"]=3
    if wastes[waste]["type"]=="W3":
        if wastes[waste]["quantity"]<= 11813:
            wastes[waste]["size"]=1
        if (wastes[waste]["quantity"]>11813) and (wastes[waste]["quantity"]<=59063):
            wastes[waste]["size"]=2
        if wastes[waste]["quantity"]>59063:
            wastes[waste]["size"]=3
    if wastes[waste]["type"]=="W4":
        if wastes[waste]["quantity"]<= 28200:
            wastes[waste]["size"]=1
        if (wastes[waste]["quantity"]>28200) and (wastes[waste]["quantity"]<=141000):
            wastes[waste]["size"]=2
        if wastes[waste]["quantity"]>141000:
            wastes[waste]["size"]=3
    if wastes[waste]["type"]=="W5":
        if wastes[waste]["quantity"]<= 19342:
            wastes[waste]["size"]=1
        if (wastes[waste]["quantity"]>19342) and (wastes[waste]["quantity"]<=96710):
            wastes[waste]["size"]=2
        if wastes[waste]["quantity"]>96710:
            wastes[waste]["size"]=3
    if wastes[waste]["type"]=="W6":
        if wastes[waste]["quantity"]<= 1560000:
            wastes[waste]["size"]=1
        if (wastes[waste]["quantity"]>1560000) and (wastes[waste]["quantity"]<=7800000):
            wastes[waste]["size"]=2
        if wastes[waste]["quantity"]>7800000:
            wastes[waste]["size"]=3
    if wastes[waste]["type"]=="W7":
        if wastes[waste]["quantity"]<= 200000:
            wastes[waste]["size"]=1
        if (wastes[waste]["quantity"]>200000) and (wastes[waste]["quantity"]<=1000000):
            wastes[waste]["size"]=2
        if wastes[waste]["quantity"]>1000000:
            wastes[waste]["size"]=3
```

```
# Remove real quantitis from the list since we will not use it again
for waste in wastes:
    wastes[waste].pop("quantity")
```

In [ ]:

```
#Dictionary of urban farming systems
    # UF1: Vermiculture, UF2: Aquaculture, UF3: Mushroom, UF4: NFT, UF5: Medai Beds, UF6:
Raised Beds, UF7: Water Culture, UF8: Plant Factory, UF9: Aeroponics
    # "S" : supplementary system, "F" : food production system
    # 3 : high, 2 : medium, 1 : low, 0 : none
uf_systems = {
"UF1":{"tag":"UF1","type":"S","weight":3,"solar":1,"in":
["W1","W2","W3","W6"],"supplement":None,"out":["S4","S5"]},
"UF2":{"tag":"UF2","type":"SF","weight":3,"solar":2,"in":["W7"],"supplement":["S5"],"out":
["O4","S2"]},
"UF3":{"tag":"UF3","type":"F","weight":2,"solar":1,"in":
["W2","W3","W4","W6"],"supplement":None,"out":["O3","S4"]},
"UF4":{"tag":"UF4","type":"F","weight":1,"solar":3,"in":["W5","W6","W7"],"supplement":
["S2"],"out":["O1","W1"]},
"UF5":{"tag":"UF5","type":"F","weight":1,"solar":3,"in":["W5","W6","W7"],"supplement":
["S2"],"out":["O1","O2","W1"]},
"UF6":{"tag":"UF6","type":"F","weight":3,"solar":3,"in":["W6"],"supplement":["S4"],"out":
["O1","O2","W1"]},
"UF7":{"tag":"UF7","type":"F","weight":3,"solar":3,"in":["W5","W6","W7"],"supplement":
["S2"],"out":["O1","W1"]},
"UF8":{"tag":"UF8","type":"F","weight":3,"solar":1,"in":["W5","W6"],"supplement":
["S2"],"out":["O1","W1","W7"]},
"UF9":{"tag":"UF9","type":"F","weight":1,"solar":3,"in":["W5","W6","W7"],"supplement":
["S2"],"out":["O1","W1"]}
}

critical_items=["W1","W2","W3", "W4"]
non_critical_items=["W5","W6","W7"]
```

Stage 1 | Search Radius = 100 m

Data Processing

In [ ]:

- STAGE 1 -

```
# Make a dictionary of vacant spaces and potential uf systems based on structural capacity
& solar exposure
# Rule 1: Solar exposure of space == Solar exposure demand of system
#         Structural capacity of space >= Weight of space
#         Size of space == Quantity of waste

v_potential = {}
for k,v in vacant_spaces.items():
    uf_list=[]
    for j,y in uf_systems.items():
        if (vacant_spaces[k]["structure"])>=(uf_systems[j]["weight"]) and
(vacant_spaces[k]["solar"])==(uf_systems[j]["solar"]) and (uf_systems[j]["type"]=="F"):
            v_potential[k]={}
            uf_list.append(uf_systems[j]["tag"])
            v_potential[k]["tag"]=vacant_spaces[k]["tag"]
            v_potential[k]["UF"]=uf_list
```

```
# Rule 2: If CO2, Heat and Rainwater are not in the same building as the vacant space,
they cannot be used.
# Remove if vacant space is in a different building than the waste source (only for CO2,
Heat and Rainwater)
for couple_count in range(len(nearby_waste100)):
    for couple in nearby_waste100:
        for index,item in enumerate(couple):
            if couple[1] in wastes:
                if vacant_spaces[couple[0]]["building"]!=wastes[couple[1]]["source"]:
                    print(couple,vacant_spaces[couple[0]]["building"],wastes[couple[1]]
["source"],wastes[couple[1]]["type"])
                    if (wastes[couple[1]]["type"]=="W5") or (wastes[couple[1]]
["type"]=="W6") or (wastes[couple[1]]["type"]=="W7"):
                        if couple in nearby_waste100:
                            print(couple,wastes[couple[1]]["type"])
                            print("removed",couple,wastes[couple[1]]
["type"],vacant_spaces[couple[0]]["building"],wastes[couple[1]]
["source"],wastes[couple[1]]["type"])
                            nearby_waste100.remove(couple)
            else:
                print(couple,"not in nearby_waste100")
```

In [ ]:

```
# Draw Graph with nodes and edges with coordinates
G=nx.Graph()
for i,j in vacant_spaces.items():
    G.add_node(i)
G.add_edges_from(nearby_waste100)
nx.draw(G, with_labels=True, node_size=10)

# Create a dictionary with vacant spaces and waste outputs them
new_waste_dict = create_edge_dict(G)
waste_dict = {}
for i,k in new_waste_dict.items():
    if i in vacant_spaces:
        waste_dict[i]=k

#List of dictionaries we will use:
# print(v_potential)
# print(waste_dict)
# print(uf_systems)

# Add needed inputs for each potential system into the v_potential dictionary
v_potential_dict={}
for i,k in v_potential.items():
    v_potential_dict[i]={}
    potential_systems = v_potential[i]["UF"]
    for item in potential_systems:
       v_potential_dict[i][item] = uf_systems[item]["in"]
```

In [ ]:

```
# Searching For Inputs
# Make a dictionary of found items nearby for each potential uf system
found_dict={}
for i,k in v_potential_dict.items():
    found_dict[i]={}
    for system,demanded in k.items():
        found_dict[i][system]={}
        found_dict[i][system]["found"]={}
        found_dict[i][system]["source"]={}
        found=[]
        source=[]
        for waste in waste_dict[i]:
            if waste in wastes:
                for each in demanded:
                    available = wastes[waste]["type"]
                    source2 = wastes[waste]["tag"]
                    print(i,wastes[waste])
                    source_size = wastes[waste]["size"]
                    if each == available:
                        found.append(available)
                        source.append(source2)
                        found_dict[i][system]["found"]=found
                        found_dict[i][system]["source"]=source
```

In [ ]:

```
# Make a dictionary of missing items for each potential uf system
for i,system in found_dict.items():
    for uf in system:
        missing=[]
        found_dict[i][uf]["missing"]={}
        for x,y in uf_systems.items():
            for item in uf_systems[x]["in"]:
                if uf == x:
                    missing.append(item)
                    found_dict[i][uf]["missing"]=missing

# In the dictionary missing items also contain found items
for i,k in found_dict.items():
    for system in k:
        for item in (found_dict[i][system]["found"]):
            if item in found_dict[i][system]["missing"]:
                missing_list=found_dict[i][system]["missing"]
                missing_list.remove(item)

# Reach sizes of duplicate items
for i,k in found_dict.items():
    for system in k:
        found_items=found_dict[i][system]["found"]
```

In [ ]:

```
# Add a new list to the found dict -> for 1 waste source matching the size of space
# waste quantity == vacant space size
for space in found_dict:
    for system in found_dict[space]:
        found_dict[space][system]["enough waste"]=[]
        found_dict[space][system]["enough source"]=[]
        for item in found_dict[space][system]["source"]:
            if wastes[item]["size"]==vacant_spaces[space]["size"]:
                if found_dict[space][system]["enough waste"]==[]:
```

```
                    print(space,system,item,wastes[item]["type"],"1")
                    found_dict[space][system]["enough waste"].append(wastes[item]["type"])
                    found_dict[space][system]["enough source"].append(wastes[item]["tag"])
                    found_dict[space][system]["found"].remove(wastes[item]["type"])
                    found_dict[space][system]["source"].remove(wastes[item]["tag"])
                    break
                elif found_dict[space][system]["enough waste"]!=[]:
                    for waste in found_dict[space][system]["enough waste"]:
                        if wastes[waste]["type"]!=wastes[item]["type"]:
                            print(space,system,item,wastes[item]["type"],"2")
                            found_dict[space][system]["enough waste"].append(wastes[item]
["type"])
                            found_dict[space][system]["enough source"].append(wastes[item]
["tag"])
                            found_dict[space][system]["found"].remove(wastes[item]
["type"])
                            found_dict[space][system]["source"].remove(wastes[item]
["tag"])
```

In [ ]:

```
# Make a dictionary holding each vacant space& found and repeating waste type & waste
sources corresponding to found items
duplicate_items={}
for space,potential in found_dict.items():
    duplicate_items[space]={}
    for system, k in potential.items():
        if len(found_dict[space][system]["found"])> 1:
            found_items3=found_dict[space][system]["found"]
            sources=[]
            for index,items in enumerate(found_items3):
                duplicate_items[space][items]={}
                count=found_items3.count(items)
                if count>1:
                    items_str=str(items)
                    sources.append(found_dict[space][system]["source"][index])
                    duplicate_items[space][items]["matching sources"]=sources
```

In [ ]:

```
# Check if the total size of found waste matches the vacant space size
# sum (waste quantity) == vacant space size
satisfying_duplicate={}
not_enough_waste={}
for space, k in duplicate_items.items():
    print(space)
    satisfying_duplicate[space]={}
    not_enough_waste[space]={}
    for key,value in k.items():
        print(key)
        satisfying_duplicate[space][key]={}
        not_enough_waste[space][key]={}
        waste_matched=[]
        not_matched=[]
        if duplicate_items[space][key]!=[]:
            for m,n in duplicate_items[space][key].items():
                matched=duplicate_items[space][key][m]
                found_new=[]
                for items in matched:
                    found_new.append(items)
                    waste_matched.append(wastes[items]["size"])
                    if sum(waste_matched)==vacant_spaces[space]["size"]:
                        print(space,vacant_spaces[space]["size"], waste_matched, "add to
found list and remove from missing")
                        print(found_new, "is found")
                        print(waste_matched)
                        satisfying_duplicate[space][key]=found_new
                        break
                    else:
                        # print(sum(waste_matched),"And",vacant_spaces[space]
["size"],items)
                        not_matched.append(items)
                        print(not_matched,"is not matched")
                        not_enough_waste[space][key]=not_matched
```

In [ ]:

```
# Remove duplicate items if they are also in satisfying_duplicate items list until there
is only one of each waste type left
for space,potential in found_dict.items():
    for system in potential.keys():
        for key, value in satisfying_duplicate.items():
            for waste in value.keys():
                if key==space:
                        if len(found_dict[space][system]["found"])>0:
                            print("there are found items")
                            if len(satisfying_duplicate[key][waste])>0:
                                print("there are satisfying duplicate items")
                                if waste in found_dict[space][system]["found"]:
                                    print(waste,"is in found dict",space,system)
                                    found_items = found_dict[space][system]["found"]
                                    found_dict[space][system]["enough
waste"].append(waste)
                                    for item in satisfying_duplicate[key][waste]:
                                        found_dict[space][system]["enough
source"].append(item)

                                        found_dict[space][system]["source"].remove(item)
                                    for found_range in range(len(found_items)):
                                        for found in found_items:
                                            print(found_items,space,system)
                                            counter=found_items.count(found)
                                            print(waste,counter,space,system)
                                            if counter>1:
                                                print(found,"REMOVED",space,system)
                                                found_items.remove(found)
```

In [ ]:

```
# Remove duplicate items from "found list" if they are not in satisfying_duplicate items
list
for space in found_dict.keys():
    for system in found_dict[space].keys():
        found_items = found_dict[space][system]["found"]
        missing_items = found_dict[space][system]["missing"]
```

```
        duplicate=list_duplicates(found_items)
        for key, value in satisfying_duplicate.items():
            for waste in value.keys():
                if key==space:
                    for found in found_items:
                        if found in duplicate:
                            print("duplicate",space,found,system)
                            if satisfying_duplicate[space][found]==[]:
                                print("duplicate not satisfying",space,found,system)
                                found_items.remove(found)
                                if found not in missing_items:
                                    missing_items.append(found)
                                if (found in missing_items) and (found in found_items):
                                    found_items.remove(found)
for space in found_dict:
    for system in found_dict[space]:
        source=found_dict[space][system]["source"]
        for index,item in enumerate(source):
            waste=wastes[item]["type"]
            if waste in found_dict[space][system]["found"]:
                print("found",space,item,waste)
            else:
                source.remove(item)
                print(item,"removed from",space,system,waste)
```

In [ ]:

```
# Add symbiosis percentage to found_dict
for space in found_dict:
    for system in found_dict[space]:
        found_dict[space][system]["circularity"]=[]
        found=len`(found_dict[space][system]["enough waste"])
        missing=len(found_dict[space][system]["missing"])
        total=found+missing
        found_dict[space][system]["circularity"]=found/total
```

```
# Make a copy of found dict, sort system based on circularities
sorted_dict={}
for space in found_dict:
    system=sorted(found_dict[space], key=lambda x: (found_dict[space][x]['circularity']),
reverse=True)
    sorted_dict[space]={}
    for item in system:
        values={}
        values=(found_dict[space][item])
        sorted_dict[space][item]=values
```

Assign Food Production Systems

In [ ]:

```
# Check if the criteria is satisfied
# Rule 3: Assign a system if there are at least 1 found item
# Rule 4: Assign a system if there is no missing items
# Rule 5: Assign a system if there is one missing item which is not critical
# Rule 6: Assign a system if there is two missing items which are not critical
# Rule 7: If a found waste source is already used, it cannot be used again
# Rule 8: If a critical item is missing, that system cannot be asigned
```

```
occupied={}
used_waste=[]
used_waste_source=[]
used_waste_source_temp=[]
new_edges=[]
occupied_dict={}

for space in sorted_dict:
    print("looking for", space)
    print("for",space,sorted_dict[space],"is possible")
    occupied_dict[space]={}
    occupied_dict[space]["system"]={}
    occupied_dict[space]["found"]={}
    occupied_dict[space]["source"]={}
    occupied_dict[space]["missing"]={}
    for index,system in enumerate(sorted_dict[space]):
        print("looking for system", system)
        occupied[space]={}
        occupied[space]["system"]={}
        found_list=sorted_dict[space][system]["enough waste"]
        sources_list=sorted_dict[space][system]["enough source"]
        missing_list=sorted_dict[space][system]["missing"]
        print(len(missing_list),"is length for",space,system)

        if len(occupied[space]["system"])==0:
            print(space,"is not occupied run for",system)
            if len(missing_list)==0:
                for source in sources_list:
                    if source in used_waste_source:
                        print(used_waste_source,"is used",space,system)
                        pass
                    elif source not in used_waste_source:
                        print(used_waste_source,"is used",space,system)
                        occupied[space]["system"]=system
                        occupied_dict[space]["system"]=system
                        occupied_dict[space]["found"]=sorted_dict[space][system]["enough
waste"]
                        occupied_dict[space]["source"]=sorted_dict[space][system]["enough
source"]
                        for items in found_list:
                            used_waste.append(items)
                        used_waste_source.append(source)
                        edge_tuple=(source,space)
                        new_edges.append(edge_tuple)
                        print("no missing items:", space, system, "assign")
                if len(occupied[space]["system"])>0:
                    print(space,system,"will break")
                    break
            elif len(missing_list)==1:
                print("one item",system,space)
                for missing in missing_list:
                    if missing in non_critical_items:
                        print("one non critical item",missing,system,space)
                        for source in sources_list:
                            if source not in used_waste_source:
                                print("Used Non Critical Source",source)
                                occupied[space]["system"]=system
```

```
                                occupied_dict[space]["system"]=system
                                occupied_dict[space]["found"]=sorted_dict[space][system]
["enough waste"]
                                occupied_dict[space]["source"]=sorted_dict[space][system]
["enough source"]
                                occupied_dict[space]["missing"]=sorted_dict[space][system]
["missing"]
                                for items in found_list:
                                    used_waste.append(items)
                                used_waste_source.append(source)
                                edge_tuple=(source,space)
                                new_edges.append(edge_tuple)
                                print("one non critical missing items:", space, system,
"assign")
                            else:
                                print(source,"already used")
                    else:
                        print(space,"critical item missing:",missing,"for",system)
                if len(occupied[space]["system"])>0:
                    print(space,system,"will break")
                    break
            elif len(missing_list)==2:
                print("two items",system,space)
                for missing in missing_list:
                    print("two items",missing)
                    if missing in non_critical_items:
                        print("two items",missing,"not critical")
                        for source in sources_list:
                            if source not in used_waste_source:
                                print("Used Non Critical Source",source)
                                occupied[space]["system"]=system
                                occupied_dict[space]["system"]=system
                                occupied_dict[space]["found"]=sorted_dict[space][system]
["enough waste"]
                                occupied_dict[space]["source"]=sorted_dict[space][system]
["enough source"]
                                occupied_dict[space]["missing"]=sorted_dict[space][system]
["missing"]
                                for items in found_list:
                                    used_waste.append(items)
                                print("two non critical missing items:", space,
system,"assign")
                                print(space,occupied_dict[space])
                                used_waste_source.append(source)
                                edge_tuple=(source,space)
                                new_edges.append(edge_tuple)
                    else:
                        print(space,"critical item missing:",missing,"for",system)
                if len(occupied[space]["system"])>0:
                    print(space,system,"will break")
                    break
```

Create New List To Be Used In Next Steps

In [ ]:

```
# Add circularity & outputs to occupied_dict
for space in occupied_dict:
    occupied_dict[space]["circularity"]={}
    occupied_dict[space]["outputs"]={}
    occupied_dict[space]["supplements"]={}
    if len(occupied_dict[space]["system"])!=0:
        system=occupied_dict[space]["system"]
        outputs=(uf_systems[system]["out"])
        supplements=uf_systems[system]["supplement"]
        occupied_dict[space]["circularity"]=sorted_dict[space][system]["circularity"]
        occupied_dict[space]["supplements"]=supplements
        occupied_dict[space]["outputs"]=outputs

# Remove empty spaces from occupied_dict
remove=[]
for space in occupied_dict:
    if len(occupied_dict[space]["system"])==0:
        remove.append(space)
for items in remove:
    occupied_dict.pop(items)

# Remove occupied spaces from found_dict
remove2=[]
for space in sorted_dict:
    if space in occupied_dict:
        remove2.append(space)
for items in remove2:
    sorted_dict.pop(items)
```

Assign Food Producing Supplementary Systems

Prepare Data

In [ ]:

```
# Look for a system that can supply needed supplement ("S2" = nutrient dense water)
# Put the findings in a dictionary
v_supplement_fs={}
for space in occupied_dict:
    v_supplement_fs[space]={}
    v_supplement_fs[space]["supplement"]={}
    v_supplement_fs[space]["fs_system"]={}
    v_supplement_fs[space]["supplement"]={}
    v_supplement_fs[space]["supplement source"]={}
    v_supplement_fs[space]["fs_demand"]={}
    v_supplement_fs[space]["fs_demand source"]={}
    v_supplement_fs[space]["potential source"]={}
    supplement=occupied_dict[space]["supplements"]
    if supplement!=None:
        for item in supplement:
            if item=="S2":
                for i in uf_systems:
                    out=uf_systems[i]["out"]
                    if "S2" in out:
                        print("maybe",space, i,uf_systems[i]["in"])
                        v_supplement_fs[space]["fs_system"]=i
                        v_supplement_fs[space]["supplement"]=item
                        v_supplement_fs[space]["fs_demand"]=uf_systems[i]["in"]
    else:
        v_supplement_fs[space]["fs_system"]={}
```

In [ ]:

```
# We created a dictionary for spaces which need supplement to store system, supplementing
neighbor and supplement type for each space found what kind of system and supplement and
input is necessary
# Check the neighbors of supplement needing space to see if there is a potential neighbor
with the same size, enough structural capacity and same solar exosure
# Rule 8: (supplement needing space's size) == (its neighbors size)
# Rule 9: (neighbor's solar exposure) == (needed system's solar exposure)
# Rule 10: (neighbor's structural capacity) >= (needed system's weight)
for space in v_supplement_fs:
    potential=[]
    for space2,neighbor in nearby_space100:
        if space==space2:
            if v_supplement_fs[space]["fs_system"]!=None:
                if (len(v_supplement_fs[space]["supplement source"])==0) and
(len(v_supplement_fs[space]["fs_system"])!=0):
                    system=v_supplement_fs[space]["fs_system"]
                    print(system,"system")
                    print(vacant_spaces[space]["size"],vacant_spaces[neighbor]["size"])
                    if (vacant_spaces[space]["size"]==vacant_spaces[neighbor]["size"]) and
(vacant_spaces[neighbor]["structure"]>=uf_systems[system]["weight"]) and
(vacant_spaces[neighbor]["solar"]>=uf_systems[system]["solar"]):
                        potential.append(neighbor)
                        v_supplement_fs[space]["potential source"]=potential
        elif space==neighbor:
                print(neighbor,space2,"reverse is available")
                if v_supplement_fs[space]["fs_system"]!=None:
                    if (len(v_supplement_fs[space]["supplement source"])==0) and
(len(v_supplement_fs[space]["fs_system"])!=0):
                        print(space2,"is available")
                        system=v_supplement_fs[space]["fs_system"]
                        if (vacant_spaces[space]["size"]==vacant_spaces[space2]["size"])
and (vacant_spaces[space2]["structure"]>=uf_systems[system]["weight"]) and
(vacant_spaces[space2]["solar"]>=uf_systems[system]["solar"]):
                            potential.append(space2)
                            print(potential)
                            print(space,potential)
                            print(space,"matches",space2,"and",system)
                            v_supplement_fs[space]["potential source"]=potential
                        else:
                            print("structure",space,vacant_spaces[space2]
["structure"],system,uf_systems[system]["weight"])
                            print("sun",space,vacant_spaces[space2]
["solar"],system,uf_systems[system]["solar"])
                    else:
                        print(space,space2,"not available",v_supplement_fs[space]
["fs_system"],"no need for supplement")
```

In [ ]:

```
# Check if potential sources have necessary waste sources nearby
# Rule 11: For food producing supplementary systems all the items should be found
for space in v_supplement_fs:
    potential=v_supplement_fs[space]["potential source"]
    if len(v_supplement_fs[space]["supplement source"])==0:
        for vacant in potential:
            nearby_list=waste_dict[vacant]
            print(nearby_list)
            for nearby in nearby_list:
                print(vacant, nearby,"is",wastes[nearby]["type"])
                if nearby not in used_waste_source:
                    if wastes[nearby]["size"]==vacant_spaces[space]["size"]:
                        print(nearby,"not used")
                        if v_supplement_fs[space]["fs_demand"]==wastes[nearby]["type"]:
                            print("for",space,vacant,"is potential and has",nearby,"as a
source of",wastes[nearby]["type"])
                            v_supplement_fs[space]["supplement source"]=vacant
                            print(v_supplement_fs[space]["supplement source"])
                            v_supplement_fs[space]["fs_demand source"]=nearby
                            print(v_supplement_fs[space]["fs_demand source"])
                            used_waste_source.append(nearby)
                            edge_tuple1=(nearby,vacant)
                            edge_tuple2=(vacant,space)
                            new_edges.append(edge_tuple1)
                            new_edges.append(edge_tuple2)
                            if len(v_supplement_fs[space]["supplement source"])>0:
                                break
                        elif v_supplement_fs[space]["fs_demand"]!=wastes[nearby]["type"]:
                            v_supplement_fs[space]["supplement source"]=vacant
                            print(v_supplement_fs[space]["supplement source"])
                            v_supplement_fs[space]["fs_demand source"]="supply externally"
                            print(v_supplement_fs[space]["fs_demand source"])
                            used_waste_source.append(nearby)
                            edge_tuple1=(nearby,vacant)
                            edge_tuple2=(vacant,space)
                            new_edges.append(edge_tuple1)
                            new_edges.append(edge_tuple2)
                            if len(v_supplement_fs[space]["supplement source"])>0:
                                break

            if len(v_supplement_fs[space]["supplement source"])>0:
                            break
```

In [ ]:

```
# Add newly assigned food-producing supplementary systems to the "occupied nodes" list
for space in v_supplement_fs:
    if len(v_supplement_fs[space]["fs_system"])!=0:
        if (len(v_supplement_fs[space]["supplement source"])!=0) and
(v_supplement_fs[space]["fs_demand source"]!="supply externally"):
            occ=v_supplement_fs[space]["supplement source"]
            occupied_dict[occ]={}
            occupied_dict[occ]["found"]=v_supplement_fs[space]["fs_demand"]
            occupied_dict[occ]["system"]=v_supplement_fs[space]["fs_system"]
            system=occupied_dict[occ]["system"]
            occupied_dict[occ]["source"]=v_supplement_fs[space]["fs_demand source"]
            occupied_dict[occ]["supplements"]=uf_systems[system]["supplement"]
            occupied_dict[occ]["circularity"]=1
        elif v_supplement_fs[space]["fs_demand source"]=="supply externally":
            occ=v_supplement_fs[space]["supplement source"]
            occupied_dict[occ]={}
            occupied_dict[occ]["found"]=None
            occupied_dict[occ]["system"]=v_supplement_fs[space]["fs_system"]
            system=occupied_dict[occ]["system"]
            occupied_dict[occ]["source"]=v_supplement_fs[space]["fs_demand source"]
            occupied_dict[occ]["supplements"]=uf_systems[system]["supplement"]
            occupied_dict[occ]["circularity"]=0
```

In [ ]:

```
#   Make a dictionary of source: waste type: size: receiver:tuple
network_dict={}
for source,space in new_edges:
    type_list=[]
    network_dict[source]={}
    network_dict[source]["type"]={}
    network_dict[source]["size"]={}
    network_dict[source]["receiver"]={}

for source,space in new_edges:
    network_dict[source]["receiver"]=space
    if source in wastes:
        network_dict[source]["type"]=wastes[source]["type"]
        network_dict[source]["size"]=wastes[source]["size"]
    elif source in vacant_spaces:
        network_dict[source]["size"]=vacant_spaces[source]["size"]
        network_dict[source]["type"]=occupied_dict[source]["found"]
```

```
# Make a dictionary of source: waste type: size: receiver:tuple
network_dict={}
for source,space in new_edges:
    type_list=[]
    network_dict[source]={}
    network_dict[source]["type"]={}
    network_dict[source]["size"]={}
    network_dict[source]["receiver"]={}

for source,space in new_edges:
    network_dict[source]["receiver"]=space
    if source in wastes:
        network_dict[source]["type"]=wastes[source]["type"]
        network_dict[source]["size"]=wastes[source]["size"]
    elif source in vacant_spaces:
        network_dict[source]["size"]=vacant_spaces[source]["size"]
        network_dict[source]["type"]=occupied_dict[source]["found"]
```

Assigning Supplementary Systems

In [ ]:

```
# Now we have food supplying supplementary systems
# We still need to check if these systems need supplements ("S4" fish food or "S5"
fertiliser)
for space in occupied_dict:
    system=occupied_dict[space]["system"]
    value=uf_systems[system]["supplement"]
    if value!=None:
        for supplement in value:
            if (supplement=="S5") or (supplement=="S4"):
                v_supplement_fs[space]={}
                v_supplement_fs[space]["supplement"]=supplement
                for uf in uf_systems:
                    for out in uf_systems[uf]["out"]:
                        if supplement == out:
                            v_supplement_fs[space]["fs_system"]=uf
                            v_supplement_fs[space]["fs_demand"]=uf_systems[uf]["in"]
                v_supplement_fs[space]["supplement source"]={}
                v_supplement_fs[space]["fs_demand source"]={}
                v_supplement_fs[space]["potential source"]={}
```

In [ ]:

```
# Make a potential list of vacant spaces around the supplement needing space
# Rule 12: (supplement needing space's size) == (its neighbors size)
# Rule 13: (neighbor's solar exposure) == (needed system's solar exposure)
# Rule 14: (neighbor's structural capacity) >= (needed system's weight)
for space in v_supplement_fs:
    potential=[]
    for space2,neighbor in nearby_space100:
        if (space==space2):
            if (len(v_supplement_fs[space]["supplement source"])==0) and 
(len(v_supplement_fs[space]["fs_system"])!=0):
                system=v_supplement_fs[space]["fs_system"]
                if (vacant_spaces[space]["size"]==vacant_spaces[neighbor]["size"]) and 
(vacant_spaces[neighbor]["structure"]>=uf_systems[system]["weight"]) and 
(vacant_spaces[neighbor]["solar"]>=uf_systems[system]["solar"]):
                    potential.append(neighbor)
                    v_supplement_fs[space]["potential source"]=potential
                else:
                    print(space,vacant_spaces[neighbor]
["structure"],system,uf_systems[system]["weight"])
            else:
                print(space,neighbor,"not available",v_supplement_fs[space]
["fs_system"],"no need for supplement")
        elif space==neighbor:
            if (len(v_supplement_fs[space]["supplement source"])==0) and 
(len(v_supplement_fs[space]["fs_system"])!=0):
                print(space2,"is available")
                system=v_supplement_fs[space]["fs_system"]
                if (vacant_spaces[space]["size"]==vacant_spaces[space2]["size"]) and 
(vacant_spaces[space2]["structure"]>=uf_systems[system]["weight"]) and 
(vacant_spaces[space2]["solar"]>=uf_systems[system]["solar"]):
                    potential.append(space2)
                    v_supplement_fs[space]["potential source"]=potential
                else:
                    print("structure",space,vacant_spaces[space2]
["structure"],system,uf_systems[system]["weight"])
                    print("sun",space,vacant_spaces[space2]
["solar"],system,uf_systems[system]["solar"])
                    print("size",vacant_spaces[space]["size"]==vacant_spaces[space2]
["size"])
            else:
                print(space,space2,"not available",v_supplement_fs[space]["fs_system"],"no 
need for supplement")
```

In [ ]:

```
# Check if potential sources have necessary waste sources around
```

```
# Rule 15: For supplementary systems all the items should be found
for space in v_supplement_fs:
    potential=v_supplement_fs[space]["potential source"]
    if len(v_supplement_fs[space]["supplement source"])==0:
        for vacant in potential:
            found=[]
            found_source=[]
            nearby_list=waste_dict[vacant]
            for nearby in nearby_list:
                if nearby not in used_waste_source:
                    if wastes[nearby]["size"]==vacant_spaces[space]["size"]:
                        found.append(wastes[nearby]["type"])
                        found_source.append(nearby)
                        if v_supplement_fs[space]["fs_demand"]==found:
                            v_supplement_fs[space]["supplement source"]=vacant
                            v_supplement_fs[space]["fs_demand source"]=found_source
                            for x in found_source:
                                used_waste_source.append(x)
                                edge_tuple1=(x,vacant)
                                new_edges.append(edge_tuple1)
                            edge_tuple2=(vacant,space)
                            new_edges.append(edge_tuple2)
                            if len(v_supplement_fs[space]["supplement source"])>0:
                                # print(space,system,"will break")
                                break
            if len(v_supplement_fs[space]["supplement source"])>0:
                            # print(space,system,"will break2")
                            break
```

In [ ]:

```
# Add newly assigned supplementary systems to the "occupied nodes" list
for space in v_supplement_fs:
    occupied_dict[occ]["outputs"]={}
    if len(v_supplement_fs[space]["fs_system"])!=0:
        if (len(v_supplement_fs[space]["supplement source"])!=0):
            occ=v_supplement_fs[space]["supplement source"]
            occupied_dict[occ]={}

            occupied_dict[occ]["found"]=v_supplement_fs[space]["fs_demand"]
            occupied_dict[occ]["system"]=v_supplement_fs[space]["fs_system"]
            system=v_supplement_fs[space]["fs_system"]
            occupied_dict[occ]["source"]=v_supplement_fs[space]["fs_demand source"]
            occupied_dict[occ]["outputs"]=v_supplement_fs[space]["supplement"]
            occupied_dict[occ]["supplements"]=uf_systems[system]["supplement"]
            occupied_dict[occ]["circularity"]=1
```

In [ ]:

```
# Make a dictionary of source: waste type: size: receiver:tuple
network_dict={}
for source,space in new_edges:
    type_list=[]
    network_dict[source]={}
    network_dict[source]["type"]={}
    network_dict[source]["size"]={}
    network_dict[source]["receiver"]={}
```

```
for source,space in new_edges:
    network_dict[source]["receiver"]=space
    if source in wastes:
        network_dict[source]["type"]=wastes[source]["type"]
        network_dict[source]["size"]=wastes[source]["size"]
    elif source in vacant_spaces:
        network_dict[source]["size"]=vacant_spaces[source]["size"]
        network_dict[source]["type"]=occupied_dict[source]["outputs"]

# print(network_dict)

for space in occupied_dict:
    if occupied_dict[space]["system"]==None:
        occupied_dict.remove(space)
```

Preparation For Increasing The Search Radius

In [ ]:

```
# Export still vacant spaces, list of used waste sources, list of still available waste
sources to be used in next stage after increasing search radius
vacant_spaces2=vacant_spaces.copy()

for space in occupied_dict:
    vacant_spaces2.pop(space)

file = "vacant spaces_bigger radius.txt"
with open(str(file), 'w') as outfile:
    try:
        json.dump(vacant_spaces2, outfile)
        print(file + " has been updated successfully")
    except:
        print("Problem with updating file")


wastes2={}

for waste in wastes:
    if waste not in used_waste_source:
        wastes2[waste]={}
        wastes2[waste]["location"]=wastes[waste]["location"]
        wastes2[waste]["source"]=wastes[waste]["source"]
        wastes2[waste]["tag"]=wastes[waste]["tag"]
        wastes2[waste]["type"]=wastes[waste]["type"]
        wastes2[waste]["size"]=wastes[waste]["size"]
        wastes2[waste]["node_type"]=wastes[waste]["node_type"]

file = "available waste 2.txt"
with open(str(file), 'w') as outfile:
    try:
        json.dump(wastes2, outfile)
        print(file + " has been updated successfully")
    except:
        print("Problem with updating file")

file = "used waste.txt"
with open(str(file), 'w') as outfile:
    try:
```

```
        json.dump(used_waste_source, outfile)
        print(file + " has been updated successfully")
    except:
        print("Problem with updating file")
```

Data Preparation For Visualisation

- PREPARE DATA FOR EXPORT & EXPORT -

In [ ]:

```
# Combine wastes and vacant_spaces
workbook = load_workbook(filename="coordintes_xyz.xlsx")
workbook.sheetnames
sheet1 = workbook.worksheets[0]
sheet2 = workbook.worksheets[1]

new_edges2=[]
for space in network_dict:
    sources=(network_dict[space]["receiver"])
    tuples=(space,sources)
    new_edges2.append(tuples)
```

In [ ]:

```
# Update locations in dictionary based on excel worksheet
# This step needs to be done because the type of data for coordinates is "string" in the
excel sheets
# "strings" cannot be read in grasshopper as coordinates
coordinate_list1=[]
for value in sheet1.iter_rows(min_row=2, values_only=True):
    for index,item in enumerate(value):
            coordinate=(value[0],value[1],value[2])
    coordinate_list1.append(coordinate)

for index1,space in enumerate(vacant_spaces):
    for index2,coordinate in enumerate(coordinate_list1):
        if index1==index2:
            vacant_spaces[space]["location"]=coordinate_list1[index2]

workbook = load_workbook(filename="Node_Information_TU.xlsx")
workbook.sheetnames
sheet1 = workbook.worksheets[0]
sheet2 = workbook.worksheets[1]
wastes2={}
for value in sheet2.iter_rows(min_row=2, values_only=True):
    for index,item in enumerate(value):
        wastes2[value[0]]={}
        wastes2[value[0]]["location"]=value[1]
        wastes2[value[0]]["source"]=value[2]
        wastes2[value[0]]["type"]=value[3]
        wastes2[value[0]]["quantity"]=value[4]
        wastes2[value[0]]["tag"]=value[5]
        wastes2[value[0]]["node_type"]="waste"

workbook = load_workbook(filename="coordintes_xyz.xlsx")
workbook.sheetnames
sheet1 = workbook.worksheets[0]
sheet2 = workbook.worksheets[1]
```

```
# Construct coordinates from x,y,z values
coordinate_list2=[]
for value in sheet2.iter_rows(min_row=2, values_only=True):
    for index,item in enumerate(value):
            coordinate=(value[0],value[1],value[2])
    coordinate_list2.append(coordinate)

for index1,space in enumerate(wastes2):
    for index2,coordinate in enumerate(coordinate_list2):
        if index1==index2:
            # print(index2,coordinate,space)
            wastes2[space]["location"]=coordinate_list2[index2]

combined_dict=vacant_spaces.copy()
combined_dict.update(wastes2)

coordinates=[]
# Make a coordinate list for new_edges [(coordinates1,coordinates2),
(coordinates1,coordinates2),(coordinates1,coordinates2)]
for item in new_edges2:
        # print(item)
        coordinate_tuple=(combined_dict[item[0]]["location"],combined_dict[item[1]]
["location"])
        print(item,coordinate_tuple)
        coordinates.append(coordinate_tuple)
```

In [ ]:

```
# Make a dictionary of waste exchanges (start point, end point, waste type)
export_edges_dict={}
for index,couple in enumerate(coordinates):
        export_edges_dict[index]={}
        export_edges_dict[index]["coordinate"]=couple

export_edges_dict={}
for index,waste in enumerate(network_dict):
        export_edges_dict[index]={}
        export_edges_dict[index]["type"]=network_dict[waste]["type"]
        export_edges_dict[index]["connection"]=coordinates[index]
```

In [ ]:

```
# Make a dictionary of all spaces, systems, coordinates, symbiotic rates
export_dict={}
for space in vacant_spaces:
    export_dict[space]={}
    export_dict[space]["location"]={}
    export_dict[space]["system"]={}
for space in vacant_spaces:
    if space in occupied_dict:
        export_dict[space]["system"]=occupied_dict[space]["system"]
        export_dict[space]["location"]=combined_dict[space]["location"]
        export_dict[space]["circularity"]=occupied_dict[space]["circularity"]
    else:
        export_dict[space]["system"]=None
        export_dict[space]["location"]=None
```

Export Decisions

In [ ]:

```
# Export occupied nodes and connections to be visualised in Grasshopper
file = "occupied nodes.txt"
with open(str(file), 'w') as outfile:
    try:
        json.dump(export_dict, outfile)
        print(file + " has been updated successfully")
    except:
        print("Problem with updating file")

file = "new_edges.txt"
with open(str(file), 'w') as outfile:
    try:
        json.dump(new_edges, outfile)
        print(file + " has been updated successfully")
    except:
        print("Problem with updating file")

file = "new_edges_dict.txt"
with open(str(file), 'w') as outfile:
    try:
        json.dump(export_edges_dict, outfile)
        print(file + " has been updated successfully")
    except:
        print("Problem with updating file")

file = "coordinates.txt"
with open(str(file), 'w') as outfile:
    try:
        json.dump(coordinates, outfile)
        print(file + " has been updated successfully")
    except:
        print("Problem with updating file")
```

H.2 INCREASING RADIUS

- IMPORT TOOLS -

```
# Import data from previous stage regarding unoccupied spaces, used waste sources,
available waste sources and occupied spaces
url = "https://raw.githubusercontent.com/erengozdeanil/UF-
DecisionMaker/main/vacant%20spaces_bigger%20radius.txt"
resp = requests.get(url)
vacant_spaces = json.loads(resp.text)

url = "https://raw.githubusercontent.com/erengozdeanil/UF-
DecisionMaker/main/available%20waste%202.txt"
resp = requests.get(url)
wastes = json.loads(resp.text)

url = "https://raw.githubusercontent.com/erengozdeanil/UF-
DecisionMaker/main/occupied%20nodes.txt"
resp = requests.get(url)
occupied_nodes = json.loads(resp.text)

url = "https://raw.githubusercontent.com/erengozdeanil/UF-
DecisionMaker/main/used%20waste.txt"
resp = requests.get(url)
used_waste_source = json.loads(resp.text)
```

In [ ]:

```
# Retrieve connections between vacant spaces within radius=x
url1="https://raw.githubusercontent.com/erengozdeanil/UF-
DecisionMaker/main/Edges_vacant200.txt"
resp1 = requests.get(url1)
edges1 = json.loads(resp1.text)
#converts nested lists into a list of tuples
nearby_space300  = [tuple(i) for i in edges1]

#Retrieve connections within radius=x with identifiers
url1="https://raw.githubusercontent.com/erengozdeanil/UF-
DecisionMaker/main/Edges_try2.txt"
resp1 = requests.get(url1)
edges1 = json.loads(resp1.text)
#converts nested lists into a list of tuples
nearby_waste300 = [tuple(i) for i in edges1]
```

In [ ]:

```
# Make a new list composed of occupied nodes only
occupied={}
for space in occupied_nodes:
    if occupied_nodes[space]["system"]!=None:
        occupied[space]={}
        occupied[space]["system"]=occupied_nodes[space]["system"]
```

In [ ]:

```
# Make new lists of connections excluding used waste sources and occupied spaces
nearby_waste200=[]
for index,couple in enumerate(nearby_waste300):
    if couple[0] not in occupied:
        nearby_waste200.append(couple)

nearby_waste100=[]
for index,couple in enumerate(nearby_waste200):
    if couple[1] in wastes:
        nearby_waste100.append(couple)

nearby_space200=[]
for index,couple in enumerate(nearby_space300):
    if couple[0] not in occupied:
        nearby_space200.append(couple)

nearby_space100=[]
for index,couple in enumerate(nearby_space200):
    if couple[1] in vacant_spaces:
        nearby_space100.append(couple)
```

In [ ]:

```
# If a used waste source is in the list, remove it
for count in range(len(nearby_waste100)):
    for couple in nearby_waste100:
        if couple[1] in used_waste_source:
            # print(couple)
            nearby_waste100.remove(couple)

for couple in nearby_waste100:
    if couple[1] in used_waste_source:
        print(couple)
```

In [ ]:

```
# Remove empty waste sources from wastes list
for waste in wastes:
    for couple in nearby_waste100:
        if wastes[waste]["type"]=="None":
            if waste == couple[1]:
                print(couple, "removed")
                nearby_waste100.remove(couple)
                (couple,"removed")
print(nearby_waste100,len(nearby_waste100))
```

- REPEAT STAGE 1 STEPS -

H.3 OCCUPY ALL NODES

- IMPORT TOOLS -

- DATA PROCESSING -

```
# Assign a food production system
# Rule 1: If there are found item(s)
# Rule 2: If none of the missing items are critical
new_edges=[]
occupied_dict={}
for space in sorted_dict:
    occupied_dict[space]={}
    occupied_dict[space]["system"]={}
    occupied_dict[space]["found"]={}
    occupied_dict[space]["source"]={}
    occupied_dict[space]["missing"]={}
    for system in sorted_dict[space]:
        if len(sorted_dict[space][system]["found"])!=0:
            for missing in sorted_dict[space][system]["missing"]:
                if missing in non_critical_items:
                    occupied_dict[space]["system"]=system
                    occupied_dict[space]["found"]=(sorted_dict[space][system]["found"])
                    occupied_dict[space]["missing"]=sorted_dict[space][system]["missing"]
                    occupied_dict[space]["source"]=sorted_dict[space][system]["source"]
                    occupied_dict[space]["circularity"]=sorted_dict[space][system]
["circularity"]
                    for items in sorted_dict[space][system]["found"]:
                        used_waste_source.append(source)
                        edge_tuple=(source,space)
                        new_edges.append(edge_tuple)
            if len(occupied_dict[space]["system"])!=0:
                break
```

In [ ]:

```
# Assign a food production system even if there are not any found items
# Rule 3: If there is one missing item that is not critical
# Rule 4: If there are two missing items that are not critical
for space in sorted_dict:
    for system in sorted_dict[space]:
        if len(occupied_dict[space]["system"])==0:
            if len(sorted_dict[space][system]["found"])==0:
                if len(sorted_dict[space][system]["missing"])==1:
                    for missing in sorted_dict[space][system]["missing"]:
                        if missing in non_critical_items:
                            print(space)
                            occupied_dict[space]["system"]=system
                            occupied_dict[space]["found"]=None
                            occupied_dict[space]["missing"]=sorted_dict[space][system]
["missing"]
                            occupied_dict[space]["source"]="supply externally"
                    if len(occupied_dict[space]["system"])!=0:
                        break

                if len(sorted_dict[space][system]["missing"])==2:
                    for missing in sorted_dict[space][system]["missing"]:
                        if missing in non_critical_items:
                            print(space)
                            occupied_dict[space]["system"]=system
                            occupied_dict[space]["found"]=None
                            occupied_dict[space]["missing"]=sorted_dict[space][system]
["missing"]
                            occupied_dict[space]["source"]="supply externally"
                    if len(occupied_dict[space]["system"])!=0:
                        break
```

Create New List To Be Used In Next Steps

In [ ]:

```
# Add circularity & outputs to occupied_dict
for space in occupied_dict:
    occupied_dict[space]["circularity"]={}
    occupied_dict[space]["outputs"]={}
    occupied_dict[space]["supplements"]={}
    if len(occupied_dict[space]["system"])!=0:
        system=occupied_dict[space]["system"]
        outputs=(uf_systems[system]["out"])
        supplements=uf_systems[system]["supplement"]
        occupied_dict[space]["supplements"]=supplements
        occupied_dict[space]["outputs"]=outputs
        occupied_dict[space]["circularity"]=sorted_dict[space][system]["circularity"]

# Remove empty spaces from occupied_dict
remove=[]
for space in occupied_dict:
    if len(occupied_dict[space]["system"])==0:
        remove.append(space)
for items in remove:
    occupied_dict.pop(items)

# Remove occupied spaces from found_dict list
remove2=[]
for space in sorted_dict:
    if space in occupied_dict:
        remove2.append(space)
print(len(remove))
for items in remove2:
    sorted_dict.pop(items)
```

Assign Food Producing Supplementary Systems

In [ ]:

```
# Look for a system that can supply needed supplement ("S2" Nutrient Dense Water)
# Put the findings in a dictionary
v_supplement_fs={}
for space in occupied_dict:
    v_supplement_fs[space]={}
    v_supplement_fs[space]["supplement"]={}
    v_supplement_fs[space]["fs_system"]={}
    v_supplement_fs[space]["supplement"]={}
    v_supplement_fs[space]["supplement source"]={}
    v_supplement_fs[space]["fs_demand"]={}
    v_supplement_fs[space]["fs_demand source"]={}
    v_supplement_fs[space]["potential source"]={}
    supplement=occupied_dict[space]["supplements"]
    if supplement!= None:
        for item in supplement:
            if item=="S2":
                for i in uf_systems:
                    out=uf_systems[i]["out"]
                    if "S2" in out:
                        v_supplement_fs[space]["fs_system"]=i
                        v_supplement_fs[space]["supplement"]=item
                        v_supplement_fs[space]["fs_demand"]=uf_systems[i]["in"]
    else:
        v_supplement_fs[space]["fs_system"]=None
```

In [ ]:

```
# We created a dictionary for spaces which need supplement to store system, supplementing
neighbor and supplement type for each space found what kind of system and supplement and
input is necessary
# Check the neighbors of supplement needing space to see if there is a potential neighbor
# Rule 5: (supplement needing space's size) == (its neighbors size)
# Rule 6: (neighbor's solar exposure) == (needed system's solar exposure)
# Rule 7: (neighbor's structural capacity) >= (needed system's weight)
for space in v_supplement_fs:
    potential=[]
    if space in vacant_spaces:

        for space2,neighbor in nearby_space100:
            if neighbor in vacant_spaces:
                if space==space2:
                    if (v_supplement_fs[space]["supplement source"]==None) and
(len(v_supplement_fs[space]["fs_system"])!=0):
                        # print(neighbor,"is available")
                        system=v_supplement_fs[space]["fs_system"]
                        print(system,"system")
                        print(vacant_spaces[space]["size"],vacant_spaces[neighbor]
["size"])
                        if (vacant_spaces[space]["size"]==vacant_spaces[neighbor]["size"])
and (vacant_spaces[neighbor]["structure"]>=uf_systems[system]["weight"]) and
(vacant_spaces[neighbor]["solar"]>=uf_systems[system]["solar"]):
                            potential.append(neighbor)
                            # print(space,potential)
                            v_supplement_fs[space]["potential source"]=potential
                elif space==neighbor:
                        print(neighbor,space2,"reverse is available")
                        if (v_supplement_fs[space]["supplement source"]==None) and
(len(v_supplement_fs[space]["fs_system"])!=0):
                            print(space2,"is available")
                            system=v_supplement_fs[space]["fs_system"]
                            if (vacant_spaces[space]["size"]==vacant_spaces[space2]
["size"]) and (vacant_spaces[space2]["structure"]>=uf_systems[system]["weight"]) and
(vacant_spaces[space2]["solar"]>=uf_systems[system]["solar"]):
                                potential.append(space2)
                                print(potential)
                                print(space,potential)
                                print(space,"matches",space2,"and",system)
                                v_supplement_fs[space]["potential source"]=potential
                            else:
                                print("structure",space,vacant_spaces[space2]
["structure"],system,uf_systems[system]["weight"])
                                print("sun",space,vacant_spaces[space2]
["solar"],system,uf_systems[system]["solar"])
                        else:
```

```
occupied={}
used_waste=[]
used_waste_source=[]
used_waste_source_temp=[]
new_edges=[]
occupied_dict={}

for space in sorted_dict:
    print("looking for", space)
    print("for",space,sorted_dict[space],"is possible")
    occupied_dict[space]={}
    occupied_dict[space]["system"]={}
    occupied_dict[space]["found"]={}
    occupied_dict[space]["source"]={}
    occupied_dict[space]["missing"]={}
    for index,system in enumerate(sorted_dict[space]):
        print("looking for system", system)
        occupied[space]={}
        occupied[space]["system"]={}
        found_list=sorted_dict[space][system]["enough waste"]
        sources_list=sorted_dict[space][system]["enough source"]
        missing_list=sorted_dict[space][system]["missing"]
        print(len(missing_list),"is length for",space,system)

        if len(occupied[space]["system"])==0:
            print(space,"is not occupied run for",system)
            if len(missing_list)==0:
                for source in sources_list:
                    if source in used_waste_source:
                        print(used_waste_source,"is used",space,system)
                        pass
                    elif source not in used_waste_source:
                        print(used_waste_source,"is used",space,system)
                        occupied[space]["system"]=system
                        occupied_dict[space]["system"]=system
                        occupied_dict[space]["found"]=sorted_dict[space][system]["enough
waste"]
                        occupied_dict[space]["source"]=sorted_dict[space][system]["enough
source"]
                        for items in found_list:
                            used_waste.append(items)
                        used_waste_source.append(source)
                        edge_tuple=(source,space)
                        new_edges.append(edge_tuple)
                        print("no missing items:", space, system, "assign")
                if len(occupied[space]["system"])>0:
                    print(space,system,"will break")
                    break
            elif len(missing_list)==1:
                print("one item",system,space)
                for missing in missing_list:
                    if missing in non_critical_items:
                        print("one non critical item",missing,system,space)
                        for source in sources_list:
                            if source not in used_waste_source:
                                print("Used Non Critical Source",source)
                                occupied[space]["system"]=system
            system=occupied_dict[occ]["system"]
            occupied_dict[occ]["source"]=v_supplement_fs[space]["fs_demand source"]
            occupied_dict[occ]["supplements"]=uf_systems[system]["supplement"]
            occupied_dict[occ]["circularity"]=0
```

Assigning Supplementary Systems

In [ ]:

```
for space in v_supplement_fs:
    if (v_supplement_fs[space]["supplement source"])!={}:
        if (v_supplement_fs[space]["fs_system"])!={}:

            if len(v_supplement_fs[space]["supplement source"])!=0:
                occ=v_supplement_fs[space]["supplement source"]
                occupied_dict[occ]={}
                print(occupied_dict)
                occupied_dict[occ]["found"]=v_supplement_fs[space]["fs_demand"]
                occupied_dict[occ]["system"]=v_supplement_fs[space]["fs_system"]
                occupied_dict[occ]["source"]=v_supplement_fs[space]["fs_demand source"]
                occupied_dict[occ]["circularity"]=1

# Now we have food supplying supplementary systems
# We still need to check if these systems need supplements ("S4" fish food or "S5" fertiliser)
for space in occupied_dict:
    system=occupied_dict[space]["system"]
    value=uf_systems[system]["supplement"]
    if value!=None:
        for supplement in value:
            if (supplement=="S5") or (supplement=="S4"):
                v_supplement_fs[space]={}
                v_supplement_fs[space]["supplement"]=supplement
                for uf in uf_systems:
                    for out in uf_systems[uf]["out"]:
                        if supplement == out:
                            v_supplement_fs[space]["fs_system"]=uf
                            v_supplement_fs[space]["fs_demand"]=uf_systems[uf]["in"]
                    if len(v_supplement_fs[space]["fs_system"])>0:
                        break
                v_supplement_fs[space]["supplement source"]={}
                v_supplement_fs[space]["fs_demand source"]={}
                v_supplement_fs[space]["potential source"]={}
```

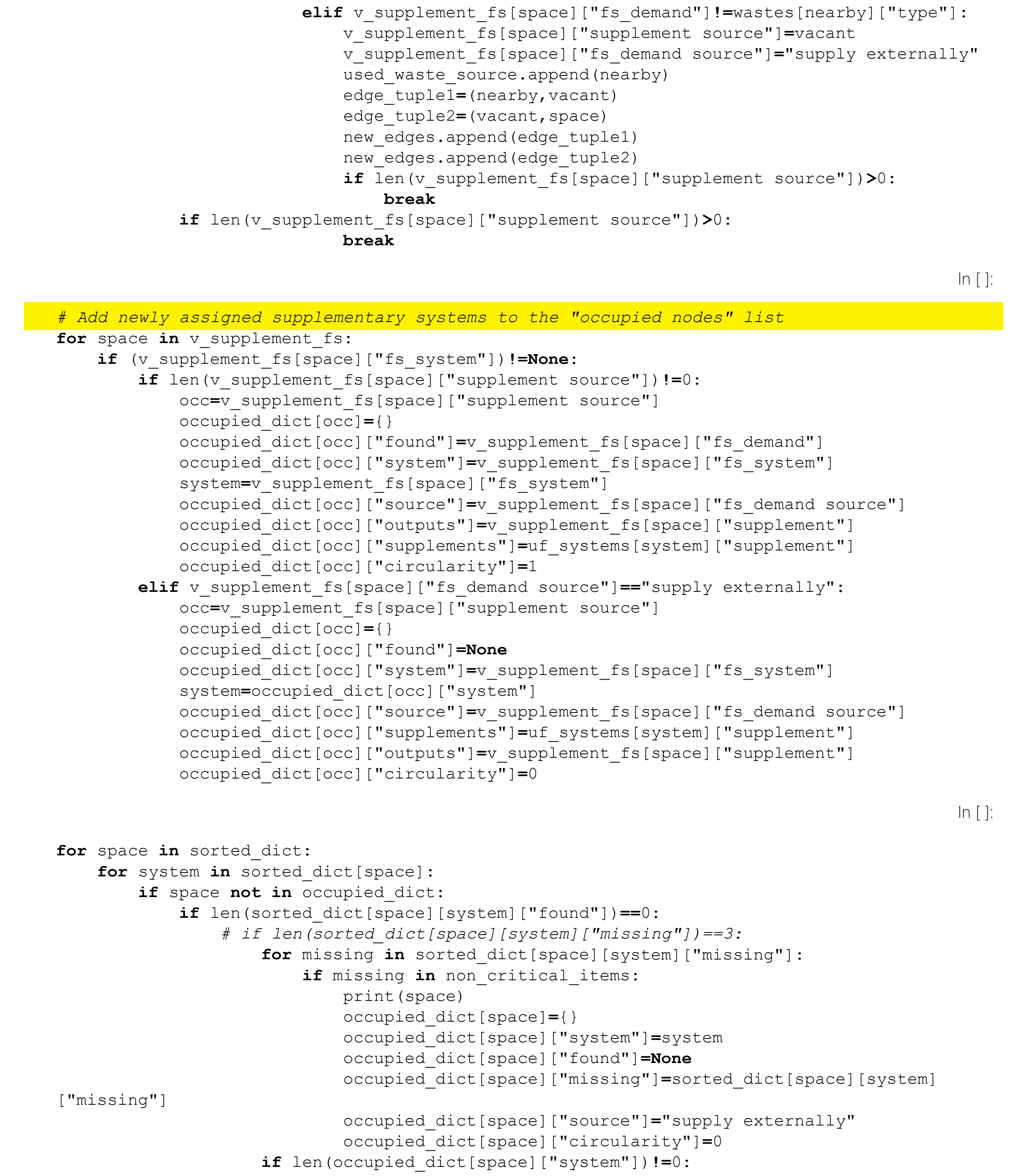
In [ ]:

```
# Make a potential list of vacant spaces around the supplement needing space
# Rule 12: (supplement needing space's size) == (its neighbors size)
# Rule 13: (neighbor's solar exposure) == (needed system's solar exposure)
# Rule 14: (neighbor's structural capacity) >= (needed system's weight)
for space in v_supplement_fs:
    potential=[]
    if space in vacant_spaces:
        if neighbor in vacant_spaces:
            for space2,neighbor in nearby_space100:
                if space2 in vacant_spaces:
                    if (space==space2):
                        print(space2,neighbor)
```

```
                        if (len(v_supplement_fs[space]["supplement source"])==0) and (v_supplement_fs[space]["fs_system"]!=None):
                            system=v_supplement_fs[space]["fs_system"]
                            if (vacant_spaces[space]["size"]==vacant_spaces[neighbor]["size"]) and (vacant_spaces[neighbor]["structure"]>=uf_systems[system]["weight"]) and (vacant_spaces[neighbor]["solar"]>=uf_systems[system]["solar"]):
                                potential.append(neighbor)
                                v_supplement_fs[space]["potential source"]=potential
                            else:
                                print(space,vacant_spaces[neighbor]["structure"],system,uf_systems[system]["weight"])
                        else:
                            print(space,neighbor,"not available",v_supplement_fs[space]["fs_system"],"no need for supplement")
                    elif space==neighbor:
                        print(neighbor,space2,"reverse is available")
                        if (v_supplement_fs[space]["supplement source"]=={}) and ((v_supplement_fs[space]["fs_system"])!=None):
                            system=v_supplement_fs[space]["fs_system"]
                            if (vacant_spaces[space]["size"]==vacant_spaces[space2]["size"]) and (vacant_spaces[space2]["structure"]>=uf_systems[system]["weight"]) and (vacant_spaces[space2]["solar"]>=uf_systems[system]["solar"]):
                                potential.append(space2)
                                v_supplement_fs[space]["potential source"]=potential
                            else:
                                print("structure",space,vacant_spaces[space2]["structure"],system,uf_systems[system]["weight"])
                                print("sun",space,vacant_spaces[space2]["solar"],system,uf_systems[system]["solar"])
                                print("size",vacant_spaces[space]["size"]==vacant_spaces[space2]["size"])
                        else:
                            print(space,space2,"not available",v_supplement_fs[space]["fs_system"],"no need for supplement")
```

In [ ]:

```
# Check if potential sources have necessary waste sources around
# Rule 15: For supplementary systems all the items should be found
for space in v_supplement_fs:
    potential=v_supplement_fs[space]["potential source"]
    if len(v_supplement_fs[space]["supplement source"])==0:
        for vacant in potential:
            nearby_list=waste_dict[vacant]
            for nearby in nearby_list:
                if nearby not in used_waste_source:
                    if wastes[nearby]["size"]==vacant_spaces[space]["size"]:
                        if v_supplement_fs[space]["fs_demand"]==wastes[nearby]["type"]:
                            v_supplement_fs[space]["supplement source"]=vacant
                            v_supplement_fs[space]["fs_demand source"]=nearby
                            used_waste_source.append(nearby)
                            edge_tuple1=(nearby,vacant)
                            edge_tuple2=(vacant,space)
                            new_edges.append(edge_tuple1)
                            new_edges.append(edge_tuple2)
                            if len(v_supplement_fs[space]["supplement source"])>0:
                                break
```



```
                        elif v_supplement_fs[space]["fs_demand"]!=wastes[nearby]["type"]:
                            v_supplement_fs[space]["supplement source"]=vacant
                            v_supplement_fs[space]["fs_demand source"]="supply externally"
                            used_waste_source.append(nearby)
                            edge_tuple1=(nearby,vacant)
                            edge_tuple2=(vacant,space)
                            new_edges.append(edge_tuple1)
                            new_edges.append(edge_tuple2)
                            if len(v_supplement_fs[space]["supplement source"])>0:
                                break
            if len(v_supplement_fs[space]["supplement source"])>0:
                            break
```

In [ ]:

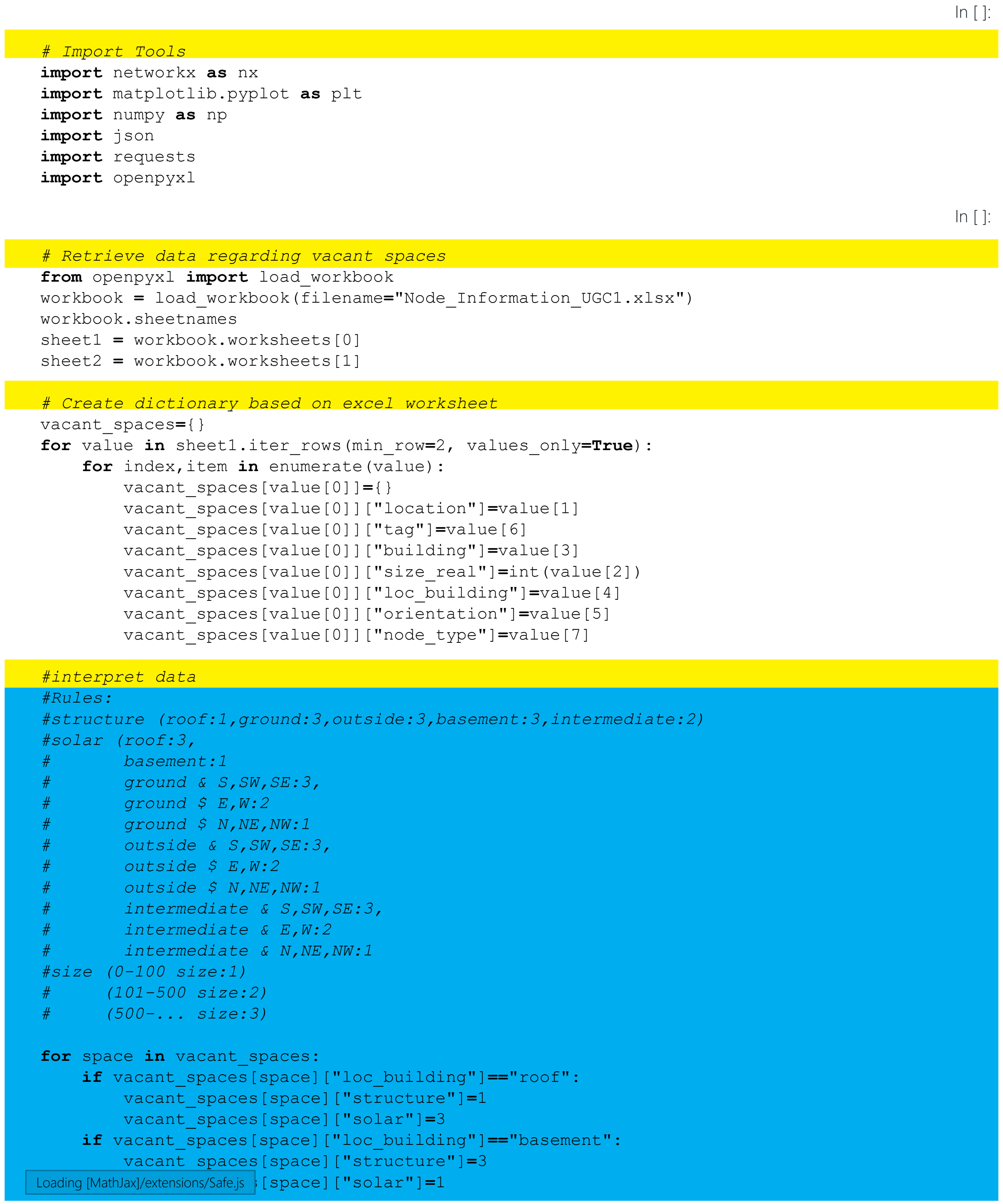
```
# Add newly assigned supplementary systems to the "occupied nodes" list
for space in v_supplement_fs:
    if (v_supplement_fs[space]["fs_system"])!=None:
        if len(v_supplement_fs[space]["supplement source"])!=0:
            occ=v_supplement_fs[space]["supplement source"]
            occupied_dict[occ]={}
            occupied_dict[occ]["found"]=v_supplement_fs[space]["fs_demand"]
            occupied_dict[occ]["system"]=v_supplement_fs[space]["fs_system"]
            system=v_supplement_fs[space]["fs_system"]
            occupied_dict[occ]["source"]=v_supplement_fs[space]["fs_demand source"]
            occupied_dict[occ]["outputs"]=v_supplement_fs[space]["supplement"]
            occupied_dict[occ]["supplements"]=uf_systems[system]["supplement"]
            occupied_dict[occ]["circularity"]=1
        elif v_supplement_fs[space]["fs_demand source"]=="supply externally":
            occ=v_supplement_fs[space]["supplement source"]
            occupied_dict[occ]={}
            occupied_dict[occ]["found"]=None
            occupied_dict[occ]["system"]=v_supplement_fs[space]["fs_system"]
            system=occupied_dict[occ]["system"]
            occupied_dict[occ]["source"]=v_supplement_fs[space]["fs_demand source"]
            occupied_dict[occ]["supplements"]=uf_systems[system]["supplement"]
            occupied_dict[occ]["outputs"]=v_supplement_fs[space]["supplement"]
            occupied_dict[occ]["circularity"]=0
```

In [ ]:

```
for space in sorted_dict:
    for system in sorted_dict[space]:
        if space not in occupied_dict:
            if len(sorted_dict[space][system]["found"])==0:
                # if len(sorted_dict[space][system]["missing"])==3:
                    for missing in sorted_dict[space][system]["missing"]:
                        if missing in non_critical_items:
                            print(space)
                            occupied_dict[space]={}
                            occupied_dict[space]["system"]=system
                            occupied_dict[space]["found"]=None
                            occupied_dict[space]["missing"]=sorted_dict[space][system]
["missing"]
                            occupied_dict[space]["source"]="supply externally"
                            occupied_dict[space]["circularity"]=0
                    if len(occupied_dict[space]["system"])!=0:
```

- PREPARE DATA FOR EXPORT & EXPORT -

H.4 STATISTICS



In [ ]:

```
# Import Tools
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
import json
import requests
import openpyxl
```

In [ ]:

```
# Retrieve data regarding vacant spaces
from openpyxl import load_workbook
workbook = load_workbook(filename="Node_Information_UGC1.xlsx")
workbook.sheetnames
sheet1 = workbook.worksheets[0]
sheet2 = workbook.worksheets[1]

# Create dictionary based on excel worksheet
vacant_spaces={}
for value in sheet1.iter_rows(min_row=2, values_only=True):
    for index,item in enumerate(value):
        vacant_spaces[value[0]]={}
        vacant_spaces[value[0]]["location"]=value[1]
        vacant_spaces[value[0]]["tag"]=value[6]
        vacant_spaces[value[0]]["building"]=value[3]
        vacant_spaces[value[0]]["size_real"]=int(value[2])
        vacant_spaces[value[0]]["loc_building"]=value[4]
        vacant_spaces[value[0]]["orientation"]=value[5]
        vacant_spaces[value[0]]["node_type"]=value[7]

#interpret data
#Rules:
#structure  (roof:1,ground:3,outside:3,basement:3,intermediate:2)
#solar  (roof:3,
#       basement:1
#       ground & S,SW,SE:3,
#       ground $ E,W:2
#       ground $ N,NE,NW:1
#       outside & S,SW,SE:3,
#       outside $ E,W:2
#       outside $ N,NE,NW:1
#       intermediate & S,SW,SE:3,
#       intermediate & E,W:2
#       intermediate & N,NE,NW:1
#size  (0-100 size:1)
#     (101-500 size:2)
#     (500-... size:3)

for space in vacant_spaces:
    if vacant_spaces[space]["loc_building"]=="roof":
        vacant_spaces[space]["structure"]=1
        vacant_spaces[space]["solar"]=3
    if vacant_spaces[space]["loc_building"]=="basement":
        vacant_spaces[space]["structure"]=3
Loading [MathJax]/extensions/Safe.js [space]["solar"]=1
```

```
    if (vacant_spaces[space]["loc_building"]=="ground") and (((vacant_spaces[space]
["orientation"])=="S") or ((vacant_spaces[space]["orientation"])=="SE") or
((vacant_spaces[space]["orientation"])=="SW")):
        vacant_spaces[space]["solar"]=3
        vacant_spaces[space]["structure"]=3
    if (vacant_spaces[space]["loc_building"]=="ground") and (((vacant_spaces[space]
["orientation"])=="N") or ((vacant_spaces[space]["orientation"])=="NE") or
((vacant_spaces[space]["orientation"])=="NW")):
        vacant_spaces[space]["solar"]=1
        vacant_spaces[space]["structure"]=3
    if (vacant_spaces[space]["loc_building"]=="ground") and (((vacant_spaces[space]
["orientation"])=="E") or ((vacant_spaces[space]["orientation"])=="W")):
        vacant_spaces[space]["solar"]=2
        vacant_spaces[space]["structure"]=3
    if (vacant_spaces[space]["loc_building"]=="intermediate") and (((vacant_spaces[space]
["orientation"])=="S") or ((vacant_spaces[space]["orientation"])=="SE") or
((vacant_spaces[space]["orientation"])=="SW")):
        vacant_spaces[space]["solar"]=3
        vacant_spaces[space]["structure"]=2
    if (vacant_spaces[space]["loc_building"]=="intermediate") and (((vacant_spaces[space]
["orientation"])=="N") or ((vacant_spaces[space]["orientation"])=="NE") or
((vacant_spaces[space]["orientation"])=="NW")):
        vacant_spaces[space]["solar"]=1
        vacant_spaces[space]["structure"]=2
    if (vacant_spaces[space]["loc_building"]=="intermediate") and (((vacant_spaces[space]
["orientation"])=="E") or ((vacant_spaces[space]["orientation"])=="W")):
        vacant_spaces[space]["solar"]=2
        vacant_spaces[space]["structure"]=2
    if (vacant_spaces[space]["loc_building"]=="outside") and (((vacant_spaces[space]
["orientation"])=="S") or ((vacant_spaces[space]["orientation"])=="SE") or
((vacant_spaces[space]["orientation"])=="SW")):
        vacant_spaces[space]["solar"]=3
        vacant_spaces[space]["structure"]=3
    if (vacant_spaces[space]["loc_building"]=="outside") and (((vacant_spaces[space]
["orientation"])=="N") or ((vacant_spaces[space]["orientation"])=="NE") or
((vacant_spaces[space]["orientation"])=="NW")):
        vacant_spaces[space]["solar"]=1
        vacant_spaces[space]["structure"]=3
    if (vacant_spaces[space]["loc_building"]=="outside") and (((vacant_spaces[space]
["orientation"])=="E") or ((vacant_spaces[space]["orientation"])=="W")):
        vacant_spaces[space]["solar"]=2
        vacant_spaces[space]["structure"]=3

# Interpret size based on predefined ranges
for space in vacant_spaces:
    if vacant_spaces[space]["size_real"]<=200:
        vacant_spaces[space]["size"]=1
    if (vacant_spaces[space]["size_real"]>200) and (vacant_spaces[space]
["size_real"]<=1000) :
        vacant_spaces[space]["size"]=2
    if (vacant_spaces[space]["size_real"]>1000):
        vacant_spaces[space]["size"]=3

# Retrieve waste sources
# Create dictionary based on excel worksheet
wastes={}
for value in sheet2.iter_rows(min_row=2, values_only=True):
```

```
    for index,item in enumerate(value):
            wastes[value[0]]={}
            wastes[value[0]]["location"]=value[1]
            wastes[value[0]]["tag"]=value[5]
            wastes[value[0]]["type"]=value[3]
```

In [ ]:

```
workbook = load_workbook(filename="Node_Information_UGC1.xlsx")
workbook.sheetnames
sheet1 = workbook.worksheets[0]
sheet2 = workbook.worksheets[1]
wastes={}
for value in sheet2.iter_rows(min_row=2, values_only=True):
    for index,item in enumerate(value):
        if value[4]!=0:
            wastes[value[0]]={}
            wastes[value[0]]["location"]=value[1]
            wastes[value[0]]["source"]=value[2]
            wastes[value[0]]["tag"]=value[5]
            wastes[value[0]]["type"]=value[3]
            wastes[value[0]]["quantity"]=int(value[4])
            wastes[value[0]]["node_type"]="waste"


# Assign ranges to quantities
for waste in wastes:
    if wastes[waste]["type"]=="W1":
        if wastes[waste]["quantity"]<= 183705:
            wastes[waste]["size"]=1
        if (wastes[waste]["quantity"]>183705) and (wastes[waste]["quantity"]<=918523):
            wastes[waste]["size"]=2
        if wastes[waste]["quantity"]>918523:
            wastes[waste]["size"]=3
    if wastes[waste]["type"]=="W2":
        if wastes[waste]["quantity"]<= 67500:
            wastes[waste]["size"]=1
        if (wastes[waste]["quantity"]>67500) and (wastes[waste]["quantity"]<=337500):
            wastes[waste]["size"]=2
        if wastes[waste]["quantity"]>337500:
            wastes[waste]["size"]=3
    if wastes[waste]["type"]=="W3":
        if wastes[waste]["quantity"]<= 11813:
            wastes[waste]["size"]=1
        if (wastes[waste]["quantity"]>11813) and (wastes[waste]["quantity"]<=59063):
            wastes[waste]["size"]=2
        if wastes[waste]["quantity"]>59063:
            wastes[waste]["size"]=3
    if wastes[waste]["type"]=="W4":
        if wastes[waste]["quantity"]<= 28200:
            wastes[waste]["size"]=1
        if (wastes[waste]["quantity"]>28200) and (wastes[waste]["quantity"]<=141000):
            wastes[waste]["size"]=2
        if wastes[waste]["quantity"]>141000:
            wastes[waste]["size"]=3
    if wastes[waste]["type"]=="W5":
        if wastes[waste]["quantity"]<= 19342:
            wastes[waste]["size"]=1
        if (wastes[waste]["quantity"]>19342) and (wastes[waste]["quantity"]<=96710):
            wastes[waste]["size"]=2
        if wastes[waste]["quantity"]>96710:
            wastes[waste]["size"]=3
    if wastes[waste]["type"]=="W6":
        if wastes[waste]["quantity"]<= 1560000:
            wastes[waste]["size"]=1
        if (wastes[waste]["quantity"]>1560000) and (wastes[waste]["quantity"]<=7800000):
            wastes[waste]["size"]=2
        if wastes[waste]["quantity"]>7800000:
            wastes[waste]["size"]=3
    if wastes[waste]["type"]=="W7":
        if wastes[waste]["quantity"]<= 200000:
            wastes[waste]["size"]=1
        if (wastes[waste]["quantity"]>200000) and (wastes[waste]["quantity"]<=1000000):
            wastes[waste]["size"]=2
        if wastes[waste]["quantity"]>1000000:
            wastes[waste]["size"]=3
```

In [ ]:

```
# Import Used Waste List
url1="https://raw.githubusercontent.com/erengozdeanil/UF-
DecisionMaker/main/used%20waste_ugc.txt"
resp1 = requests.get(url1)
used_waste1 = json.loads(resp1.text)
```

In [ ]:

```
# Import Space & System List
url = "https://raw.githubusercontent.com/erengozdeanil/UF-
DecisionMaker/main/occupied%20nodes_ugc.txt"
resp = requests.get(url)
occupied_nodes1 = json.loads(resp.text)

url = "https://raw.githubusercontent.com/erengozdeanil/UF-
DecisionMaker/main/occupied%20nodes4_ugc.txt"
resp = requests.get(url)
occupied_nodes2 = json.loads(resp.text)
```

In [ ]:

```
# Combine occupied spaces list from all of the previous stages
occ_space={}
vacant=[]
x1=[]
x2=[]
x3=[]
x4=[]
for space in occupied_nodes1:
    if occupied_nodes1[space]["system"]==None:
        continue
    else:
        x1.append(space)
        occ_space[space]={}
        occ_space[space]["size"]=vacant_spaces[space]["size_real"]
        occ_space[space]["system"]=occupied_nodes1[space]["system"]
```

```
for space in occupied_nodes2:
    if occupied_nodes2[space]["system"]==None:
        continue
    else:
        x2.append(space)
        occ_space[space]={}
        occ_space[space]["size"]=vacant_spaces[space]["size_real"]
        occ_space[space]["system"]=occupied_nodes2[space]["system"]

for space in vacant_spaces:
    if space not in occ_space:
        vacant.append(space)
```

In [ ]:

```
#Yields per m2
# UF1 :
uf1_product="worm"
uf1_yield=4.87
# UF2 :
uf2_product="fish"
uf2_yield=15
# UF3 :
uf3_product="mushroom"
uf3_yield=125
# UF4 :
uf4_product="small veg"
uf4_yield=99.9
# UF5 :
uf5_product="small veg"
uf5_product2="big veg"
uf5_yield=12.5
uf5_yield2=89.6
# UF6 :
uf6_product="small veg"
uf6_product2="big veg"
uf6_yield=12.5
uf6_yield2=2.7
# UF7 :
uf7_product="small veg"
uf7_yield=12.5
# UF8 :
uf8_product="small veg"
uf8_yield=87.4
# UF9 :
uf9_product="small veg"
uf9_yield=12.5
```

In [ ]:

```
# For each occupied space multiply space' size with (yield per m2) based on the system
for space in occ_space:
    occ_space[space]["produce"]=None
    occ_space[space]["produce2"]=None
    occ_space[space]["yield"]=None
    occ_space[space]["yield2"]=None
    if occ_space[space]["system"]=="UF1":
        occ_space[space]["produce"]=uf1_product
        occ_space[space]["yield"]=occ_space[space]["size"]*uf1_yield
    if occ_space[space]["system"]=="UF2":
        occ_space[space]["produce"]=uf2_product
        occ_space[space]["yield"]=occ_space[space]["size"]*uf2_yield
    if occ_space[space]["system"]=="UF3":
        occ_space[space]["produce"]=uf3_product
        occ_space[space]["yield"]=occ_space[space]["size"]*uf3_yield
    if occ_space[space]["system"]=="UF4":
        occ_space[space]["produce"]=uf4_product
        occ_space[space]["yield"]=occ_space[space]["size"]*uf4_yield
    if occ_space[space]["system"]=="UF5":
        occ_space[space]["produce"]=uf5_product
        occ_space[space]["yield"]=occ_space[space]["size"]*uf5_yield
        occ_space[space]["produce2"]=uf5_product2
        occ_space[space]["yield2"]=occ_space[space]["size"]*uf5_yield2
    if occ_space[space]["system"]=="UF6":
        occ_space[space]["produce"]=uf6_product
        occ_space[space]["yield"]=occ_space[space]["size"]*uf6_yield
        occ_space[space]["produce2"]=uf6_product2
        occ_space[space]["yield2"]=occ_space[space]["size"]*uf6_yield2
    if occ_space[space]["system"]=="UF7":
        occ_space[space]["produce"]=uf7_product
        occ_space[space]["yield"]=occ_space[space]["size"]*uf7_yield
    if occ_space[space]["system"]=="UF8":
        occ_space[space]["produce"]=uf8_product
        occ_space[space]["yield"]=occ_space[space]["size"]*uf8_yield
    if occ_space[space]["system"]=="UF9":
        occ_space[space]["produce"]=uf9_product
        occ_space[space]["yield"]=occ_space[space]["size"]*uf9_yield
```

In [ ]:

```
# For each waste type sum used waste quantities
yields={}
worm=[]
fish=[]
mushroom=[]
smallveg=[]
bigveg=[]
for space in occ_space:
    yields["worm"]={}
    yields["fish"]={}
    yields["mushroom"]={}
    yields["small veg"]={}
    yields["big veg"]={}
    if occ_space[space]["produce"]=="worm":
        worm.append(occ_space[space]["yield"])
    if occ_space[space]["produce"]=="fish":
        fish.append(occ_space[space]["yield"])
    if occ_space[space]["produce"]=="mushroom":
        mushroom.append(occ_space[space]["yield"])
    if occ_space[space]["produce"]=="small veg":
        smallveg.append(occ_space[space]["yield"])
    if occ_space[space]["produce2"]=="big veg":
        bigveg.append(occ_space[space]["yield"])
```

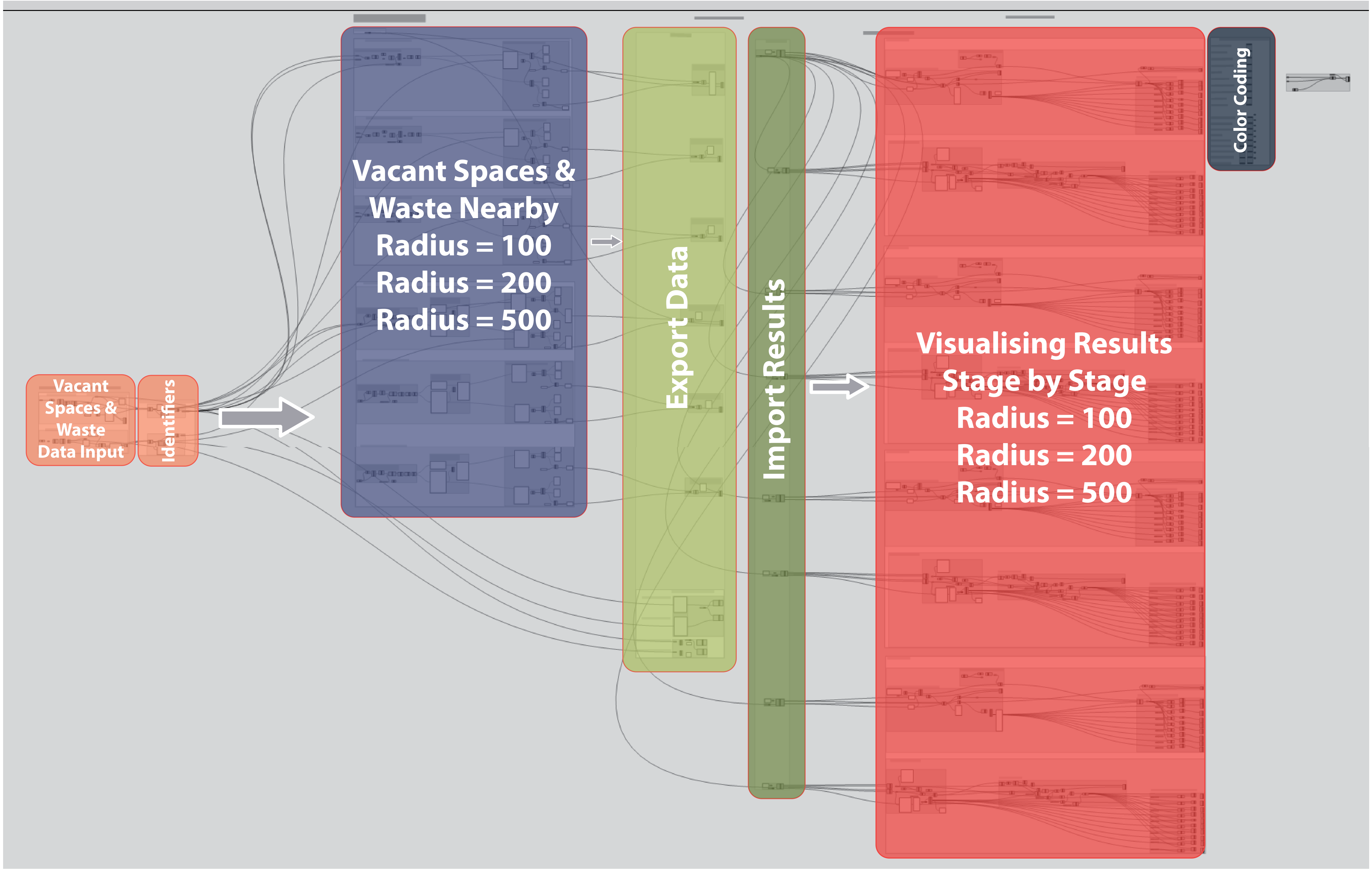
```
# Add total used waste quantities to a list
worm_kg=sum(worm)
fish_kg=sum(fish)
mushroom_kg=sum(mushroom)
smallveg_kg=sum(smallveg)
bigveg_kg=sum(bigveg)

for product in yields:
    yields[product]["number"]=[]
    yields[product]["weight"]=[]
    if product=="worm":
        yields["worm"]["weight"]=worm_kg
        worm_count=int(worm_kg/0.00025)
        yields["worm"]["number"]=worm_count
    if product=="fish":
        yields["fish"]["weight"]=fish_kg
        yields["fish"]["number"]=int(fish_kg/(0.25))
    if product=="mushroom":
        yields["mushroom"]["weight"]=mushroom_kg
        yields["mushroom"]["number"]=int(mushroom_kg/(0.15))
    if product=="small veg":
        yields["small veg"]["weight"]=smallveg_kg
        yields["small veg"]["number"]=int(smallveg_kg/(0.14))
    if product=="big veg":
        yields["big veg"]["weight"]=bigveg_kg
        yields["big veg"]["number"]=int(bigveg_kg/(0.015))
print(yields)
```

In [ ]:

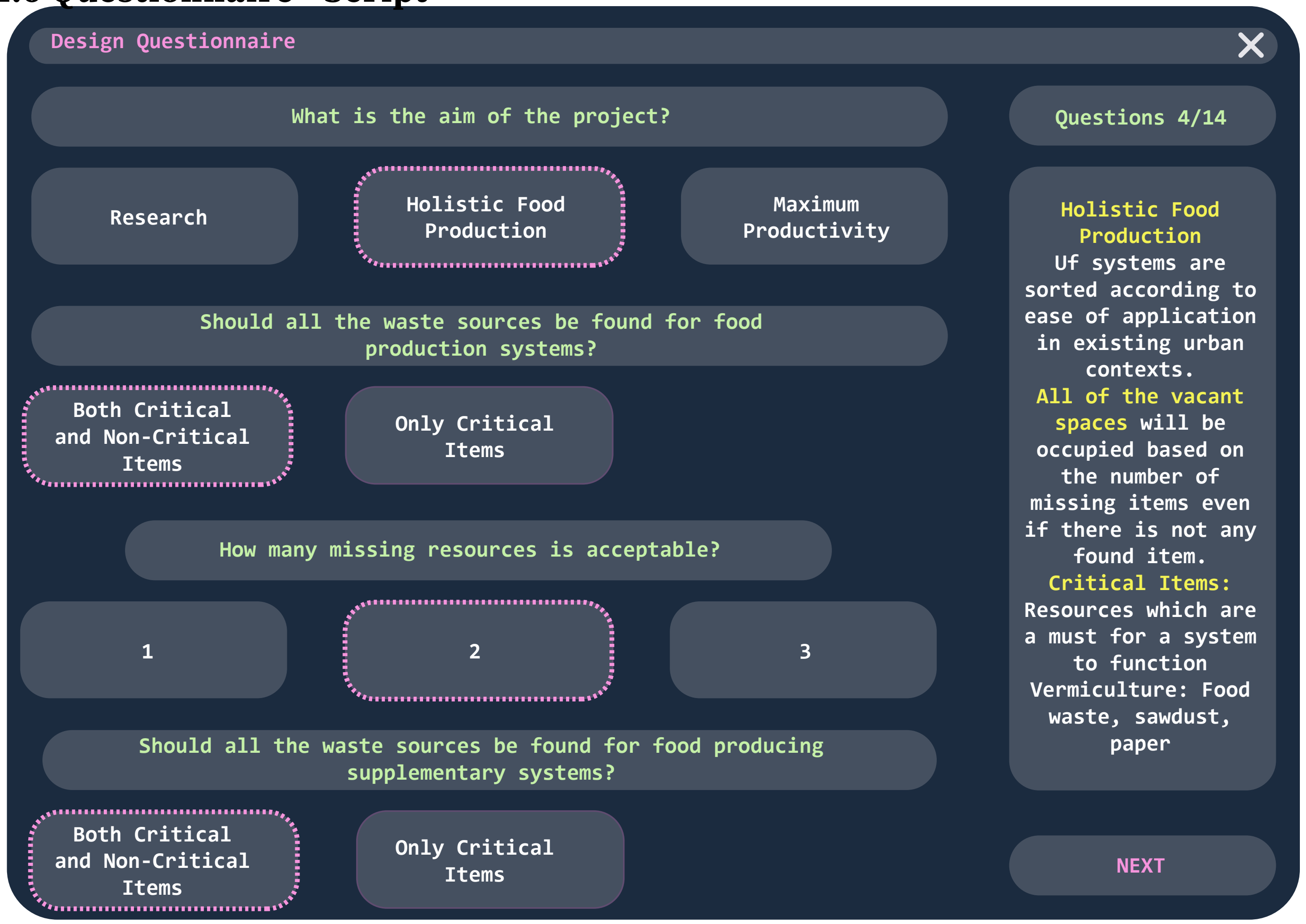
```
# Export data to excel
import pandas as pd
df = pd.DataFrame.from_dict(occ_space, orient='index') # convert dict to dataframe
df.to_csv('UGC space system yield.csv') # write dataframe to file

import pandas as pd
df = pd.DataFrame.from_dict(yields, orient='index') # convert dict to dataframe
df.to_csv('UGC yields.csv') # write dataframe to file
```

In [ ]:

H.5 Grasshopper Script



H.6 Questionnaire - Script



Design Questionnaire

What is the aim of the project?

Research [✓ Holistic Food Production] Maximum Productivity

Should all the waste sources be found for food production systems?

[✓ Both Critical and Non-Critical Items] Only Critical Items

How many missing resources is acceptable?

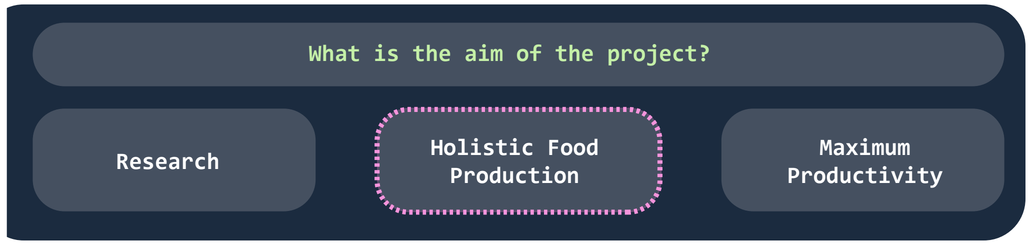
1 [✓ 2] 3

Should all the waste sources be found for food producing supplementary systems?

[✓ Both Critical and Non-Critical Items] Only Critical Items

Questions 4/14

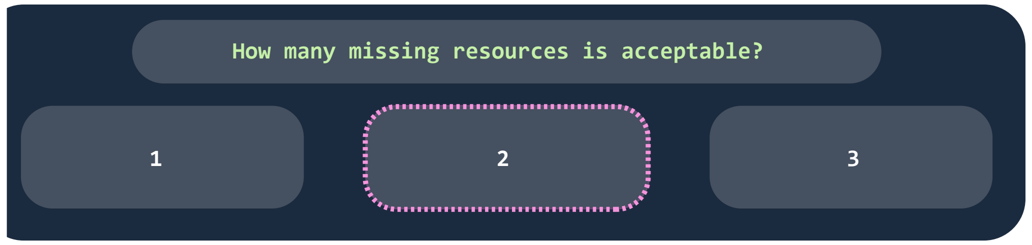
Holistic Food Production
Uf systems are sorted according to ease of application in existing urban contexts.
All of the vacant spaces will be occupied based on the number of missing items even if there is not any found item.
Critical Items:
Resources which are a must for a system to function
Vermiculture: Food waste, sawdust, paper



What is the aim of the project?

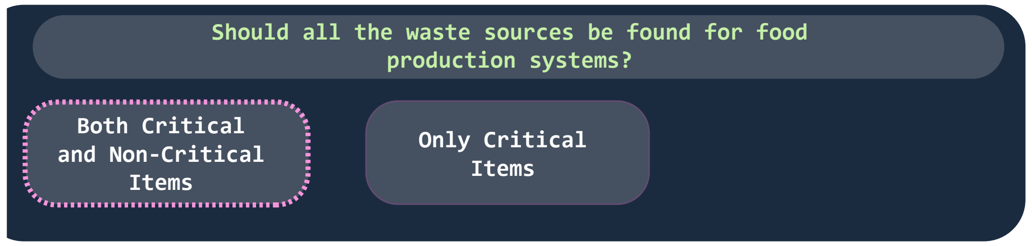
Research [✓ Holistic Food Production] Maximum Productivity

```
#Dictionary of urban farming systems
    # UF1: Vermiculture, UF2: Aquaculture, UF3: Mushroom, UF4: NFT, UF5: Medai Beds, UF6:
Raised Beds, UF7: Water Culture, UF8: Plant Factory, UF9: Aeroponics
    # "S" : supplementary system, "F" : food production system
    # 3 : high, 2 : medium, 1 : low, 0 : none
uf_systems = {
"UF1":{"tag":"UF1","type":"S","weight":3,"solar":1,"in":
["W1","W2","W3","W6"],"supplement":None,"out":["S4","S5"]},
"UF2":{"tag":"UF2","type":"SF","weight":3,"solar":2,"in":["W7"],"supplement":["S5"],"out":
["O4","S2"]},
"UF3":{"tag":"UF3","type":"F","weight":2,"solar":1,"in":
["W2","W3","W4","W6"],"supplement":None,"out":["O3","S4"]},
"UF4":{"tag":"UF4","type":"F","weight":1,"solar":3,"in":["W5","W6","W7"],"supplement":
["S2"],"out":["O1","W1"]},
"UF5":{"tag":"UF5","type":"F","weight":1,"solar":3,"in":["W5","W6","W7"],"supplement":
["S2"],"out":["O1","O2","W1"]},
"UF6":{"tag":"UF6","type":"F","weight":3,"solar":3,"in":["W6"],"supplement":["S4"],"out":
["O1","O2","W1"]},
"UF7":{"tag":"UF7","type":"F","weight":3,"solar":3,"in":["W5","W6","W7"],"supplement":
["S2"],"out":["O1","W1"]},
"UF8":{"tag":"UF8","type":"F","weight":3,"solar":1,"in":["W5","W6"],"supplement":
["S2"],"out":["O1","W1","W7"]},
"UF9":{"tag":"UF9","type":"F","weight":1,"solar":3,"in":["W5","W6","W7"],"supplement":
["S2"],"out":["O1","W1"]}
}
```



How many missing resources is acceptable?

1 [✓ 2] 3

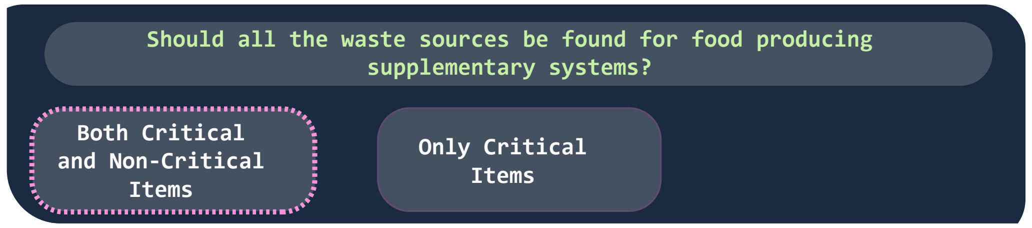


Should all the waste sources be found for food production systems?

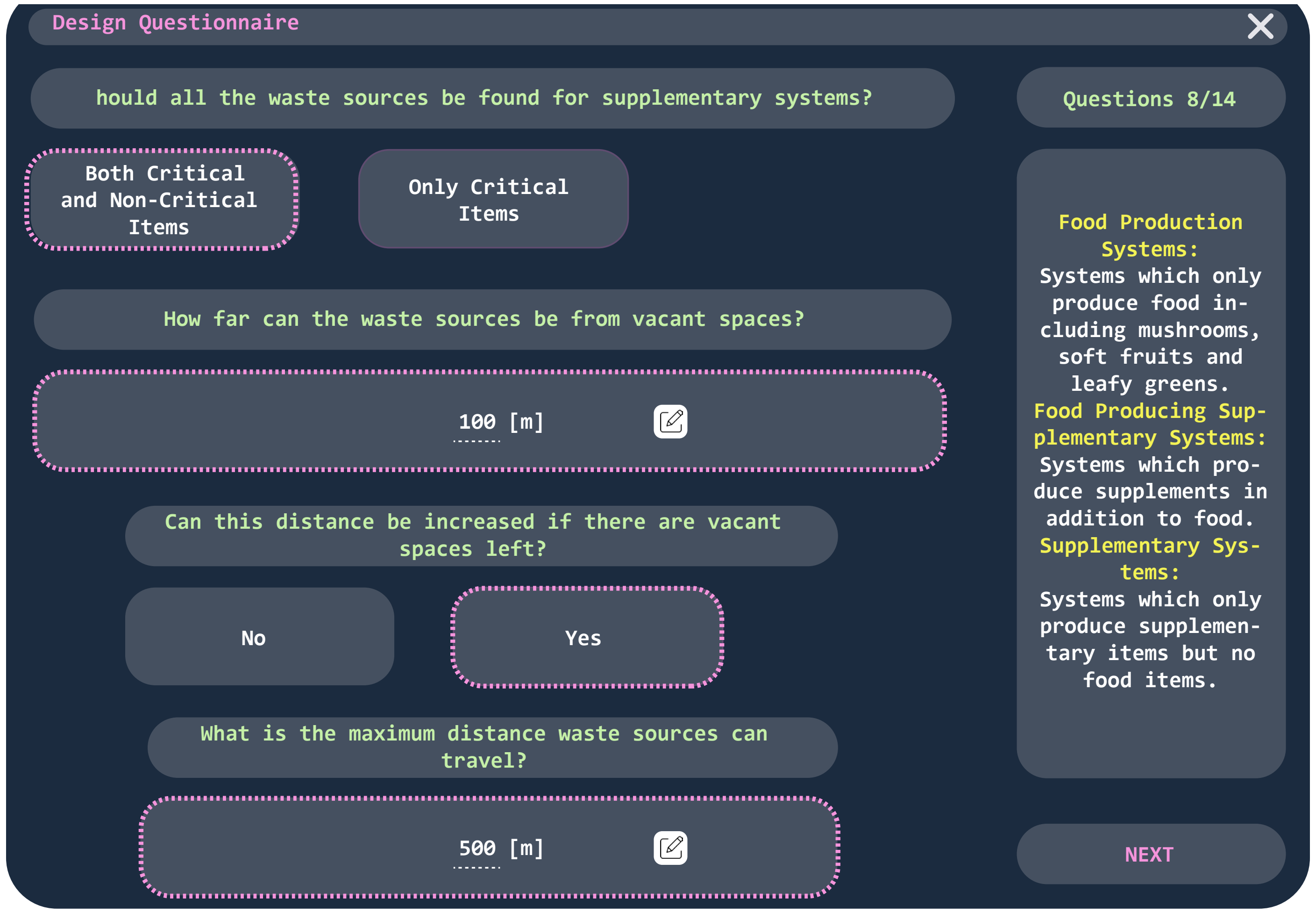
[✓ Both Critical and Non-Critical Items] Only Critical Items

```
occupied={}
used_waste=[]
used_waste_source=[]
used_waste_source_temp=[]
new_edges=[]
occupied_dict={}

for space in sorted_dict:
    print("looking for", space)
    print("for",space,sorted_dict[space],"is possible")
    occupied_dict[space]={}
    occupied_dict[space]["system"]={}
    occupied_dict[space]["found"]={}
    occupied_dict[space]["source"]={}
    occupied_dict[space]["missing"]={}
    for index,system in enumerate(sorted_dict[space]):
        print("looking for system", system)
        occupied[space]={}
        occupied[space]["system"]={}
        found_list=sorted_dict[space][system]["enough waste"]
        sources_list=sorted_dict[space][system]["enough source"]
        missing_list=sorted_dict[space][system]["missing"]
        print(len(missing_list),"is length for",space,system)

        if len(occupied[space]["system"])==0:
            print(space,"is not occupied run for",system)
            if len(missing_list)==0:
                for source in sources_list:
                    if source in used_waste_source:
                        print(used_waste_source,"is used",space,system)
                        pass
                    elif source not in used_waste_source:
                        print(used_waste_source,"is used",space,system)
                        occupied[space]["system"]=system
                        occupied_dict[space]["system"]=system
                        occupied_dict[space]["found"]=sorted_dict[space][system]["enough
waste"]

                        occupied_dict[space]["source"]=sorted_dict[space][system]["enough
source"]
                        for items in found_list:
                            used_waste.append(items)
                        used_waste_source.append(source)
                        edge_tuple=(source,space)
                        new_edges.append(edge_tuple)
                        print("no missing items:", space, system, "assign")
                if len(occupied[space]["system"])>0:
                    print(space,system,"will break")
                    break
            elif len(missing_list)==1:
                print("one item",system,space)
                for missing in missing_list:
                    if missing in non_critical_items:
                        print("one non critical item",missing,system,space)
                        for source in sources_list:
                            if source not in used_waste_source:
                                print("Used Non Critical Source",source)
                                occupied_dict[space]["found"]=sorted_dict[space][system]
["enough waste"]

                                occupied_dict[space]["source"]=sorted_dict[space][system]
["enough source"]

                                occupied_dict[space]["missing"]=sorted_dict[space][system]
["missing"]

                                for items in found_list:
                                    used_waste.append(items)
                                used_waste_source.append(source)
                                edge_tuple=(source,space)
                                new_edges.append(edge_tuple)
                                print("one non critical missing items:", space, system,
"assign")
                            else:
                                print(source,"already used")
                    else:
                        print(space,"critical item missing:",missing,"for",system)
                if len(occupied[space]["system"])>0:
                    print(space,system,"will break")
                    break
            elif len(missing_list)==2:
                print("two items",system,space)
                for missing in missing_list:
                    print("two items",missing)
                    if missing in non_critical_items:
                        print("two items",missing,"not critical")
                        for source in sources_list:
                            if source not in used_waste_source:
                                print("Used Non Critical Source",source)
                                occupied[space]["system"]=system
                                occupied_dict[space]["system"]=system
                                occupied_dict[space]["found"]=sorted_dict[space][system]
["enough waste"]

                                occupied_dict[space]["source"]=sorted_dict[space][system]
["enough source"]

                                occupied_dict[space]["missing"]=sorted_dict[space][system]
["missing"]

                                for items in found_list:
                                    used_waste.append(items)
                                print("two non critical missing items:", space,
system,"assign")

                                print(space,occupied_dict[space])
                                used_waste_source.append(source)
                                edge_tuple=(source,space)
                                new_edges.append(edge_tuple)
                    else:
                        print(space,"critical item missing:",missing,"for",system)
                if len(occupied[space]["system"])>0:
                    print(space,system,"will break")
                    break
```

```
critical_items=["W1","W2","W3", "W4"]
non_critical_items=["W5","W6","W7"]
```


Should all the waste sources be found for food producing supplementary systems?
Both Critical and Non-Critical Items
Only Critical Items

```
# Check if potential sources have necessary waste sources nearby
# Rule 11: For food producing supplementary systems all the items should be found
for space in v_supplement_fs:
    potential=v_supplement_fs[space]["potential source"]
    if len(v_supplement_fs[space]["supplement source"])==0:
        for vacant in potential:
            nearby_list=waste_dict[vacant]
            print(nearby_list)
            for nearby in nearby_list:
                print(vacant, nearby,"is",wastes[nearby]["type"])
                if nearby not in used_waste_source:
                    if wastes[nearby]["size"]==vacant_spaces[space]["size"]:
                        print(nearby,"not used")
                        if v_supplement_fs[space]["fs_demand"]==wastes[nearby]["type"]:
                            print("for",space,vacant,"is potential and has",nearby,"as a
source of",wastes[nearby]["type"])
                            v_supplement_fs[space]["supplement source"]=vacant
                            print(v_supplement_fs[space]["supplement source"])
                            v_supplement_fs[space]["fs_demand source"]=nearby
                            print(v_supplement_fs[space]["fs_demand source"])
                            used_waste_source.append(nearby)
                            edge_tuple1=(nearby,vacant)
                            edge_tuple2=(vacant,space)
                            new_edges.append(edge_tuple1)
                            new_edges.append(edge_tuple2)
                            if len(v_supplement_fs[space]["supplement source"])>0:
                                break
                        elif v_supplement_fs[space]["fs_demand"]!=wastes[nearby]["type"]:
                            v_supplement_fs[space]["supplement source"]=vacant
                            print(v_supplement_fs[space]["supplement source"])
                            v_supplement_fs[space]["fs_demand source"]="supply externally"
                            print(v_supplement_fs[space]["fs_demand source"])
                            used_waste_source.append(nearby)
                            edge_tuple1=(nearby,vacant)
                            edge_tuple2=(vacant,space)
                            new_edges.append(edge_tuple1)
                            new_edges.append(edge_tuple2)
                            if len(v_supplement_fs[space]["supplement source"])>0:
                                break

            if len(v_supplement_fs[space]["supplement source"])>0:
                            break
```



Design Questionnaire

hould all the waste sources be found for supplementary systems?

Both Critical and Non-Critical Items

Only Critical Items

How far can the waste sources be from vacant spaces?

100 [m]

Can this distance be increased if there are vacant spaces left?

No

Yes

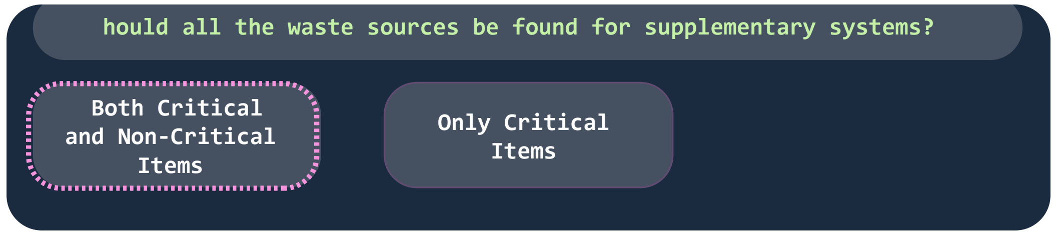
What is the maximum distance waste sources can travel?

500 [m]

Questions 8/14

Food Production Systems: Systems which only produce food including mushrooms, soft fruits and leafy greens.
Food Producing Supplementary Systems: Systems which produce supplements in addition to food.
Supplementary Systems: Systems which only produce supplementary items but no food items.

NEXT

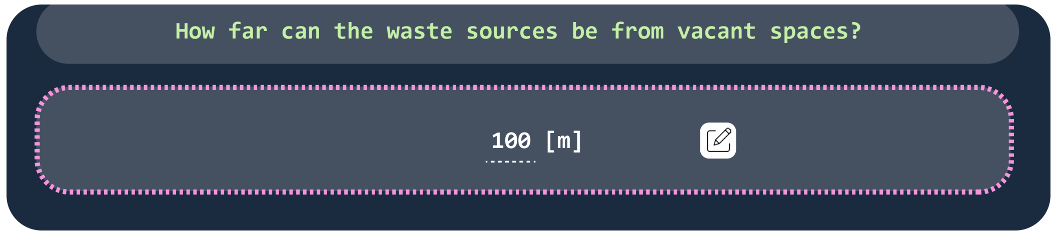


hould all the waste sources be found for supplementary systems?

Both Critical and Non-Critical Items

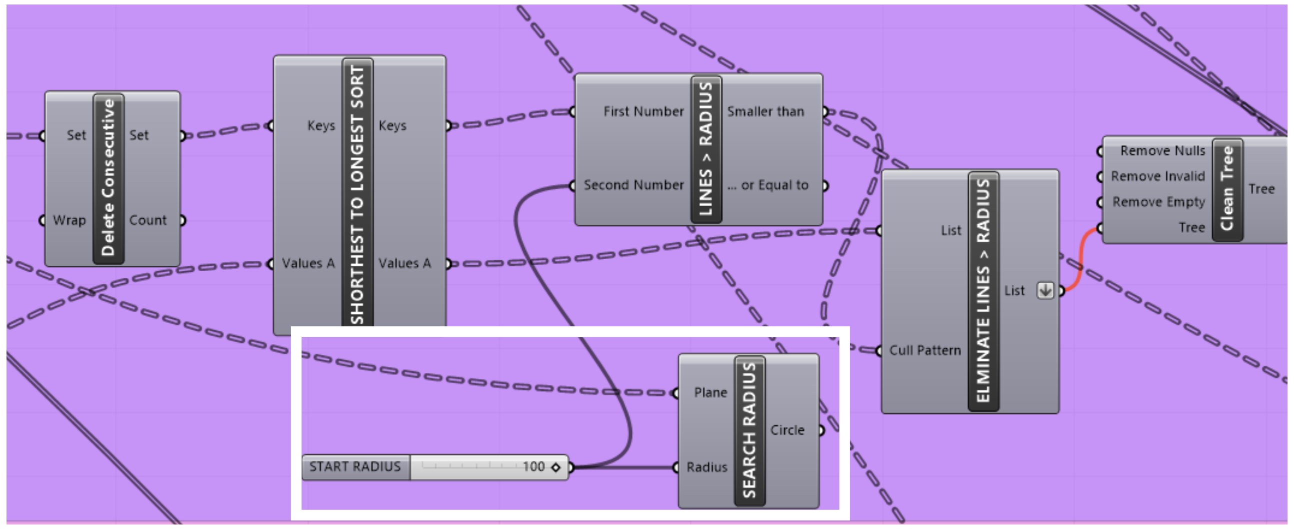
Only Critical Items

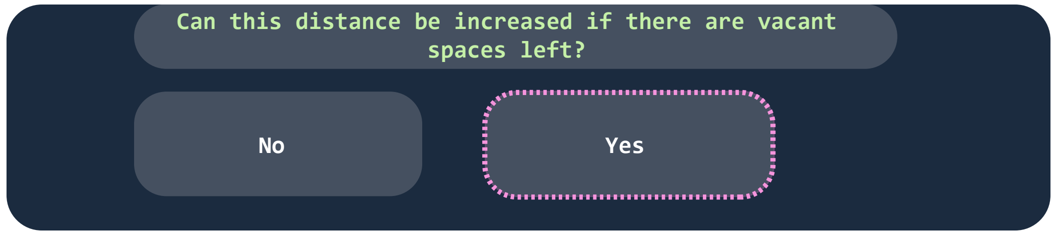
```
# Rule 15: For supplementary systems all the items should be found
for space in v_supplement_fs:
    potential=v_supplement_fs[space]["potential source"]
    if len(v_supplement_fs[space]["supplement source"])==0:
        for vacant in potential:
            found=[]
            found_source=[]
            nearby_list=waste_dict[vacant]
            for nearby in nearby_list:
                if nearby not in used_waste_source:
                    if wastes[nearby]["size"]==vacant_spaces[space]["size"]:
                        found.append(wastes[nearby]["type"])
                        found_source.append(nearby)
                        if v_supplement_fs[space]["fs_demand"]==found:
                            v_supplement_fs[space]["supplement source"]=vacant
                            v_supplement_fs[space]["fs_demand source"]=found_source
                            for x in found_source:
                                used_waste_source.append(x)
                                edge_tuple1=(x,vacant)
                                new_edges.append(edge_tuple1)
                            edge_tuple2=(vacant,space)
                            new_edges.append(edge_tuple2)
                            if len(v_supplement_fs[space]["supplement source"])>0:
                                # print(space,system,"will break")
                                break
            if len(v_supplement_fs[space]["supplement source"])>0:
                            # print(space,system,"will break2")
                            break
```



How far can the waste sources be from vacant spaces?

100 [m]



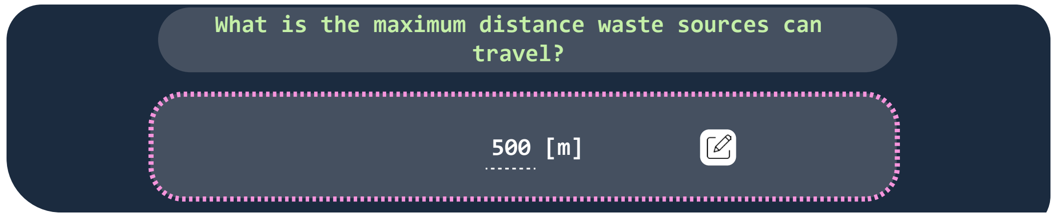


Can this distance be increased if there are vacant spaces left?

Preparation For Increasing The Search Radius

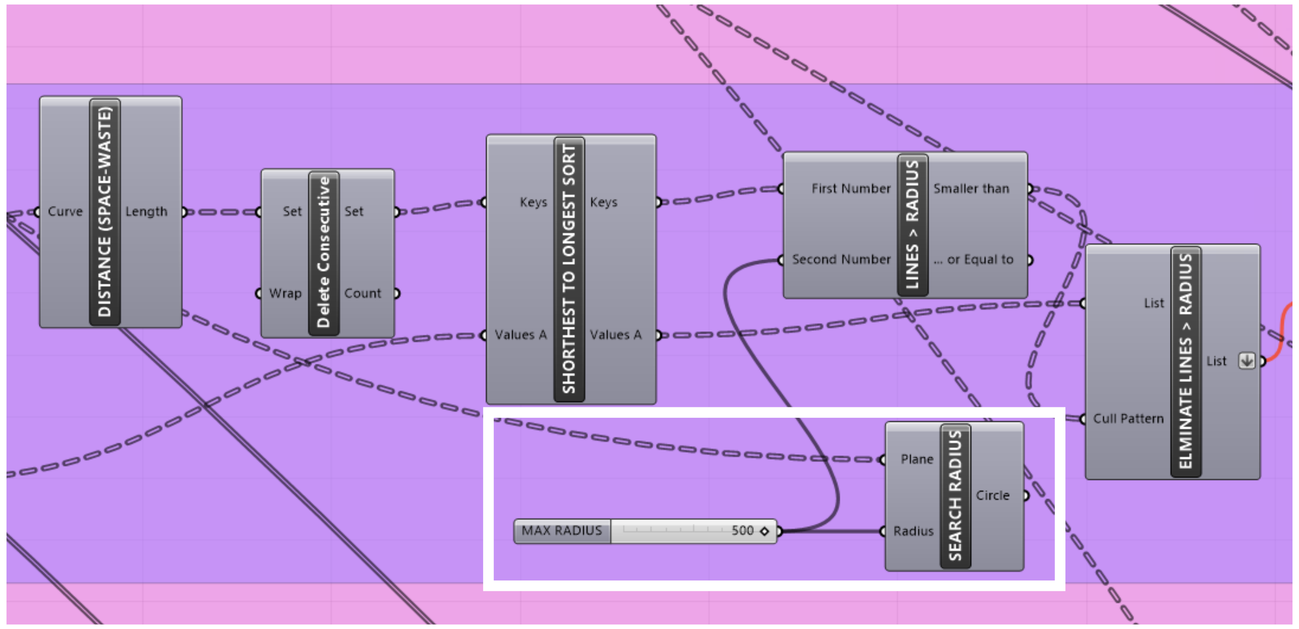
In [ ]:

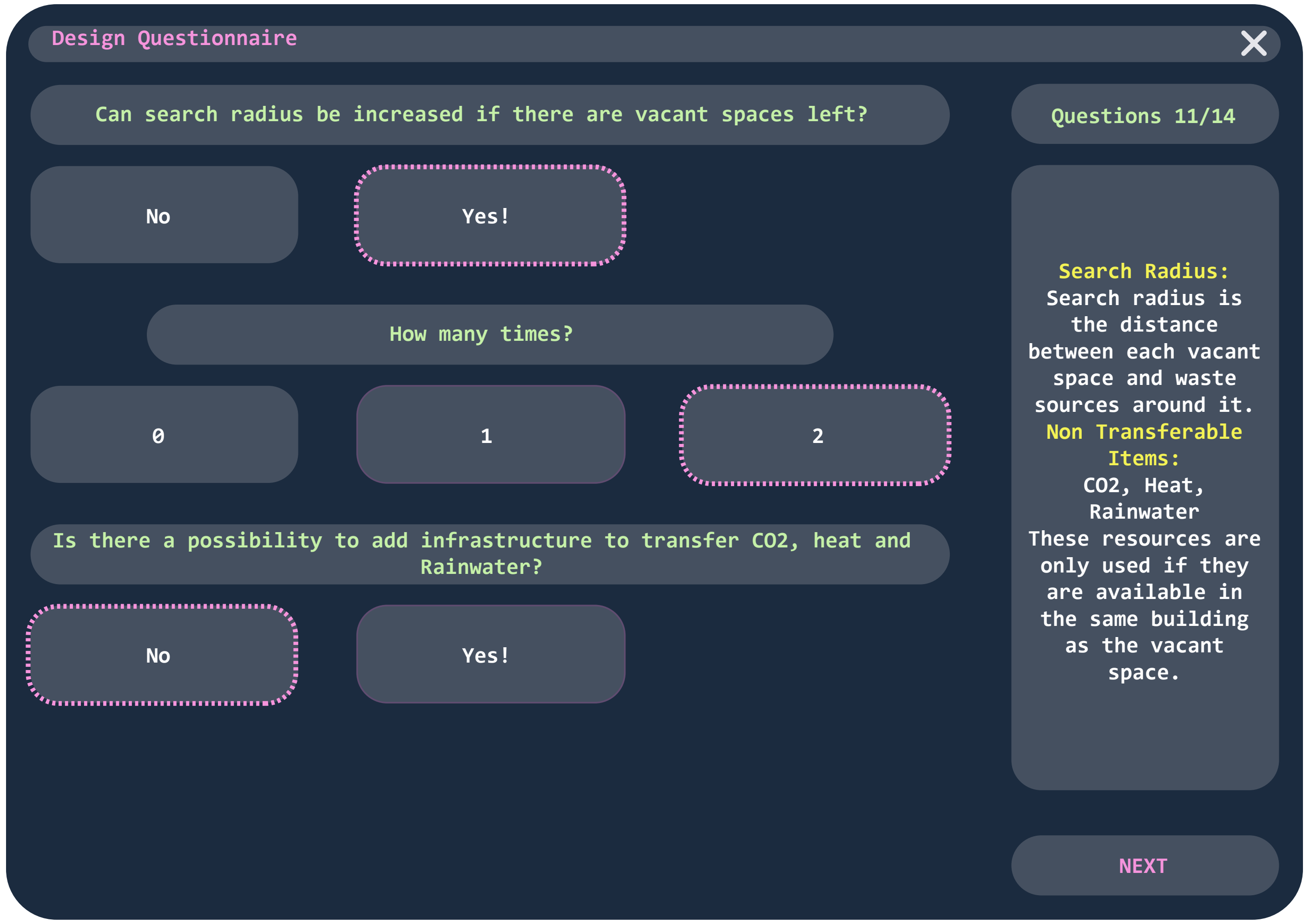
```
# Export still vacant spaces, list of used waste sources, list of still available waste
sources to be used in next stage after increasing search radius
vacant_spaces2=vacant_spaces.copy()

for space in occupied_dict:
    vacant_spaces2.pop(space)

file = "vacant spaces_bigger radius.txt"
with open(str(file), 'w') as outfile:
    try:
        json.dump(vacant_spaces2, outfile)
        print(file + " has been updated successfully")
    except:
        print("Problem with updating file")


wastes2={}

for waste in wastes:
    if waste not in used_waste_source:
        wastes2[waste]={}
        wastes2[waste]["location"]=wastes[waste]["location"]
        wastes2[waste]["source"]=wastes[waste]["source"]
        wastes2[waste]["tag"]=wastes[waste]["tag"]
        wastes2[waste]["type"]=wastes[waste]["type"]
        wastes2[waste]["size"]=wastes[waste]["size"]
        wastes2[waste]["node_type"]=wastes[waste]["node_type"]

file = "available waste 2.txt"
with open(str(file), 'w') as outfile:
    try:
        json.dump(wastes2, outfile)
        print(file + " has been updated successfully")
    except:
        print("Problem with updating file")

file = "used waste.txt"
with open(str(file), 'w') as outfile:
    try:
        json.dump(used_waste_source, outfile)
        print(file + " has been updated successfully")
    except:
        print("Problem with updating file")
```



What is the maximum distance waste sources can travel?

500 [m]





Design Questionnaire

Can search radius be increased if there are vacant spaces left?

No [✓ Yes!]

Questions 11/14

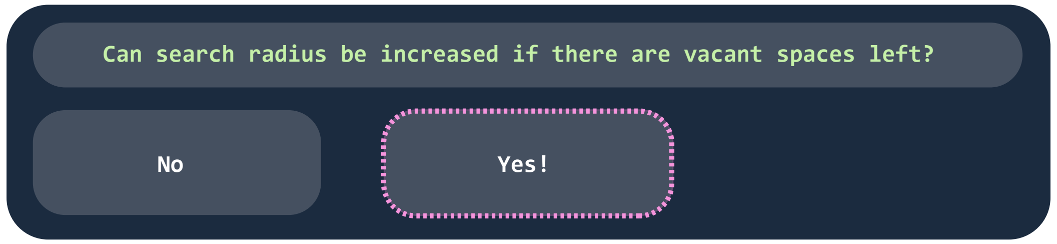
How many times?

0 1 [✓ 2]

Is there a possibility to add infrastructure to transfer CO2, heat and Rainwater?

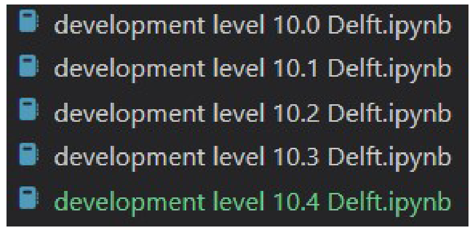
[✓ No] Yes!

Search Radius: Search radius is the distance between each vacant space and waste sources around it.
Non Transferable Items: CO2, Heat, Rainwater
These resources are only used if they are available in the same building as the vacant space.



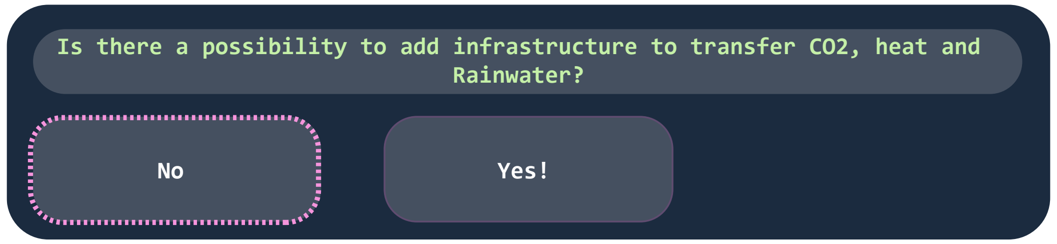
Can search radius be increased if there are vacant spaces left?

No [✓ Yes!]



Stage 1 : radius = 100m	development level 10.0 Delft.ipynb
Stage 2 : radius = 200m	development level 10.1 Delft.ipynb
Stage 3 : radius = 500m	development level 10.2 Delft.ipynb
Stage 4 : radius = 500m occuppy all nodes	development level 10.3 Delft.ipynb
Stage 5 : statistics	development level 10.4 Delft.ipynb

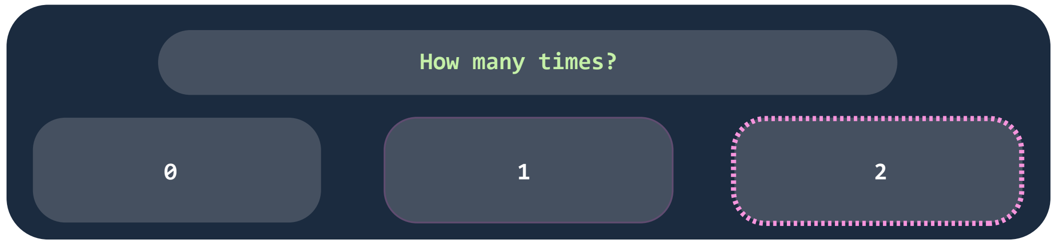
NUMBER OF STEPS DEFINE NUMBER OF STAGES



Is there a possibility to add infrastructure to transfer CO2, heat and Rainwater?

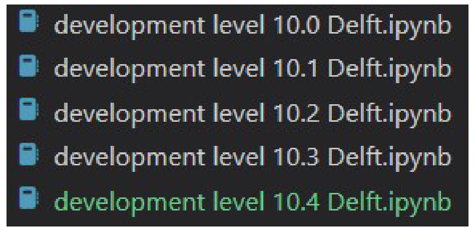
[✓ No] Yes!

```
# Rule 2: If CO2, Heat and Rainwater are not in the same building as the vacant space,
they cannot be used.
# Remove if vacant space is in a different building than the waste source (only for CO2,
Heat and Rainwater)
for couple_count in range(len(nearby_waste100)):
    for couple in nearby_waste100:
        for index,item in enumerate(couple):
            if couple[1] in wastes:
                if vacant_spaces[couple[0]]["building"]!=wastes[couple[1]]["source"]:
                    print(couple,vacant_spaces[couple[0]]["building"],wastes[couple[1]]
["source"],wastes[couple[1]]["type"])
                    if (wastes[couple[1]]["type"]=="W5") or (wastes[couple[1]]
["type"]=="W6") or (wastes[couple[1]]["type"]=="W7"):
                        if couple in nearby_waste100:
                            print(couple,wastes[couple[1]]["type"])
                            print("removed",couple,wastes[couple[1]]
["type"],vacant_spaces[couple[0]]["building"],wastes[couple[1]]
["source"],wastes[couple[1]]["type"])
                            nearby_waste100.remove(couple)
            else:
                print(couple,"not in nearby_waste100")
```



How many times?

0 1 [✓ 2]



Stage 1 : radius = 100m	development level 10.0 Delft.ipynb
Stage 2 : radius = 200m	development level 10.1 Delft.ipynb
Stage 3 : radius = 500m	development level 10.2 Delft.ipynb
Stage 4 : radius = 500m occuppy all nodes	development level 10.3 Delft.ipynb
Stage 5 : statistics	development level 10.4 Delft.ipynb