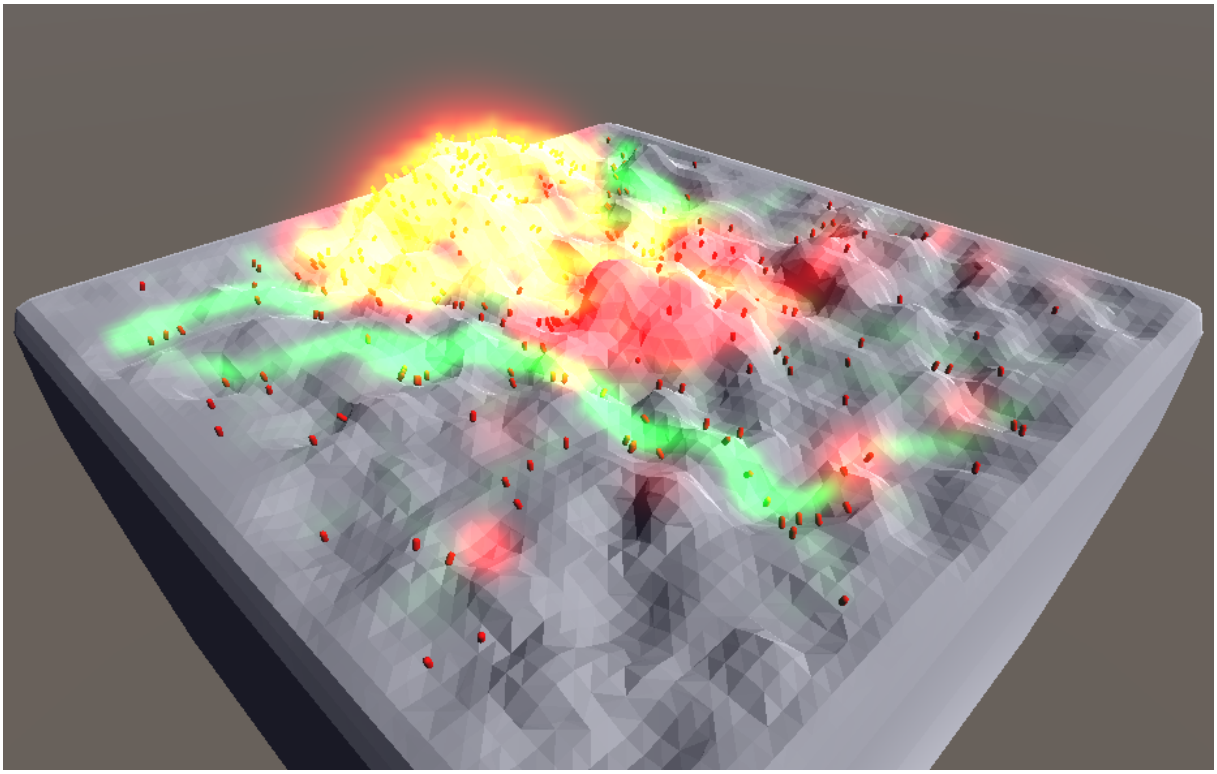
TermiteSim

Developing a termite-like builder swarm simulator for experimentation and validation of control strategies.



MSc Thesis By Jeroen Lieveloo

26 April 2023

TU Delft Integrated Product Design

Chair: Dr. Jordan H. Boyle

Mentor: Dr. Jered H. Vroon

Abstract

Automation in the form of swarm robots is an increasingly feasible opportunity for solving complex physical problems. However, how such a swarm is to be instructed and controlled is one of the many topics that requires further research. For this, we can look to nature for inspiration, as there are many nest building swarms. Perhaps the most notable of which are the mound building termites. These termites build incredibly intricate mounds that contain a royal chamber, nurseries, fungal gardens, covered walkways, and even ventilation shafts to thermoregulate the inside of the mound. There are some attempts to capture this behaviour in custom simulators. The problem is that these simulators have limited capabilities. That is why a new simulator specifically for termite-like builder robot swarms is required for further research in this field. In this project, the different requirements for such a simulator are identified and used to develop an open-source state of the art simulator, TermiteSim. TermiteSim is designed for experimenting with different control strategies and tuning the simulation parameters to recreate the desired behaviour. In other words: a tool for researchers to design desired behaviour of a builder robot swarm. In this thesis TermiteSim is used to simulate several termite-like building behaviours described in the literature and the resulting structures are compared to those of other simulators, which TermiteSim was successfully able to recreate.

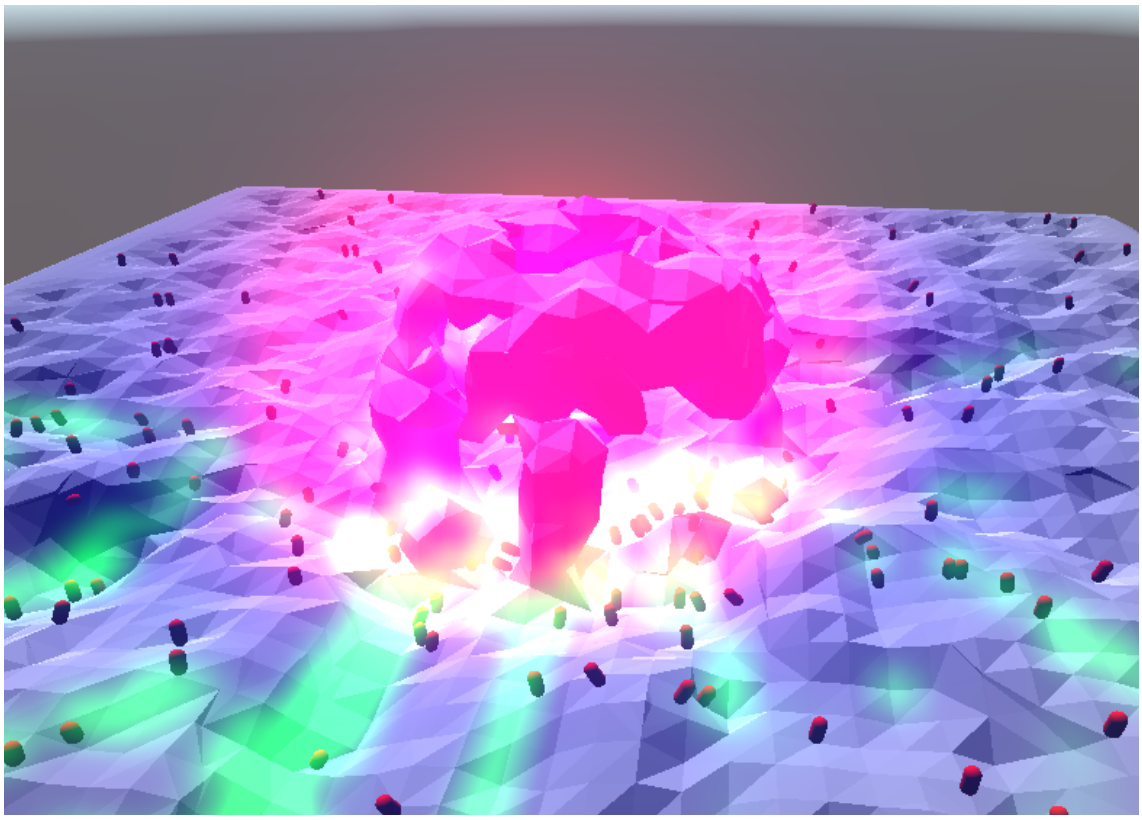


Figure 1. Preview of TermiteSim showing the creation of a royal chamber with entrances, and the pheromones that led to this construction.

Table of Content

1. Introduction

Despite the construction industry being a 10.7$ trillion global output industry (Oxford Economics, 2021), it has yet to automate like other industries have. Most of the work in construction robotics focuses on 3D printing with a robotic arm that deposits concrete in layers on top of each other to create the desired structure. An alternative approach is swarm robotics. Swarm robotics offers several advantages over traditional approaches to robotics. For example, it allows for better scalability, as adding more robots to the swarm does not significantly increase the complexity of the system. Swarm robotics also provides robustness, as the absence of a central control unit makes the system less susceptible to failure. In addition, swarm robotics allows for greater adaptability, as the robots in the swarm can adapt to changes in the environment in a decentralized way.

This thesis looks at mound building termites as inspiration for robot swarms in construction. Mound building termites are social insects that live in highly organized colonies and are known for their remarkable ability to construct intricate mounds. One of the most interesting aspects of termite mounds is their architectural design. The interior of the mound is hollow, with multiple chambers that serve different purposes, such as the nursery galleries, the royal chamber, and fungal gardens. The construction of mounds by termites is an amazing feat of engineering, and scientists have been studying these insects for decades to understand the mechanisms behind their coordination. The current understanding of the underlying principles is still incomplete, and there being some disagreements about these mechanisms. The traditional explanation for the coordination observed in termites focuses on the idea of stigmergy as first proposed by Grasse in his 1959 paper, which is a method of communication. Namely, indirect communication through changes in the environment. This can be topological heterogeneities by placing or removing material as well as pheromone gradients through the sensing and secretion of pheromones as described in Section 1.4.2. These mechanisms alone lead to simple behaviour of termites following each other, making holes at excavation sites and building piles of soil. But together, they lead to the complex emergent behaviour of mound building. A behaviour that no single rule can accomplish.

To study the complex behaviour of termites and test our understanding of it, theoretical and computational models can be a powerful tools. These models simplify the context of termites and boil their behaviour down to a set of equations or behavioral rules. Studying these models has provided valuable insight into the mechanisms behind swarm intelligence and collective decision-making and has helped us better understand how to apply the concept of swarm intelligence in swarm robotics. However, more research is needed as there are still many unknowns of the underlying mechanisms of their behaviour. Previous models and simulations led to new insights in termite building behaviour. But there is potential for a lot more to be discovered through the experimentation with simulations. Previous simulations show that we can recreate termite-like architecture with the emergent behavior of termite-like agents, but how can we leverage similar mechanisms to create something else? What if we want to build something that is not a termite mound? The outcome of emergent behaviour is by definition undefined, and the more complex the behavioral rules, the less predictable the result will be.

To solve this we can once again draw inspiration from nature, but this time in another form: evolution. Just like termites as a species went through thousands of years of trial and error to come to an emergent building behaviour that meets their needs, so can we iterate on behavioral rules that result in the structure we want to create. Preferably much faster then evolution. The problem is that, currently, there are no robot swarm simulators that are suitable for simulating termite like building. That is why a new simulator specifically for termite like builder robot swarms is required for further research in this field. This project aims to create an open-source simulator of a termite-like builder robot swarm for experimenting with different control strategies and adjusting them accordingly.

1.1. Robotic Construction

Everyone can relate to the frustrations of a construction project in the neighborhood; blocked roads, loud noises, and they often take years before they are finished.

The construction industry with a 10.7$ trillion global output is expected to grow even further by 42% to reach 15.2$ trillion by 2030 (Oxford Economics, 2021). Despite its size, the construction industry has yet to automate like other industries have. However, this trend is shifting as robotics technology advances. Automating construction makes it possible for robots to take over many of the dangerous, dirty and dull task that are required. Most of the work in construction robotics focuses on 3D printing with a robotic arm that deposits concrete in layers on top of each other to create the desired structure. These robot arms are automated, following the predefined instructions given to it. Allowing it to build a structure with minimal assistance. An example of this type of 3D printed construction is Von Perry, which aims to tackle the global housing shortage to efficiently build cost-effective houses. They used Cobod's BOD2 3D printer to 3D print the first house to get the same certification as regular houses require for living as seen in Figure 2.

Figure 2. Von Perry (2021). 3D printing a house. https://automate.construction/2021/03/30/the-most-innovative-3d-printed-house-in-the-world/

Such approaches are interesting but are also quite limited. One limitation is the size of the printing area. Which is limited to the reach of the robot arm or printer construction. Another limitation is the shape of the structure which has to be designed as such that it can be 3D printed. 3D printers are limited in their ability to build horizontal structures without support material. Von Perry solves this problem by manually placing metal strips to bridge the gaps where needed. However, there are other approaches that don't utilize a single sophisticated robot as their main method for construction.

1.2. Swarm robotics

Inspired by the behaviour and collective intelligence of social animals such as ants, bees, and termites, swarm robotics refers to the use of this decentralized approach to control a group of robots. The robots in the swarm are designed to communicate with each other and coordinate their actions to achieve a common goal. The coordination is achieved through simple rules and local interactions, without the need for a central control unit. These simple rules can cause the emergence of complex behaviour (J. F. Boudet, et al. 2021), which is the principle behind swarm intelligence.

Swarm robotics offers several advantages over traditional approaches to robotics. For example, it allows for better scalability, as adding more robots to the swarm does not significantly increase the complexity of the system. Swarm robotics also provides robustness, as the absence of a central control unit makes the system less susceptible to failure. In addition, swarm robotics allows for

greater adaptability, as the robots in the swarm can adapt to changes in the environment in a decentralized way.

There are two main types of swarm robotics: homogeneous and heterogeneous swarm robotics. Homogeneous swarm robotics involves many identical robots that perform the same tasks. In this type of swarm, the robots work together to achieve the goal, relying on their collective intelligence to solve problems and make decisions. One advantage of homogeneous swarm robotics is that they have built in redundancy, making them highly robust. Since all robots are identical, it is easy to swap out robots that are damaged or malfunctioning without affecting the overall performance of the swarm. An example of a homogenous swarm is the Kilobot swarm seen in Figure 3, a small 14$ open-source robot.

Heterogeneous swarm robotics involves a group of robots with different capabilities and strengths. In this type of swarm, robots are assigned tasks based on their abilities, allowing for a more efficient system. An example of a heterogenous swarm is the Swarmanoid project by Dorigo m., et al. 2012. In Figure 4 the Swarmanoid project can be seen where 3 foot-bots are carrying 1 hand-bot.

Figure 3. Asuscreative (2014). Kilobot robot swarm [photograph]. Wikipedia. https://commons.wikimedia.org/wiki/File:Kilobot_robot_swarm.JPG

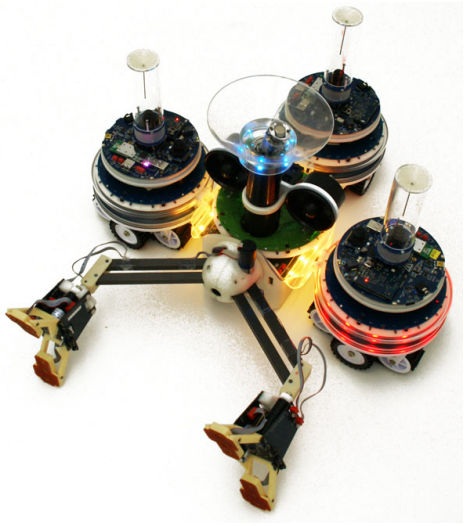


Figure 4. Swarmanoid project (2012). Interaction between real foot-bots and one hand-bot [photograph]. Swarmanoid. https://www.swarmanoid.org/swarmanoid_hardware.php.html

1.3. Swarm robotics construction

These ideas are now starting to be used in robotic for construction. One such example is HyperTunnel. Traditionally, tunnels are created using the drill and blast method, which includes drilling and controlled explosives. Otherwise, large tunnel boring machines are used which often weight thousands of tonnes and use a giant cutting head to bore their way through the ground. HyperTunnel takes a new approach to tunnel construction which employs swarms of robots that build the walls of the tunnel first and then later excavating the material inside the tunnel. They do this by drilling holes in the target area which are filled with special bore pipes in which the robots are placed. Those robots then begin to construct the shell of the tunnel by drilling access points along the pipes and use those to carve chambers and inject a composite material into the earth around the pipe to create the tunnel walls. This process utilizes the parallelism of swarm robotics which allows tunnels to be created up to 10 times faster and at half the cost of conventional methods.

However, this is still an early implementation of swarm robotics where still a lot of preparation has to be done, such as limiting the swarm to predefined paths by manually placing the pipes in which they move. Ideally, any preparation would not be needed or could also be done autonomously.

This is what project such as the Termes project are focussing on. They build a multi-robot autonomous system that can replicate a specified structure with specialized tiles. This is done by compiling the desired structure into a set of behavioral rules which the robots follow. This then leads the robots to build the desired structure. They demonstrate this process both in simulation and with their own robots. However, their approach is limited to building in a grid and relies on their specialized prebuilt tiles for building and navigation. Furthermore, it can not create any sort of overhang, as the tiles are simply stacked on top of each other and are not secured any further. That is still a long way off from how actual termites work. (Petersen K., el al. 2011.)

1.4. Inspiration from nature: Termites

Mound building termites are social insects that live in highly organized colonies and are known for their remarkable ability to construct intricate mounds. Mound building termites belong to the family Termitidae and are found in various regions of the world, including Africa, Australia, and South America. Their termite colonies are made up of different castes, each with its own specific role. The queen and the king are responsible for reproduction, while the workers take care of foraging, food supply, and construction of the mound. Lastly, the soldiers are responsible for protection against predators. All members work together in a highly organized manner to ensure the survival of the colony. One of the most interesting aspects of termite mounds is their architectural design. Mounds are constructed in a way that provides protection and ventilation for the colony. The interior of the mound is hollow, with multiple chambers that serve different purposes, such as the nursery galleries, the royal chamber, and fungal gardens (Figure 5). These mounds can reach heights of up to 9 meters and can be several meters wide (Figure 6). The material used to construct the mound is primarily made up of soil, which the workers collect and transport to the construction site. The soil is mixed saliva, and excrement to create the paste that is used for building.

Figure 5. (n.d.). Termite mound [Render]. DKfindout. https://www.dkfindout.com/us/animals-and-nature/insects/insect-colonies/

Figure 6. Brewbooks. (2009). Cathedral Termite Mound [Photograph]. Flickr. https://www.flickr.com/photos/93452909@N00/3491333666/

The construction of mounds by termites is an amazing feat of engineering, and scientists have been studying these insects for decades to understand the mechanisms behind their coordination. The current understanding of the underlying principles is still incomplete, and there being some disagreements about these mechanisms. What is known is that termites have the tendency to wander around, dig, form pellets from soil, and deposit those pellets. Sane S., Et al. 2020 describes such tendencies as 'Fixed action patterns' which are fixed innate behaviour triggered by certain stimuli that trigger these pre-programmed responses.

1.4.1. Stigmergy

The traditional explanation for the coordination observed in termites focuses on the idea of stigmergy as first proposed by Grasse in his 1959 paper, which is a method of communication. Namely, indirect communication through changes in the environment. This can be topological heterogeneities by placing or removing material as well as pheromone gradients through the sensing and secretion of pheromones as described below. These changes act as stimuli to trigger the before mentioned 'fixed action patterns' (Perna A., Theraulaz G. 2017).

1.4.2. Pheromones

Pheromones are one of the major mechanisms that facilitate mound building behaviour. These pheromones are secreted by the termites and can stimulate or inhibit certain behaviour. This has extensively been researched in Bruinsma's 1979 thesis which contains a detailed description of termites behaviour in a multitude of scenarios. He argues that there are a three pheromones that play a significant role in construction: the queen pheromone, the cement pheromone, and the trail pheromone. Although the exact effects of all of these pheromones are not fully understood, some fixed action patterns are proven to be triggered by the presence of these pheromones. A summary of the observed effects of pheromones can be seen in Table 1. Here the observed distance that the termite is attracted to move towards the pheromone, stimulated to deposit, and stimulated to pick up new material, is described. If no specific distance is observed, it will simply say yes, stimulate or inhibit. If no observation regarding the effect was described, the space is left empty.

In Table 2 can be seen how these observations were interpreted for the project. Here the attraction is based on whether the termite is carrying a soil pellet, and where in the gradient it is stimulated to deposit that soil pellet, and where in the gradient it is stimulated to pick up new soil pellets.

	Attraction	Deposit	Pickup
Queen	Yes	2 - 5 cm	0 - 0.5 cm
Cement	1.5 cm	Stimulate	Stimulate
Trail	0.5 - 0.8cm?	Inhibit	-

Table 1. Observations of the effects of pheromones

	Attraction	Deposit	Pickup
Queen	If holding material	At edge of gradient	At center of gradient

	Attraction	Deposit	Pickup
Cement	If holding material	At center of gradient	-
Trail	Yes	At edge of gradient	-

Table 2. Proposed effect of pheromones

Queen pheromone

In a termite mound, the royal chamber is a vault in the center of the mound that houses the king and queen. In order to research the emergence of the royal chamber, Bruinsma placed a queen on top of some soil in a petri dish with worker termites and described the following sequence of events in the behavior of the workers in his 1979 paper:

1. *"Grasping of soil pellets near the queen;*
2. *Transport of the pellets to the site of deposition, a zone around the queen located approximately 2 - 5 cm distance from her;*
3. *Deposition and cementing of the soil granules somewhere in that zone. After about 40-60 min of building activity, workers start to concentrate their depositions in one or more specific areas in the deposition zone. This leads to the construction of incipient pillars or columns, at 1.9 - 2.6 cm from the queen. These pillars are lengthened until they reach a certain height of 0.5 - 0.8 cm. Building workers then change the direction of building at the pillar apex in a lateral sense: the formation lamellae. The growing lamellae are extended and connected to one another, to form a roof over the queen, while pillars are connected to form a wall."*

He proposes that this is behaviour is stimulated by the queen pheromone that is excreted from the queens body. Further experiments that manipulate the distribution of the queens pheromone conform this hypothesis. However, whether the deposition of soil pellets is stimulated within a range window of density of queen pheromone or after threshold density of queen pheromone is reached is still unclear. One experiment indicates that it is of probabilistic nature, as seen in Figure 7, where the ratio of grasping and depositing of soil pellets is shown to be proportional to the distance from the queen.

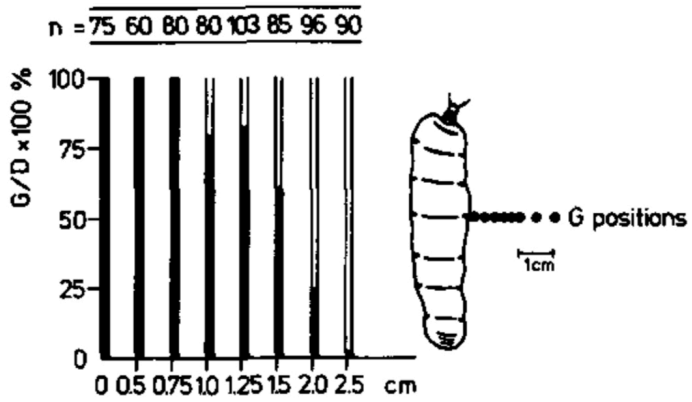


Figure 7. O. H. Bruinsma, (1979). The ratio between grasping (G) and deposition (D) behaviour with respect to metal spheres (diameter: 0.2) placed at defined distances from the queen. n: the number of workers displaying a building response. Data are based on a combination of two tests, each employing 160 major workers.

Cement pheromone

When a termite picks up soil it will knead it with its mandibles and transform it into a paste by mixing it with its saliva which contains this cement pheromone. When the termite places the pellet, this cement pheromone will diffuse into the air around the pellet, attract other termites within 2cm and stimulates them to deposit their soil-pellets on top of it (Bruinsma, 1979), (Grasse, 1959). This causes a positive feedback loop that results in what Bruinsma describes as 'deposition zones'. These deposition zones, over time, turn into pillars that are connected by lamellae to form the walls of the mound.

However, Green et al. 2017 casts doubt on this theory and proposes in his research that it is not the aggregated disposition of soil pellets, but rather that the termites themselves tend to aggregate to excavation sites at the edge of which they deposit the material. Furthermore, Calovi D., et al.2019 argues that it is instead the local surface curvature of the terrain that stimulated the deposition of soil, which would also have a positive feedback loop mechanism.

Trail pheromone

(Mitaka Y., Akino T. 2021) The trail pheromones is a trail that is continuously excreted by worker termites and placed underneath them on top of the soil. The trail pheromone attracts nearby termites to walk towards it resulting in a positive feedback loop that causes the emergence of trails that act as paths for the termites to navigate between the mound, deposition zones and other points of interest. This serves as a means for directional orientation.

An increase in trail pheromone around the foot of a pillar increases the height of the pillar before it is laterally expanded with lamellae (Bruinsma, 1979), which indicates that the presence of trail pheromone also inhibits deposition. Which would logically also prevent them from blocking their tunnels.

1.4.3. Other Stimuli

Bruinsma's 1979 thesis also indicates that there are other stimuli that play a role in construction. He experimented with tactile stimuli (small metal globes) and found that they stimulated deposition. He also shows that the difference in air between the inside of the mound and outside of the mound is used for orientation. This is concluded from the way worker termites repair small holes in the outer lamellae walls of the mound. This indicates that there are other stimuli besides pheromones that influence termite building behaviour. Other studies have also shown that termites also respond to a range of cues in their environment (Bonabeau E., et al. 1998) including physical cues such as the curvature of the surface they are working on (Calovi D., et al. 2019), and environmental cues such as temperature and humidity (Ockoa S., et al. 2019). Khuong, A. Et al. (2016) has also shown that termites use body templating, where the termite uses its own body size as a 'ruler' for determining at what height the transition of the walls to ceiling should occur.

1.4.4. Emergent behaviour

These mechanisms alone lead to simple behaviour of termites following each other, making holes at excavation sites and building piles of soil. But together, they lead to the complex emergent behaviour of mound building. A behaviour that no single rule can accomplish. Emergent behaviour, however, does not necessarily lead to beneficial outcomes.

The fact that this emergent behavior is positive in the case of termites is a product of evolution rather than by definition. As a theoretical example: if deposition would also be stimulated by the trail pheromone, all the entrances and exits to the mound would likely be closed-off. Trapping a part of the termites inside and locking the other part out. This would be very detrimental for their survival.

1.4.5. Models

To study the complex behaviour of termites and test our understanding of it, theoretical and computational models can be a powerful tools. These models simplify the context of termites and boil their behaviour down to a set of equations or behavioral rules.

Deneubourg, in his 1977 paper, was the first to create a mathematical model explaining the emergence of the regular spacing between pillar formation. He later wrote another paper in 1995 outlining a set of differential equations that model the size regulation of the mound in response to changes in the termite population. Bonabeau et al. built upon the pillar spacing paper of Deneubourg by creating a model from partial differential equations (Figure 8), showing the effects of a worker population on builder activity in a 1D and 2D space (Bonabeau, 1998). One of Bonabeau's findings was that a response-threshold function such as $p = \frac{s^n}{s^n + \theta^n}$ can explain the statistical relation between pheromone concentrations and actions that the termites take.

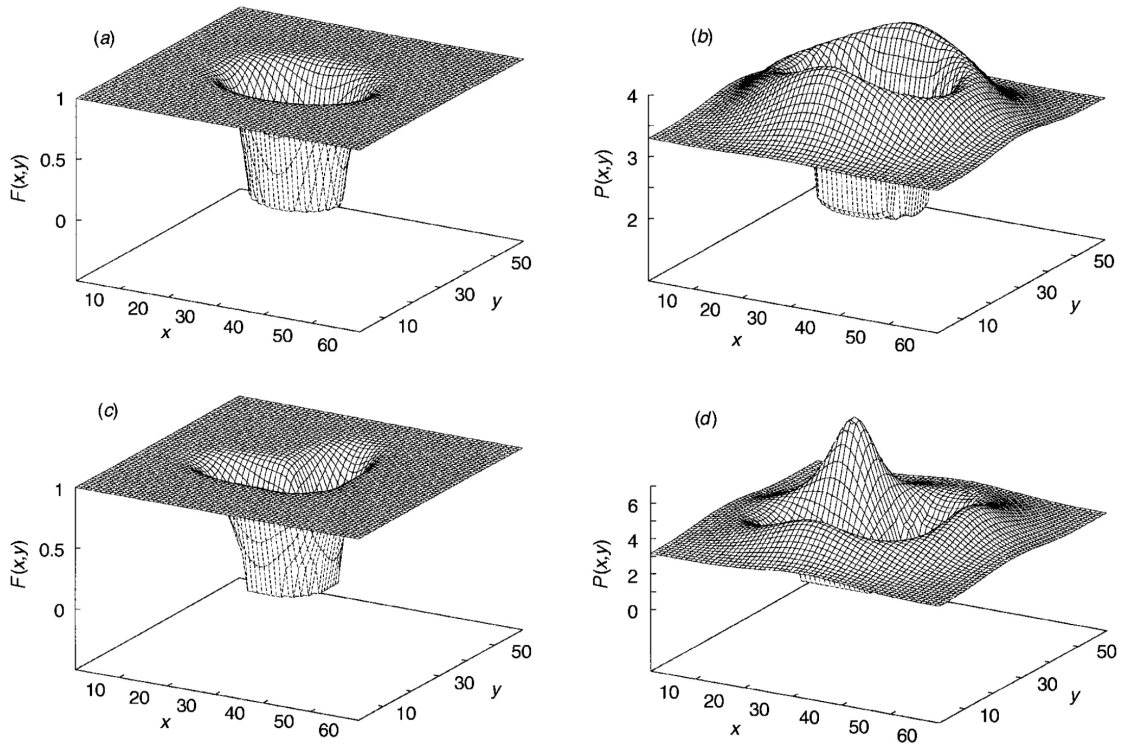


Figure 8. E. Bonabeau (1998). (a) Simulated pheromonal template created by the queen T (b) Spatial distribution of a 2D system with added chemotactic motion toward the pheromonal template represented in a chamber forms around the simulated template. (c) Simulated pheromonal template of a displaced queen. (d) Same as b with chemotactic motion toward the pheromonal template represented in c.

In 2005 Ladley took a computational approach and built a simulator for a voxelized simulation of termite building behavior. In his simulation the terrain, pheromones and termites were all represented as cubes in a cubic lattice. Ladley simulated three different pheromones following Bruinsma's findings as described before.

1. The queen pheromone that is continuously being secreted by the queen at a constant amount.

2. A trail following pheromone, emitted by a trail-following termite at each time step.

3. A cement pheromone that is emitted by a newly cube of building material. This does not emit continuously, but rather, the cube contains a finite amount of pheromone of which a proportion is emitted with each time step.

This simulation also included the effects of diffusion and wind on those pheromones which also prevents pheromones from diffusing through solid blocks of material. The termites were also modeled as cubes, and on each time step they can move to an adjacent cube. But only if that cube is not occupied by building material, and neighbors another cube that is, which prevents termites from moving through the walls and prevents them from 'flying' respectively. When a termite leaves the lattice they are removed and replaced within the lattice. Lastly, termites are not prevented from entering a location occupied by another termite. There are 3 different types of termites in the simulations:

1. Builder termites move with a random chance to move into any legal direction, but the chance is proportional to the strength of the cement pheromone gradient strength. Meaning, termites tend to move towards the peak of pheromone gradients, and stronger gradients having more influence than weak ones. In his simulation the termites move 5 times per time step. Placement of material is restricted to meet certain conditions to require support structures. Builder termites can deposit a block when the pheromone concentration of any pheromone at its location is within a range of 0.1 to 0.5. Builder termites spawn with a block to deposit, and are replaced with a new builder when that block is placed.

2. Trail-following termites lay and follow trail pheromone and are not involved in building activity. They are attracted to trail pheromone similar to how builder termites are attracted to cement pheromone. Trail-following termites enter and leave the lattice at fixed entry points.

3. Nursing termites which move back and forth to and from the queen. They also secrete and are attracted to trail pheromone.

Ladley's simulations contained a fixed number of agents that were allocated one of these roles and did not dynamically change role. A queen termite was also present in some simulations, but was not modeled like one of the termites. It was represented by several blocks in the center of the simulation which continuously secretes the queen pheromone. Ladley first recreated royal chamber construction as seen in Figure 9. And also did this under simulated windy conditions as seen in figure 10 to recreates Bruinsma's observations. In these simulations a termite population of 300 consisting of just builder termites was introduced with a queen in the center of the world. Ladley also simulated the construction of covered galleries over trails (Figure 11) and the effect of trails on royal chamber construction (Figure 12). With these simulations Ladley demonstrated that the logistic constraints, that Bonabeau's model did not include, don't prevent of the formation of termite-like structures in his simulation. And that those constrains are important or even necessary to achieve efficient construction. In ladleys simulation of covered galleries the trail is a result of the activity of the trail termites. However, trail forming is heavily guided by spawning termites with at ground level on the center of one of the edges of the world after they reach the edge of the world. Whether or not the rotation was also predefined is not clearly stated in the paper. Furthermore, the simulated termites can sense pheromones all around them instead of just the antennae, which could have affected the behaviour of the termites.

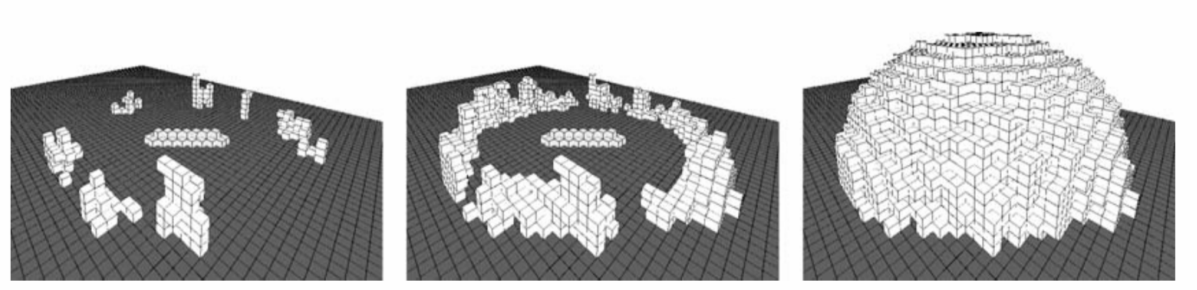


Figure 9. D. Ladley (2005). Construction of royal chamber around a queen. Showing a spherical chamber.

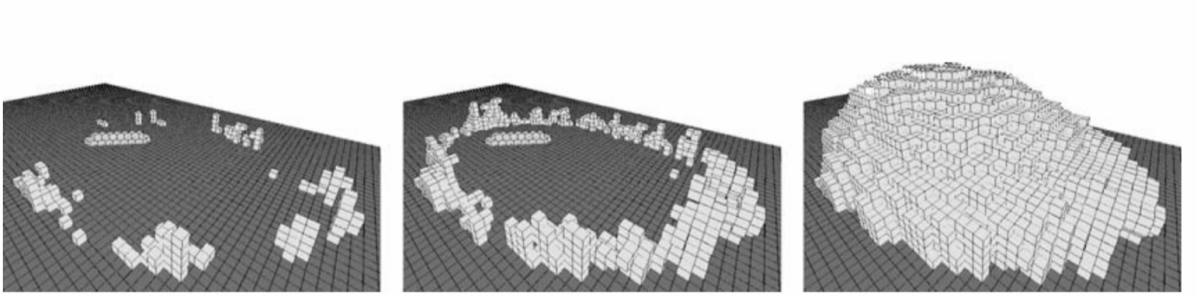


Figure 10. D. Ladley (2005). Construction of royal chamber around a queen under windy conditions. Showing an elongated chamber.

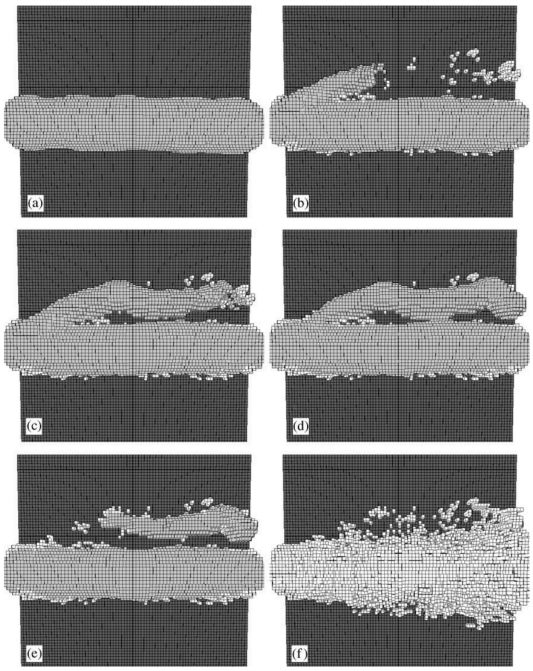


Figure 11. D. Ladley (2005). Construction of a covered walkway. (Floor in heavy tone; trail pheromone in medium tone; building pheromone in light tone.)

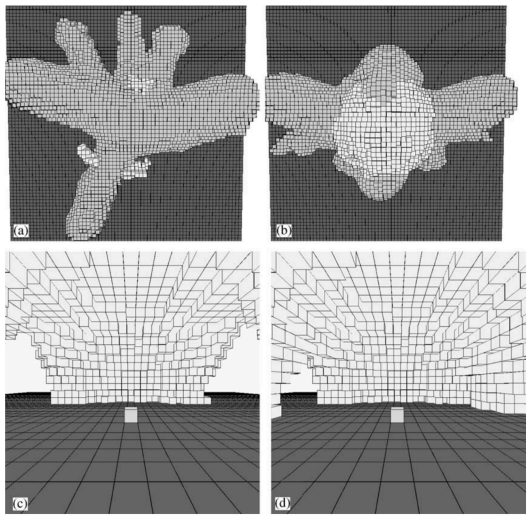


Figure 12. D. Ladley (2005). Royal chamber construction in the presence of pheromone trails. Showing how several pheromone trails (medium tone) between the central queen and the lattice periphery cause the formation of entrances in the royal chamber made out of material (light tone). Number of trails reduces over the course of the simulation.

In 2008 Feltell used a similar simulation as Ladley but instead of limiting the deposition and grasping of blocks to within a range of pheromone concentration, he instead used the pheromone concentration to influence the chance of grasping and depositing. One more limitation that was introduced is that termites have to pick up a block before they can place it down. He then simulates the construction of the royal chamber under various conditions, the construction of covered galleries, and the combination of the two. This was done by implementing the response-threshold function, proposed by Bonabeau as described above. This lead to the following equations for the drop chance (*P(drop)*) and pickup chance (*P(pickup)*).

Q = concentration of queen pheromone between 0 and 1.

T = concentration of trail pheromone between 0 and 1.

C = concentration of cement pheromone between 0 and 1.

$$P(drop)_Q = \begin{cases} 1 - T_\theta(Q+T), & \text{if } Q+T > 0 \\ 0.1, & \text{otherwise} \end{cases}$$

$$P(drop)_C = \begin{cases} 1 - T_\theta(C), & \text{if } C > 0 \\ 0.1, & \text{otherwise} \end{cases}$$

$$P(drop)_F = \begin{cases} 1, & \text{if tactle stimulus is present} \\ 0.1, & \text{otherwise} \end{cases}$$

$$P(drop) = P(drop)_Q + P(drop)_C + P(drop)_F$$

$$P(pickup) = \begin{cases} 1 - T_\theta(Q+T), & \text{if } Q+T > 0 \\ 0.1, & \text{otherwise} \end{cases}$$

Through his simulation (Figures 13 to 18) he shows that a simple response-threshold function with stigmergic mechanisms can explain many of the poorly understood behavioral patterns. And that randomness can partially account for some of the self-organization within the building process. However, Feltell did mentions the limitations of this simulation, stating the following:

“This agent-based model allows for rapid experimentation and close study of individual behaviour and interactions, but the approach suffers from many inherent drawbacks. The discrete, simplified environment and tight spatio-temporal coupling between continuous pheromone diffusion and the approximated physical material means that representative scales are inherently inaccurate. As a result many parameters were arbitrarily chosen, which detracts somewhat from the realism and applicability.” (Felltel, 2008)

Interestingly enough, neither Ladley nor Feltell noted the potential influence of limiting agent movement to the 27 cardinal directions. Only Feltell stated that the use of this 3D grid lead to the inability to apply Bruinsma’s exact measurements. Instead of using this kind of discrete, voxelized terrain, it would be interesting to recreate their models in a continuous space simulator and compare the results.

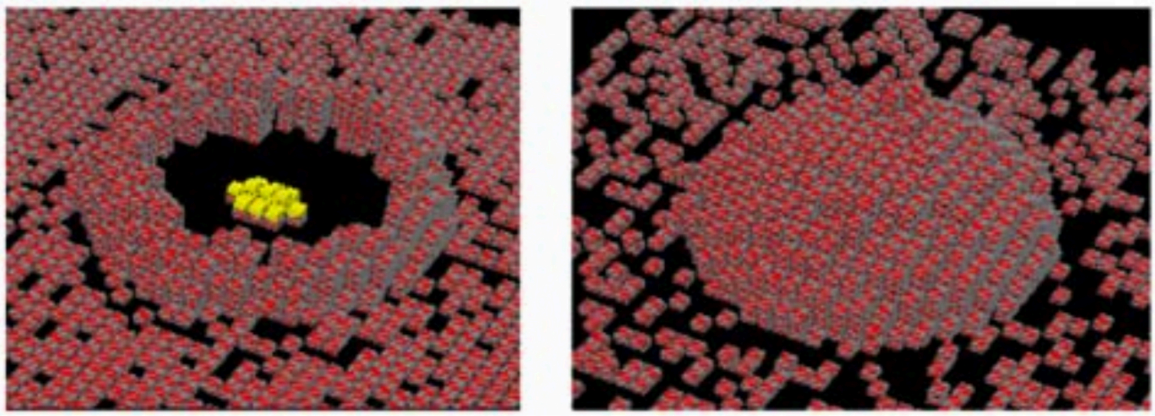


Figure 13. D. Feltell (2008). (Left) Royal chamber under construction, showing queen object in centre; (right) completed royal chamber.

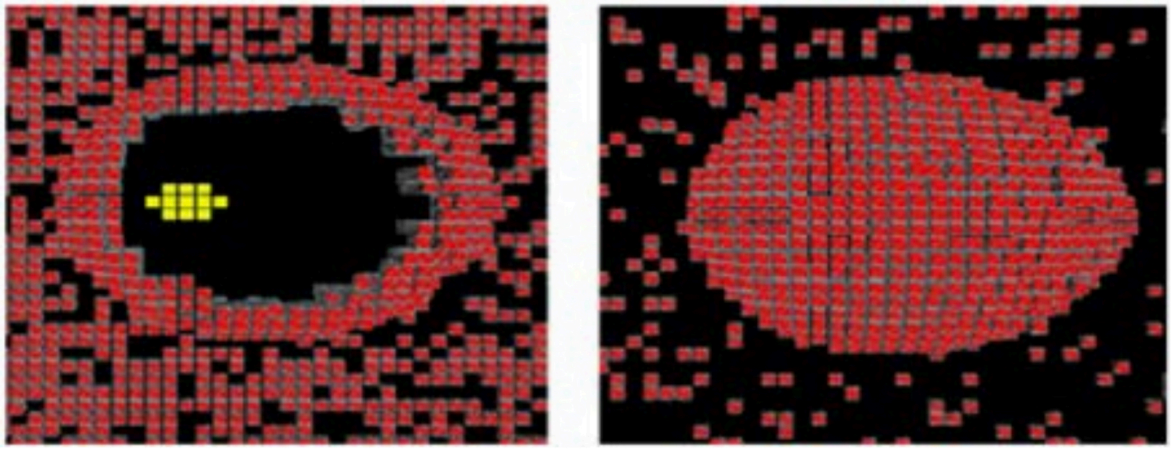


Figure 14. D. Feltell (2008). (Left) chamber under construction, under the influence of convection; (right) completed chamber.

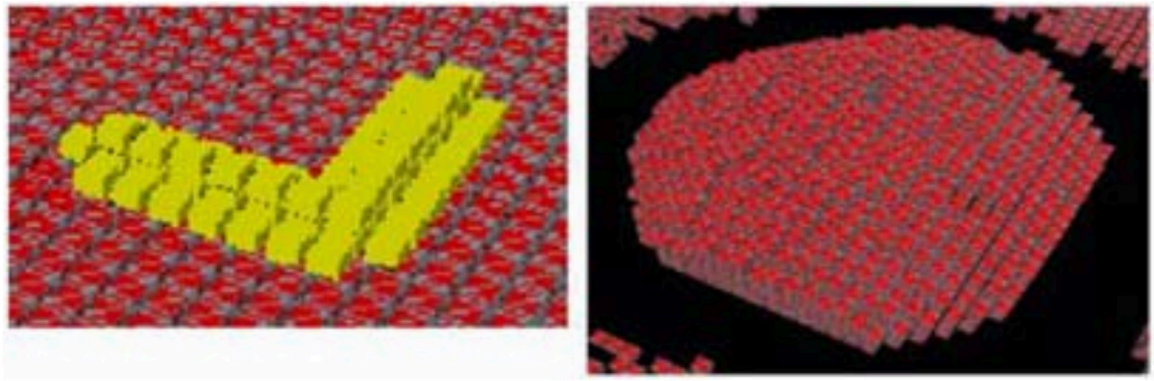


Figure 15. D. Feltell (2008). (Left) Altered shape of queen object; (right) royal chamber created with altered queen configuration

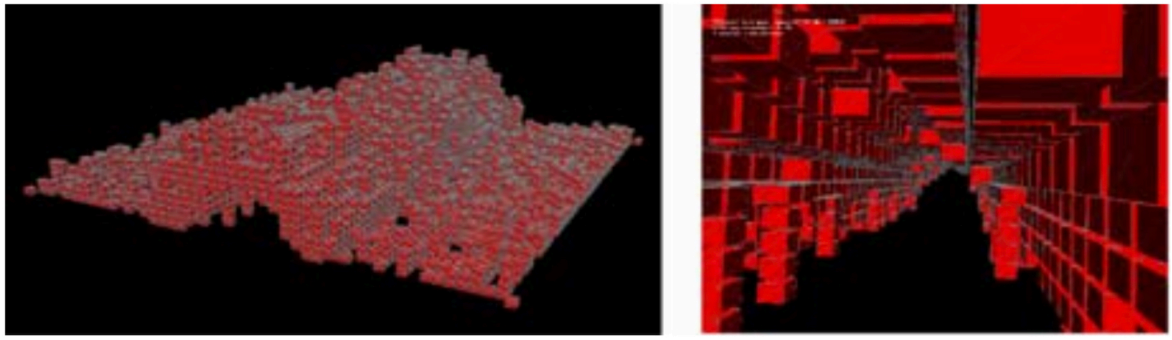


Figure 16. D. Feltell (2008). (Left) completed trail gallery; (right) view from inside gallery

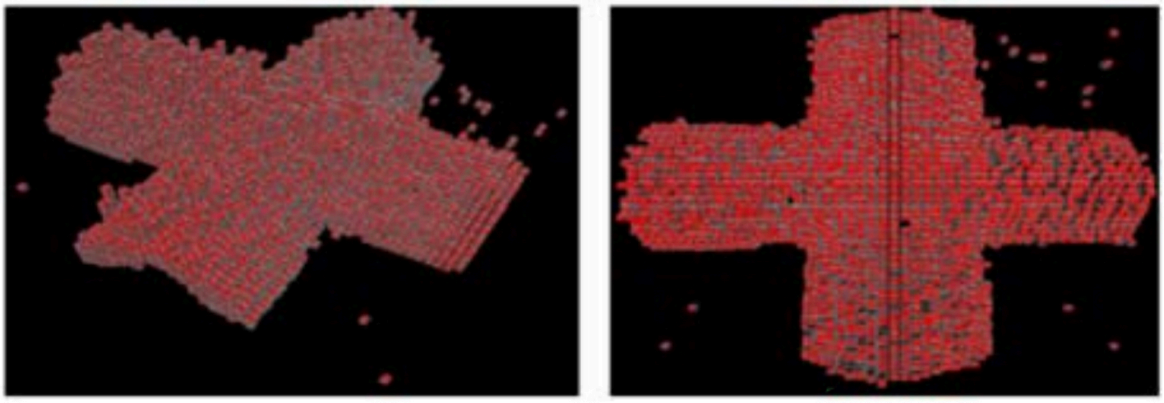


Figure 17. D. Feltell (2008). Construction of covered galleries over a crossroad of trails.

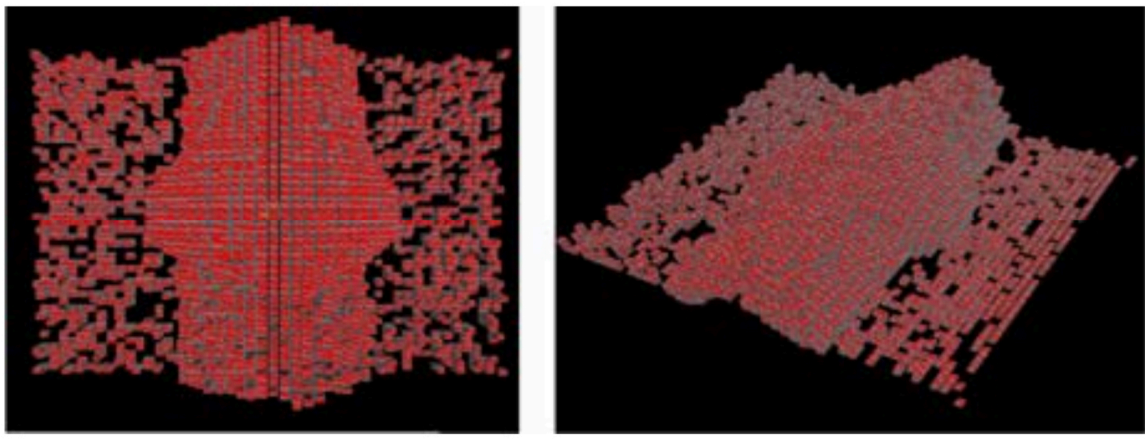


Figure 18. D. Feltell (2008). Completed structure formed from combined royal chamber and trail gallery

1.4.6. Conclusion

In conclusion, mound building termites are truly remarkable creatures that have captivated the attention of scientists and researchers for many years. The construction of their mounds is a testament to the remarkable organization and coordination that is possible in nature. Studying these insects has provided valuable insight into the mechanisms behind swarm intelligence and collective decision-making and has helped us better understand how to apply the concept of swarm intelligence in swarm robotics. However, more research is needed as there are still many unknowns of the underlying mechanisms of their behaviour. Previous models and simulations led to new insights in termite building behaviour. But there is potential for a lot more to be discovered through the experimentation with simulations.

1.5. Requirements

Termites are a great example to how decentralized control systems can be used in robot swarms for construction. Previous simulations have excellently demonstrated the value of a simulator by using them to verify existing theories on how termites work. But many questions remain still, as simulations have yet to fully recreate a termite mound. Therefore, more research is needed to experiment with and evaluate different control strategies and tune the parameters of the simulation in an attempt to learn the inner workings of a termite mound construction. This can be done with a simulator, but for further research in termite-behavior, the simulator will need to at least be able to recreate previous simulations in order to build upon them. Consequently, the simulator requires at least the same functionality as the previous simulators. Although, much research can be done with identical simulators, there is an opportunity to address the limitations of those simulators and overcome them. Looking at previous simulations, they have the 4 following functionality:

1. **A 3-Dimensional world**: if the goal is to approximate building behaviour of termites, which build in 3-dimensions, then the simulator should also be 3-dimensional to allow for similar structures to be built.

    1. The cause for many of the limitations of the simulators described in Section 1.4.5 is that material, pheromones and termites are all blocks positioned in a grid. This limits the resolution at which agents can move, turn, deposit material, and pick up material. Thus, limiting the behaviour that can be simulated. Therefore, this simulator should be continuous space, rather than grid based, as is described in more depth in Section 2.2.1.

2. **A Swarm**: The intent of this tool is to experiment with the emergent behaviour of swarm intelligence, for which multiple agents are required to form a swarm. If the simulator supports simulating multiple agents, then its a matter of performance to how many agents can be simulated. Previous simulators were able to creates swarms with a population of at least 300 agents.

    1. Agents need to have a position and orientation.
    2. Agents need to be able to interact with the terrain, pheromones, and other agents.
    3. Agents need to be visible.

3. **Pheromones**: pheromones are one of the key mechanisms with which termites communicate and plays a major role in how the termite mound comes to be shaped as described in Section 1.4.2. If our goal is to draw inspiration from them and use them as a reference to evaluate the tool, then pheromones are required feature. Diffusion is how the pheromone propagates through space. This is the cause of pheromone gradients which add a directional component to the sensing of pheromones.

    1. Pheromones need to be able to be placed in the simulation space.
    2. Pheromones need to be able to be sampled in the simulation space.
    3. The pheromones should be able to diffuse and decay.
    4. Pheromones need to be visible

4. **Terrain**: a detailed terrain is required to build the ramps, slopes and other such organic geometries required to navigate the structure itself during construction. But the simulator must also be able to generate and edit this terrain.

1. The terrain needs to support the agents. The agents need to be able to ‘stand’ on the terrain.
2. The terrain needs to be visible.
3. The terrain needs to be dynamic. In other words, be able to change shape during the simulation.

While it is not shown in the paper written about the other simulations, another required functionality is:

5. **Interface**: for adjusting the parameters that control the behaviour of the swarm. This improves the usability of the simulator and consequently, the speed at which users are able to iterate on parameter settings.

Besides the requirements, there are also some factors which should be optimized where possible. These factors can be used as metrics for evaluation if there are multiple options where all the requirements are met. Often times, these metrics conflict with each other, and a balance between those metrics will need to be chosen.

1. **Configurability**: the extend to which the simulations parameters can be changed. Allowing for more configurability will give more freedom to the user to change the behavior of the swarm.
2. **Ease of use**: how easy or difficult it is to run a simulation and adjust the parameters, reducing time spend configuring the parameters.
3. **Performance**: the simulation should be optimized to perform simulations as fast as possible. This reduces the time each simulation takes, thus increasing the speed at which the user can iterate.
4. **Realism**: the simulation should strive to be as realistic as possible to decrease the discrepancy between the simulation and termite swarms. A large discrepancy can lead to a simulation exhibiting a completely different behavior than a robot swarm with the same control strategy and parameters.

1.6. Possible approaches

If the goal is to create a simulator that does not have the same limitations as the custom simulators described in Section 1.4.5, the first step is to analyze if there are simulators available that meet the requirements. There are a lot of established, open-source, and general purpose robotics simulators which potentially could be used to simulate the termite-like builders, which will be called agents from this point on.

Each simulator is evaluated on whether they support the requirements, or if it is possible to modify the simulator to add the required functionality. All the simulators have a method for adjusting the simulation parameters, so that requirement will be left out of the comparison. A color coded table is added to each simulator section that indicates whether each function is supported (green), could potentially be modified to support (yellow), or does not support the functionality (red). All simulators support some kind of interface, so that will be left out of the comparison.

1.6.1. Available simulators

ARGoS

ARGoS is *"a multi-robot simulator. It can simulate large-scale swarms of robots of any kind efficiently. You can customize ARGoS with plugins."* It is designed around the idea of modularity with an ability to swap out modules for custom models to add or change functionality where needed. Most of the development documentation is missing. (Even though the project started in 2016 and the last update was 2022)

argos-sim.info

Multiple agents	Pheromones	Dynamic terrain	3D

CoppeliaSim

Formerly known as V-REP, Coppelia is mostly focussing on high detail simulations of a small number of robots.

coppeliarobotics.com

Multiple agents	Pheromones	Dynamic terrain	3D

Gazebo

Gazebo is a platform allowing users to implement its libraries to build their own custom simulation.

gazebosim.org

Multiple agents	Pheromones	Dynamic terrain	3D

BurlapCraft

https://h2r.cs.brown.edu/wp-content/uploads/2015/09/aluru15.pdf

A mod for Minecraft which is created to enable the use of reinforcement learning and the planning library BURLAP (MacGlashan 2015) for simple and rapid prototyping and experimentation of task performing models.

Multiple agents	Pheromones	Dynamic terrain	3D

NetLogo

NetLogo is a 2D web based agent-based modeling environment aimed at education. It is a very easy to use program that allows users to program the desired agent behaviour in one file.

https://ccl.northwestern.edu/netlogo/bind/

Multiple agents	Pheromones	Dynamic terrain	3D

Player/Stage

Stage is a simulation backend designed for simulating a population of mobile robots in a 2 dimensional bitmapped environment. It's preset comes with a standard Player interface which provides control to robots over a network connection.

playerproject.github.io

Multiple agents	Pheromones	Dynamic terrain	3D

TeamBots

TeamBots is a java based collection of programs and packages for multi-agent robotics research.

https://www.cs.cmu.edu/~trb/TeamBots/

Multiple agents	Pheromones	Dynamic terrain	3D

MORSE

MORSE is a simulator by openrobots that is able to run up to a few dozen robots.

https://www.openrobots.org/morse/doc/stable/morse.html

Multiple agents	Pheromones	Dynamic terrain	3D

IsaacSim

Developted by Nvidea, IsaacSim is a powerful simulator focussed on generating data to train AI for robotic applications.

https://developer.nvidia.com/isaac-sim

Multiple agents	Pheromones	Dynamic terrain	3D

1.6.2. Simulators vs requirements

The evaluation of the requirements can be seen in Table 3.

Name	Multiple agents	Pheromones	Dynamic terrain	3 Dimensional
ARGoS				
V-REP				
Gazebo				
BurlapCraft				
NetLogo				
Stage				
TeamBots				
MORSE				
IsaacSim				

Table 3. Simulator evaluations. (Green is supported, Yellow is potential to be added, Red is not supported without potential to be added). There is no simulator that meets all the requirements.

Unfortunately, there is no simulator currently available that provides all the required features. But there are some that could potentially be modified to support all the required features such as ARGoS, Gazebo, MORSE, and IsaacSim.

These simulators offer advanced capabilities for complex robots. However, this is overkill for the functionality required to simulate simple termite-like agents. And for the functionality that still has to be added or modified, it will have to fit into the framework of what is already there. All of the critical things that those simulators offer can be recreated in unity, while Unity gives more freedom.

1.6.3. Creating a simulator

Automation in the form of swarm robots is an increasingly feasible opportunity for solving complex physical problems. However, how such a swarm is to be instructed and controlled is one of the many topics that requires further research. The problem is that, currently, there are no robot swarm simulators that are suitable for simulating termite like building. That is why a new simulator specifically for termite like builder robot swarms is required for further research in this field. This project aims to create an open-source simulator of a termite-like builder robot swarm for experimenting with different control strategies and adjusting them accordingly. In other words: a tool for researchers to design desired behaviour of a builder robot swarm.

This will include the development of the virtual agents, pheromones, the terrain, and their interaction with an appropriate level of realism. The creation of a physics engine is outside of the scope and instead one that is readily available will be used. Several control strategies from literature will be implemented and their performance evaluated and compared to each other. Furthermore, a user interface to adjust the control strategy and environmental settings and review relevant performance data of the swarm will be created and tested.

2. Developing the simulator

As described in the previous chapter, a simulator for simulating termite-like construction can further the field of swarm robotics if the simulator has the required functionality. In this chapter these functionality will be developed into a new simulator: TermiteSim. TermiteSim is the simulator build for this project and is built with the Unity game engine (Unity, 2023) available on unity.com. Although a game and a simulator are two different interactive experiences, they both utilize many of the same functionality which unity provides. Such as the rendering of objects to the screen, managing memory, and handling physics interactions with the Nvidia PhysX engine for simulating collisions. Unity also provides templates for 2D, 3D, Virtual Reality and Augmented Reality games and cross-platform build exporting which further saves on time. Lastly, Unity provides excellent documentation and many content creators offer detailed guides on how to achieve certain functionality.

The development of the simulator is divided into 5 parts:

1. The foundation of the simulator: setting it up in Unity as a 3D environment.
2. The terrain which the agents will interact with.
3. The swarm of termite-like builder agents.
4. The pheromones which are a key mechanism in the stigmergic communication of termites.
5. The interface that is used to set up the simulations and adjust their parameters.

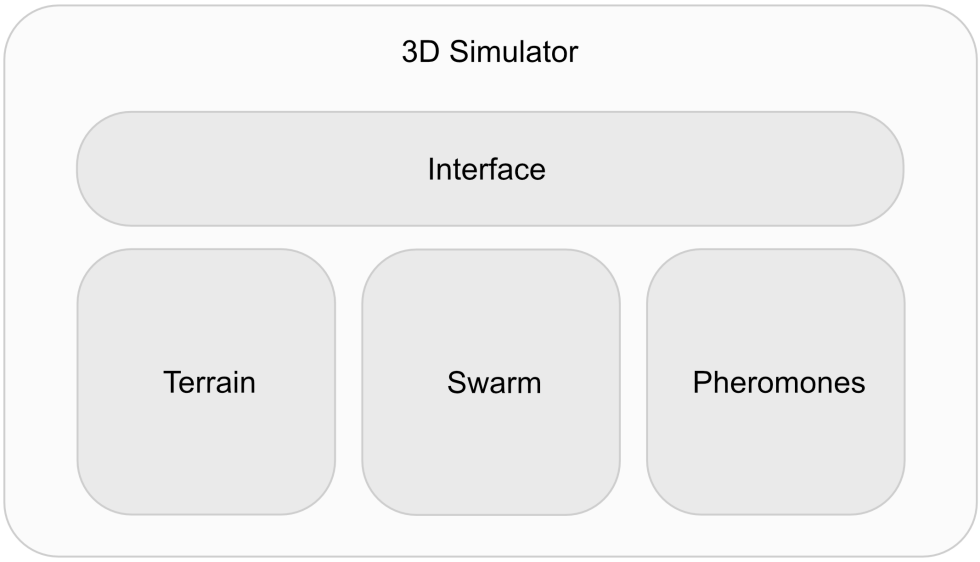


Figure 19. Simulator overview of the 5 key elements.

Each of the sections below describes one of these parts which is to meet one of the key requirements. These sections will briefly go into the considered options with their pros and cons, and go into more detail about the implementation that was ultimately chosen.

2.1. Design requirements

As described above; the 5 key requirements for a simulator are that the simulation is 3D, it is able to simulate multiple agents, that the terrain is dynamic, that there is some sort of pheromone support and an interface to adjust the parameters of the simulation.

These requirements are high level, and there are many layers to what each requirement entails. To prioritize features and define the scope of the project, a MoSCoW chart was created. This stands for Must have, Should have, Could have, and Wont have. Features that were discussed at the start of the project were categorized based on their necessity and importance. Seen in Figure 20 is the MoSCoW chart, colour coded after completion of the simulator to reflect what was and was not achieved.

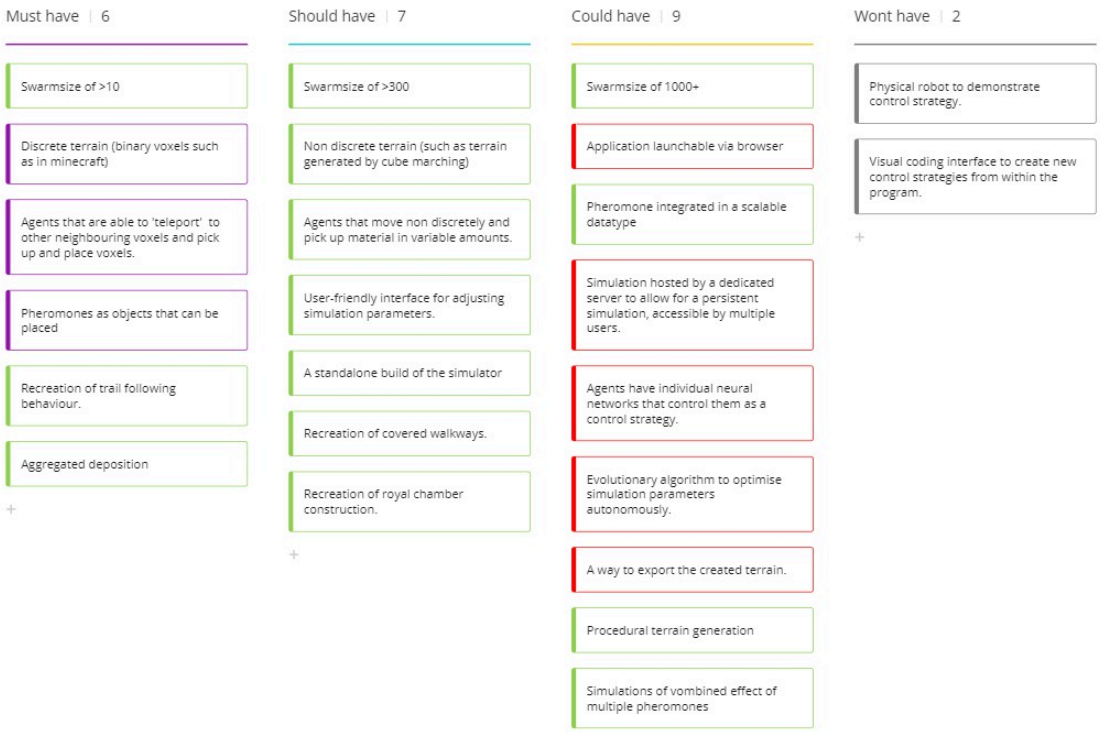


Figure 20. Color coded MoSCoW chart of project objectives. Showing what was not included because a better version was achieved (Purple), what was achieved (Green), what was out of scope (Grey), and what was not achieved (Red)

2.2. A 3-dimensional world

One of the key requirements as described above is that the simulation has to be 3 dimensional. Luckily, Unity provides a template which includes everything that is needed to make a 3D game. In large this already meets the requirement. But, there is one choice that dictates the direction of the project going forward: whether to divide that 3D space in discrete steps or continuous space. This is a choice that is very much a weigh off between to performance, simplicity and realism.

2.2.1. Options

Cell based

The simplest option is to divide the space into discrete chunks or cells and then group everything within that space as having the same position, such as in many tabletop board games who do this on a 2D board. One example of this is chess, there each piece can occupy one of the tiles and can move to other times. Chess divides the space in to squares, which would be similar to a cubic lattice structure in 3D. But there are other options for cell shapes to form a lattice, such as the octahedron, triangular prism, hexagonal prism and the rhombic dodecahedron. In principle any cell shape that is fits the criteria of a plesiohedron. However, cubic is the most straight forward and will be used as an example from here on out.

Pros:

- This lattice structure could then be used to contain all of the key elements within the simulator: the agents, the terrain, and the pheromones which can al be represented as cells. This would greatly reduce the complexity of managing these different functionalities and drastically reduce development time.
- Another benefit is that this indexing can be used for managing interaction. Meaning that instead of using a physics engine to check for collisions, only the neighboring cells can be checked. This greatly simplifies many aspect of 3D space navigation and interaction, and promises better performance and reduced system complexity in comparison to continuous space simulations.
- The balance between detail and performance can be changed by a single variable; cell density. Increasing the density of cells increases the amount of detail to better facilitate an approximation of the organic shapes of termite-like mound building. Much like increasing the pixel density (resolution) of an image defines the balance between detail and file size. However, this comes at the cost of performance. Where there will be more steps needed for the agent to travel the same distance.

Cons:

- A major caveat is the limitation of directions the agent can move in. The exact limitations are of course dependent on the lattice structure, but for a cubic lattice structure agents would be limited to moving along the axis or diagonally. The inability to support gradual turning and the resolution of movement being limited to the resolution of the grid strongly limits the range of real behaviors that can be replicated.
- There are other models that have used similar approaches with success to model termite-like building behaviour. Feltell and Ladley have created models that use a cubic lattice structure as such a cell based system as seen in Figures 9 to 18. However, they do discuss in their paper the limitation of this approach, being inaccuracies in agent to mound proportions, and arbitrarily chosen parameters.

Continuous space

Instead of dividing the space into tiles, a continuous non-discrete space is the more standard option. continuous space positioning is free from any restrictions which offers much more realism, but also the complexity that comes with it.

Pros:

- Complete freedom of navigation. Agents can move in any direction by any amount, lifting restrictions that previous models had. To be more concrete: gradual turning that happens over several simulation steps, with the rate at which agents turn being variable and configurable. And the movement speed of the agents being configurable.

Cons:

- The largest problem with this approach is the significant increase in complexity. Instead of using a Cell grid that encompasses all key elements, the agents, terrain, and pheromones will al need their own implementation and have their own methods of managing their interactions. This will greatly increase the development time required to come to an minimal viable product of the simulator.

2.2.2. Implementation

As described in Section 1.6 others have already created cell based simulators. Therefore, the choice was made to go with the more complex continuous space approach. This decision will lay the foundation for the rest of the project. For each of the key elements the implementation can now be chosen independently. As this is quite an abstract decision, much of the consequences are specific to the chosen implementation in the following sections, and thus can not be described in isolation.

2.3. The Terrain

The terrain is the first part that needs to be solved. As stated in the key requirements, the terrain will need to be dynamic so it can be changed (by the agents) while the simulation is running. The choice between different types of terrain implementation can greatly limit the shape of the structures that the agents will be able to build. There are also two unique factors that should be considered for this choice. Firstly, the limitations in what kind of geometry can be achieved, with the aim of closely resembling the organic shapes of termite mounds. Second, the realism of how the excavation and depositing process is represented.

2.3.1. Options

Cubic cells

Though a cell based structure was not chosen as a foundation for the project, individually, the terrain could still be implemented this way for good reason. An example of this type of terrain in a video game is Minecraft where only the terrain is fixed to a cubic grid, as seen in Figure 21.

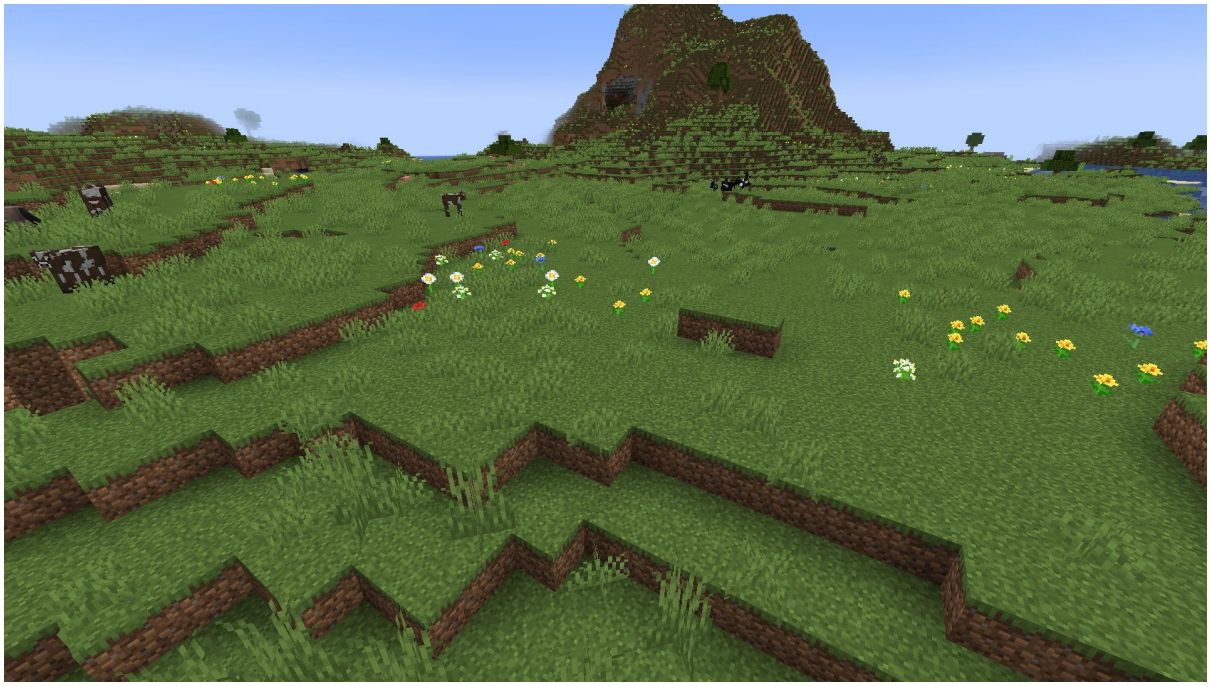


Figure 21. Hurricane Beryl. The terrain from Minecraft illustrating the possible approach of cubic cells. https://minecraft.fandom.com/wiki/Plains

Additionally to determining whether a is solid or not, it could have an internal value of how much material is present, essentially just making it a 3D scalar field. For example, each block, solid or not, has a hidden value that is between 0 and 1. When material is added to the block it increases this value, but it will only appear as a solid block once the value is above 0.5. This makes that not every deposition equals to one cube and that an agent can also spread its material deposition over several cubes. So this method does still allow for a certain granularity in the interaction between agent and terrain.

Pros:

- Reducing the terrain to individual cubes offers an easy implementation and segmentation of terrain. Much like termites handle soil pellets, each individual cube could be seen as a soil pellet. This simplifies interaction with the terrain and other elements of the simulation.

Cons:

- Having the terrain build out of cubes means that all the angles in the terrain will be orthogonal. Of course, termite mounds have organic shapes and no orthogonal edges. To approximate such shapes, the density of cubes would have to be reasonably high. Even then all the faces of the surface will be orthogonal, so agents will have to average multiple faces to calculate the curvature. This further reduces the computational advantage of this approach.
- There is no option to partially fill a cube in terms of its collision. It is either permeable or solid, no in between. This is highly unrealistic and will likely lead to unexpected emergent behavior.

Height map

A height map is a single 2D surface which is subdivided (Figure 22). Each point on the surface has a single value that represents the height of that point. This technique is often used in games to generate large scale world and is much more scalable than other methods because it reduces the problem to a 2D datatype.

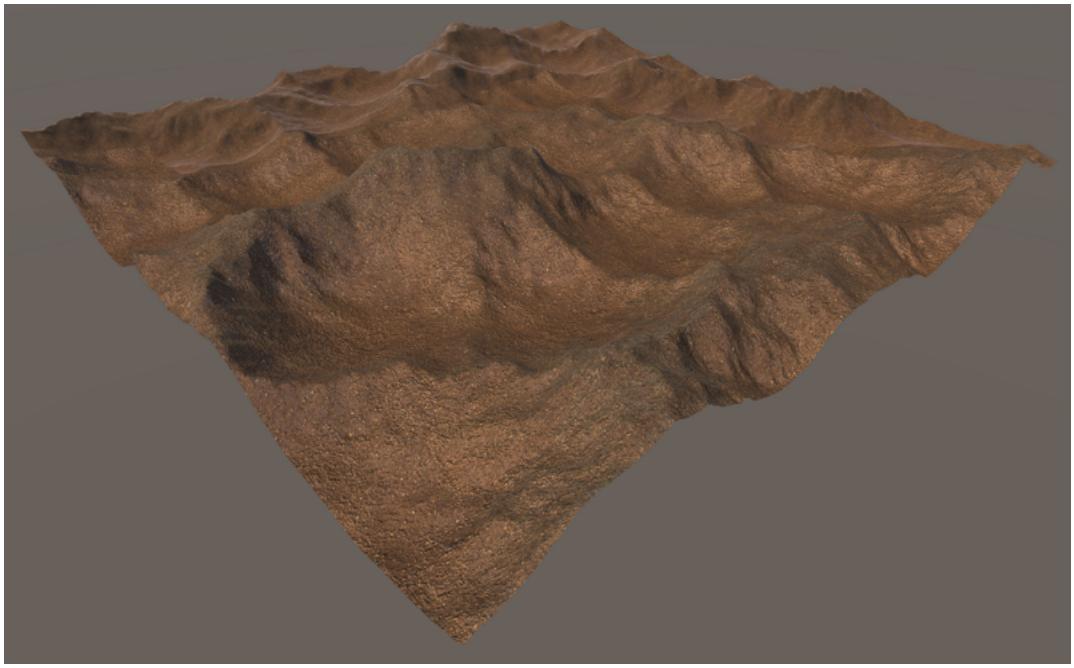


Figure 22. SideXF. Height map terrain, showing a high amount of detail, but no possibility of overhang. https://www.sidefx.com/docs/unity/_terrain.html

Pros:

- Unrivaled in term of resolution to performance ratio.

Cons:

- Does not allow overhang. Which makes it impossible for agents to dig tunnels or build roofs. this is extremely limiting for the functionality of the simulator.

Marching cubes

Marching cubes is an algorithm often used in computer graphics to visualize medical images. It takes an 3D scalar field as input and produces a surfaces mesh as seen in Figure 23. For a more detailed explanation, see Section 2.3.2.

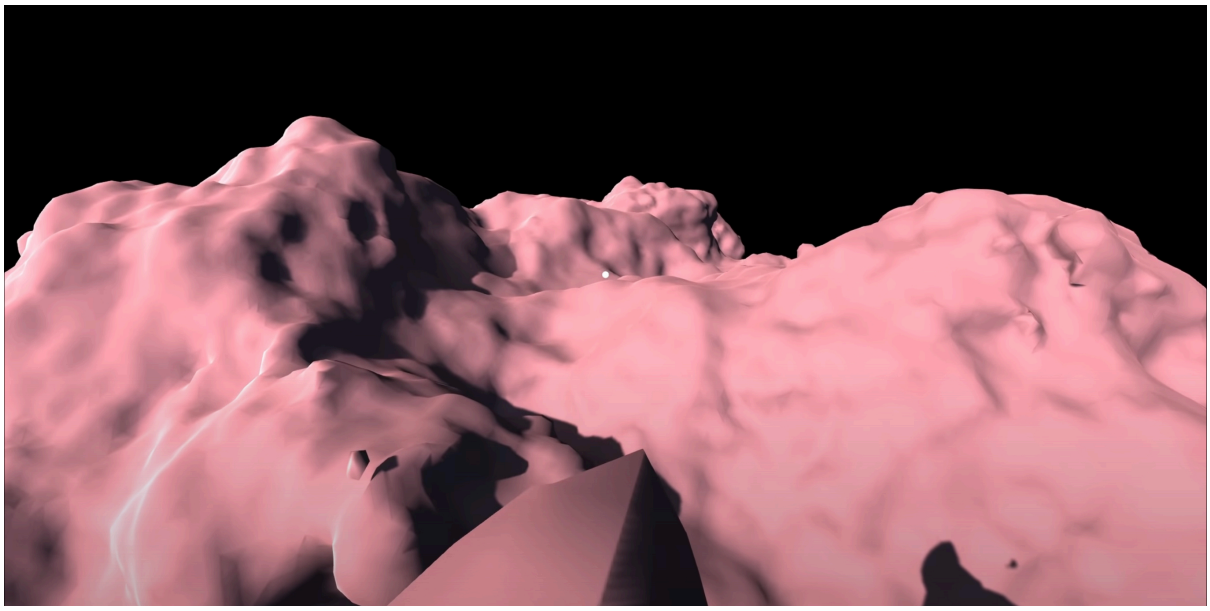


Figure 23. Sebastian Lague (2021). Marching cubes terrain giving both a high degree of detail and allowing for any kind of shape. https://www.youtube.com/watch?v=vTMEdHcKgM4

Pros:

- The underlying data structure which produces the mesh through the algorithm is the same as the cubic cell approach makes dynamically generating and editing it just as easy.
- With the extra translation step of the marching cubes algorithm, the terrain is not limited to orthogonal edges and can much more closely approximate organic shapes at lower resolutions.
- It allows these grid points to be partially filled, which is required for more organically shaped terrain.

Cons:

- The algorithm itself is quite complex and will be difficult to recreate.
- For any physics interaction, a collision mesh is needed. This collision mesh will need to be recalculated every time the mesh is changed. This is a potentially significant hit to the performance of the simulation.

Unstructured mesh

These previous examples generate a mesh based on an underlying 2D or 3D grid. Although this simplifies many interactions, it is not required per se. There is a possibility of having the terrain be represented as an unstructured mesh, where vertexes can be added or removed where needed to manage detail.

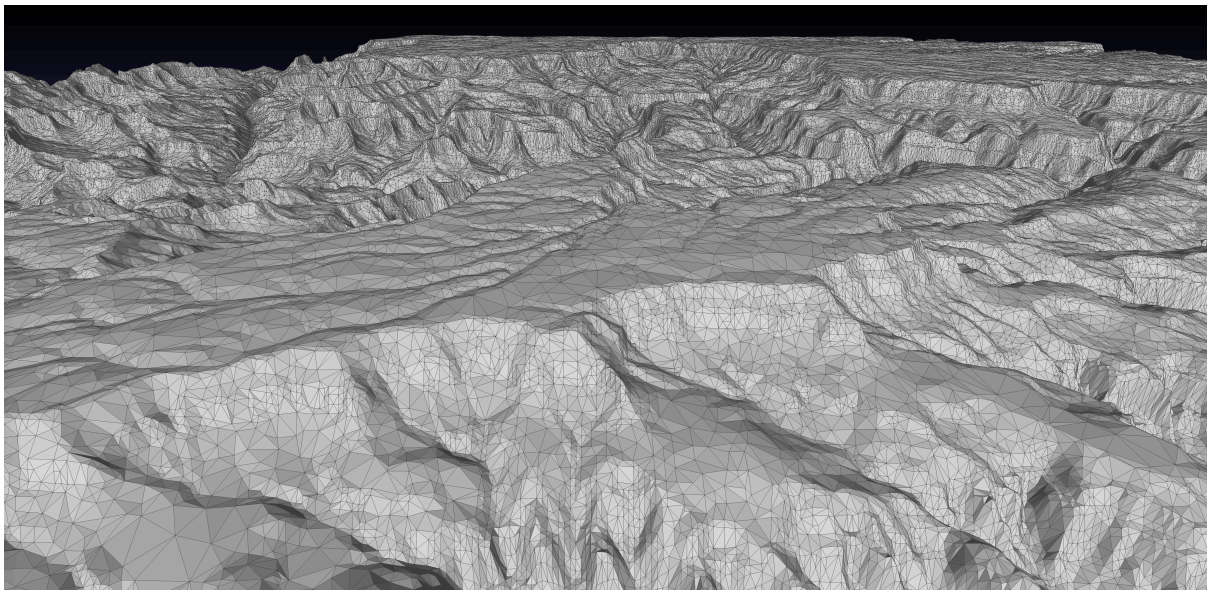


Figure 24. Michael Fogleman (2019). Unstructured Mesh demonstrating limitless detail. https://www.michaelfogleman.com/projects/hmm/

Pros:

- Amount of detail not limited by resolution. As more 'resolution' can be added locally in areas where it is needed in the form or extra vertices.
- Free from any restrictions, theoretically allowing for any shape imaginable.

Cons:

- Because of its unstructured nature, working with such a mesh is extremely complex. Especially when dynamically changing it as agents will require to excavate and deposit material.

2.3.2. Implementation

For the terrain, the highest level of detail, which the unstructured mesh approach offers, is preferable. However, the complexity around implementing it as a dynamically editable terrain is too large of an uncertainty. The marching cubes method still has a far greater amount of detail than other simulators and offers a much less complex implementation than the unstructured mesh method. It was therefore chosen as the implementation for the terrain.

Perlin noise

To generate an initial terrain, one can simply generate a flat surface. But an interesting functionality would be to generate randomized terrain geometry. This can be done by using a noise generation function retrieve a randomized value from and use that to determine its shape (Figure 25).

In the case of TermiteSim, this randomness is generated with Perlin noise. Perlin noise is a method for generating multidimensional gradient noise. A co-ordinate can be passed to the Perlin noise function, together with several parameter defining the characteristics of the desired noise. It then returns a scalar value representing the noise in that coordinate. Perlin noise specifically is often used for terrain generation because of its produced noise resembling topological features.

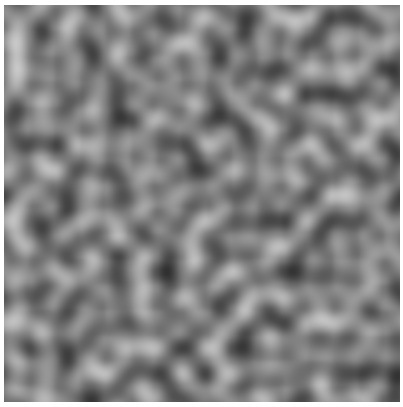


Figure 25. Lord Belbury (2022). Two-dimensional slice through 3D Perlin noise at z=0. https://en.wikipedia.org/wiki/Perlin_noise#/media/File:Perlin_noise_example.png

Cube marching algorithm

As briefly described before, the cube marching algorithm is a technique used in computer graphics to generate 3D shapes by marching through a grid of cubes or voxels. This technique is often used in medical imaging, but also in conjunction with procedural generation of 3D models, like this project.

One of the cons of marching cubes is that is is quite complex and difficult to recreate. Luckily, there was already an open-source project available that implemented the marching cubes algorithm in a terraforming game in Unity. This is the Terraforming project by Sebastian Lague who shared the project under an MIT license. His Terraform project was used as the starting point for TermiteSim. And many of the features of the terrain are from his project. These include the perlin noise sampling, the implementation of the marching cubes algorithm, the ability to dynamically edit the terrain, and the adjustability of the parameters that influences the initially generated terrain characteristics.

The marching cubes algorithm can be divided into 4 steps which are illustrated with examples taken from Sebastian Lague's video.

1. First a 3D scalar field is required, which is generated from the perlin noise function described above. (Figure 26)
2. Then an isovalue is defined, which is a value that defines the boundary between terrain and non-terrain. Referred to in Figure 27 as rock and air respectively.

3. For each line that connects a terrain and a non-terrain point, a vertex is generated on that line. These vertexes are connected to for the trianges that make up the terrain mesh. This surface mesh is also known as a isosurface. (Figure 28)

4. This process is repeated for every cube in the structure, and the triangles are stitched together to form one mesh. (Figure 29)

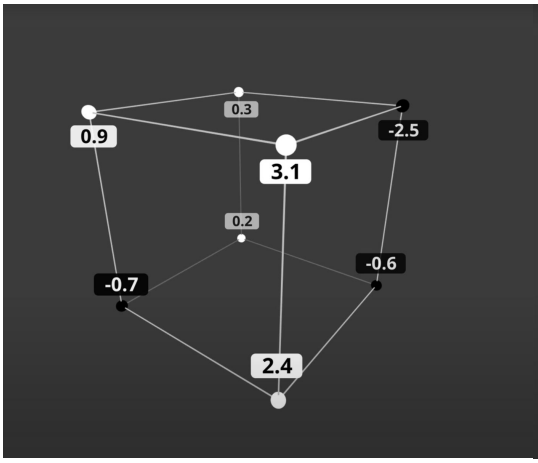


Figure 26. Determining the density value at each grid point

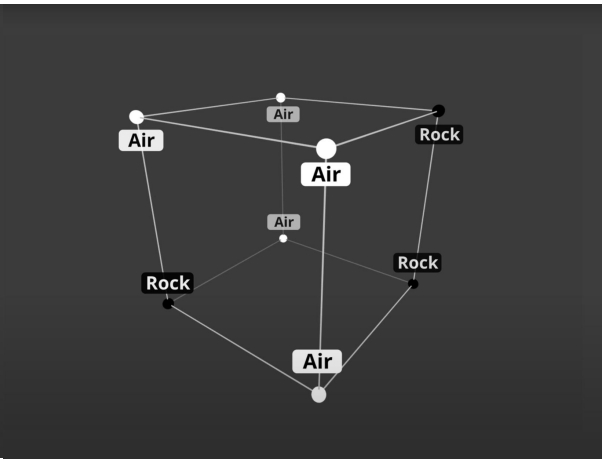


Figure 27. Determining whether each grid point is inside or outside the terrain

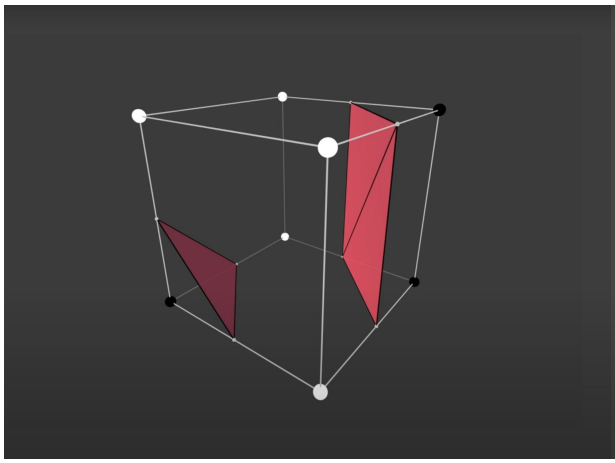


Figure 28. Placing new vertexes between the inside and outside of the terrain and connecting them

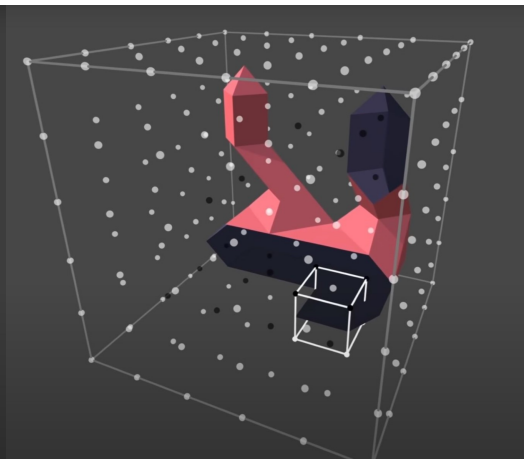


Figure 29. Connecting all the separate sections into one mesh.

Sebastian Lague (2021). Visualisation of cube marching. https://youtu.be/vTMEdHcKgM4?t=79

Code Structure

All the functionality of the terrain in TermiteSim is implemented in the GeneratedTerrain class and the compute shaders it refers to. Below is a simple flowchart of how changes to the terrain are managed.

There are 4 distinct elements that are part of this process as seen in Figure 30.

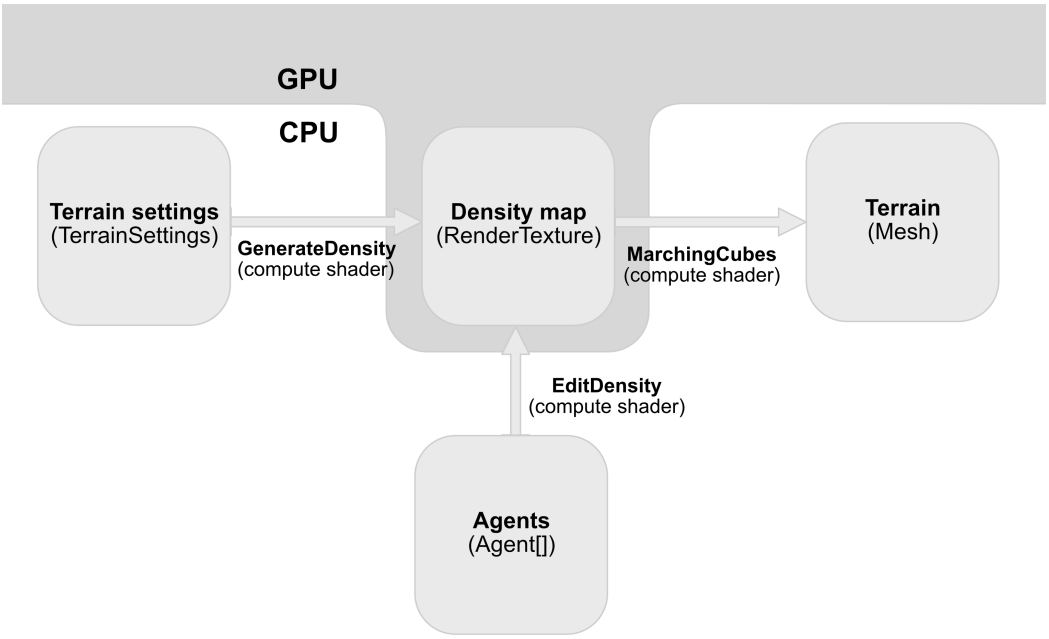


Figure 30. Data flow chart of the terrain implementation. Showing the main components of the terrain implementation.

1. Terrain: This is the terrain mesh on which the agents move and is visually represented. Whenever the mesh is edited a new collision mesh also needs to be generated to facilitate the physics collision with the agents. This is a computationally expensive process. Therefore, the terrain only recalculates its collision mesh when once per simulation step. Furthermore, the mesh is separated into chunks to only recalculate that specific chunk when it is changed as seen below in Figure 31.
2. The density map is the underlying data structure that is used to generate the terrain mesh. More specifically, it is the 3D scalar field that is passed to the marching cubes algorithm to generate the terrain mesh. This density map can be edited to 'excavate' or 'deposit' material. This will update the terrain mesh to represent that change.
3. The terrain settings: These are the parameters that are passed to the perlin noise function that ultimately influences the characteristics of the initially generated terrain. The parameters are described in further detail in the Interface section. These parameters are passed to a compute shader which samples the perlin noise function and returns the density map. This process is only done once at the initial generation of the terrain.
4. The agents are use a separate function to edit the terrain. This function only takes the parameters the amount of material being excavated up or deposited, and at which location to do so.

The interactions between the elements of this system are done with compute shaders, which take advantage of the parallel processing capabilities of the GPU which significantly speeds up these processes. However, when dispatching a compute shader to complete a certain computation, the CPU must send the required data over to the GPU using a buffer. In this case, that process becomes a bottle neck that drastically reduces performance because of the large number of agents that change the terrain using this method. To circumvent this bottleneck, the density map is stored as a 3D render texture, which is a Unity specific data type. This render texture is the specific datatype that is used to represent the scalar field that the marching cubes algorithm requires. A render texture does not need to be passed in a buffer, as it is permanently stored on the GPU, which negates the need for the CPU to process it as a buffer.

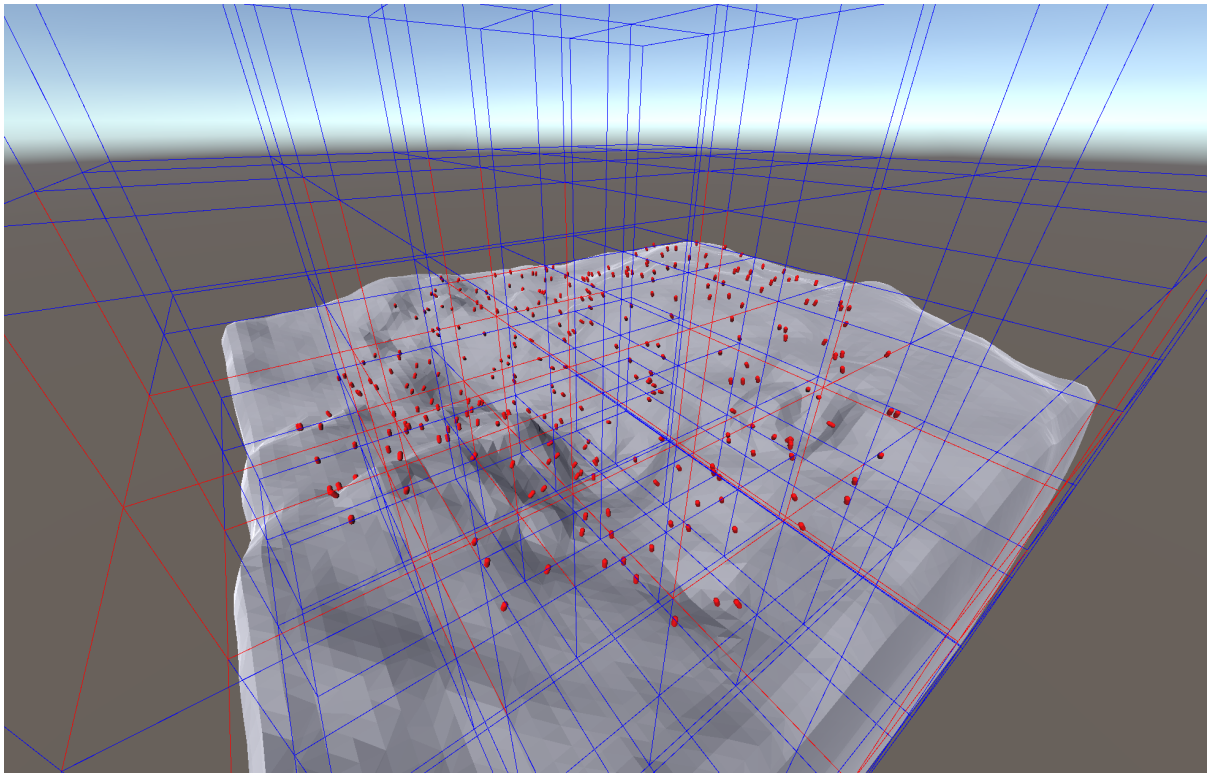


Figure 31. Terrain chunks. Showing the changed terrain that is updated this simulation step (Red outline) and the terrain that did not change in this simulation step (Blue outline).

2.4. The Swarm

How the swarm is represented is once again a decision between an appropriate yet feasible level of realism.

2.4.1. Options

Unity GameObjects

Within Unity, a GameObject is the base class for all entities represented within the game. A GameObject can easily be instantiated from within a Unity Script. It is possible to create a default agent prefab, which is a template from which multiple instances can be instantiated within the simulation. This prefab can contain the scripts that describe the behavioral rules and components that are required to perform the required interactions. This is the standard method of creating characters in Unity.

Pros:

- Allows for agents to have collision meshes, which facilitate realistic physics collisions with other agents and the terrain.
- It is a proven method, used in most cases for creating characters in unity.

Cons:

- Operations performed on GameObjects are computationally expensive.

DOTS

DOTS is shorthand for 'Data Oriented Technology Stack' which refers to a combination of technologies that implement a data-oriented design approach to entity management. Specifically, the Entity Component System, or 'ECS' is useful in the context of a swarm, and can be used to translate a swarm of GameObjects to entities. DOTS has only partially been released, and ECS has left the early-access state in December 2022 for Unity 2022.2 and later.

Pros:

- Same functional as GameObjects
- Significantly improved performance compared to standard GameObjects.

Cons:

- Fairly new technology that still has bugs, little to no documentation for implementation.

Complete Abstraction

Another method is to not instantiate the agents as unity objects, but instead as a custom class that does not contain a mesh to visualize the agent or a collision mesh to simulate collisions with. Those functionalities would need to be handled externally, outside of the definition of the agent.

Pros:

- Offers potentially the greatest performance.
- Abstraction can be done where needed throughout the project

Cons:

- Requires basic functionality to be manually handled externally.

2.4.2. Implementation

Unity entities are great for many unique objects in the scene, but for the purpose of termite simulation, that is not needed. A termite swarm of workers can be created where all of the agents in the swarm are the exact same, with no necessary option to further add capabilities during run time. Therefore, the complete abstraction approach is chosen.

Agents are presented in the simulator as very simplified representation version of termites to improve the performance of the simulator with a large-scale swarm. Agents Occupy space as a single point. This means that the agents have no shape or size, and that collisions and visualization of the agents have to be handled separately.

Visualization

Visualization is done by a separate script called the 'AgentVisualizer'. This script takes the positions and orientations of all agents. It then draws a predefined mesh at those locations using GPU instancing. GPU instancing is a very efficient way of drawing many instances of the same mesh, allowing tens-of-thousands of the same mesh to be drawn without any significant performance loss as demonstrated in Figure 32.

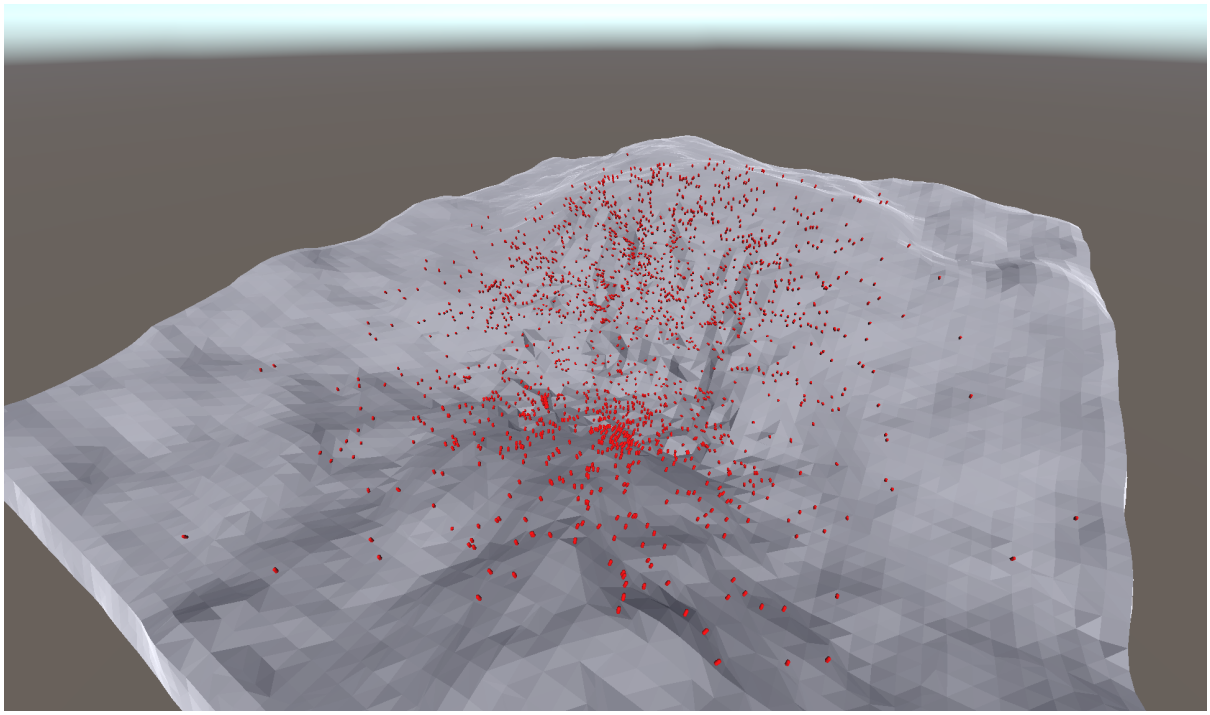


Figure 32. 2000 Agents exploring the terrain. Demonstrating a the capabilities of simulating large swarms.

Interaction with terrain

The agents interact with the terrain through raycasts, which draw a ray from the agent's location into a specified direction until a predefined distance. This ray collides with the terrain and measures the agents' distance to the terrain.

This is for instance used to check if the agent it on the ground as illustrated in Figure 33. A raycast is done from the agent position towards its feet. If the ray hits the terrain, then the agent is on the ground. If this is not the case, the agent will start falling. Falling moves the agent downwards at a constant speed. Falling does not happen often. It only occurs when another agent picks up too much material below that agent.

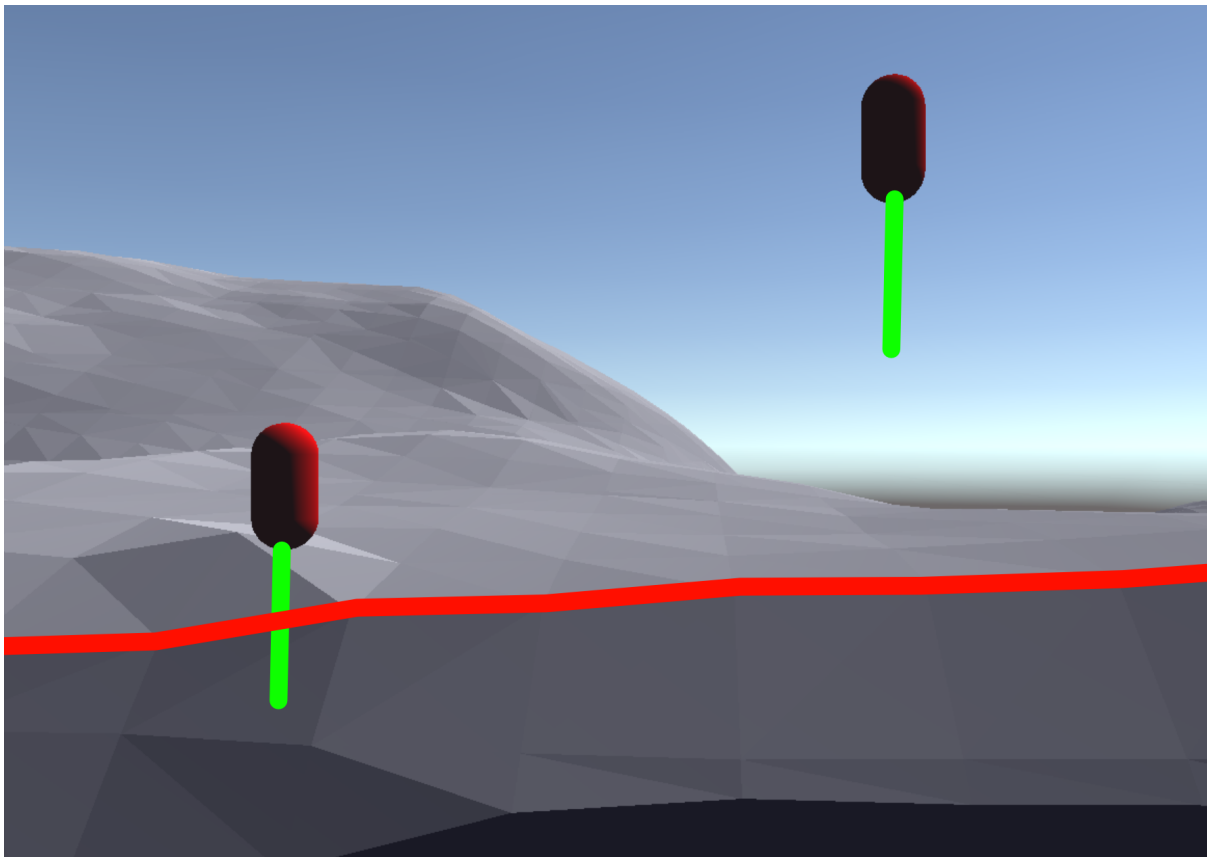


Figure 33. Grounded agent on the left and falling agent on the right. (Red line is the ground, green lines are the raycast that check if the agents are on the ground.)

Another problem occurs when another agent places too much material on top of that agent. The agent falls through the terrain because the terrain is just a surface mesh. To fix this issue, there is an additional check to see if the agent is inside the terrain illustrated in Figure 34. This is done by taking an arbitrary point outside the boundary of the terrain, and then drawing a line from the agent to that point and count how many times this line hits the surface. If this number is even, the agent is outside the terrain. If it is uneven, the agent is inside the terrain.

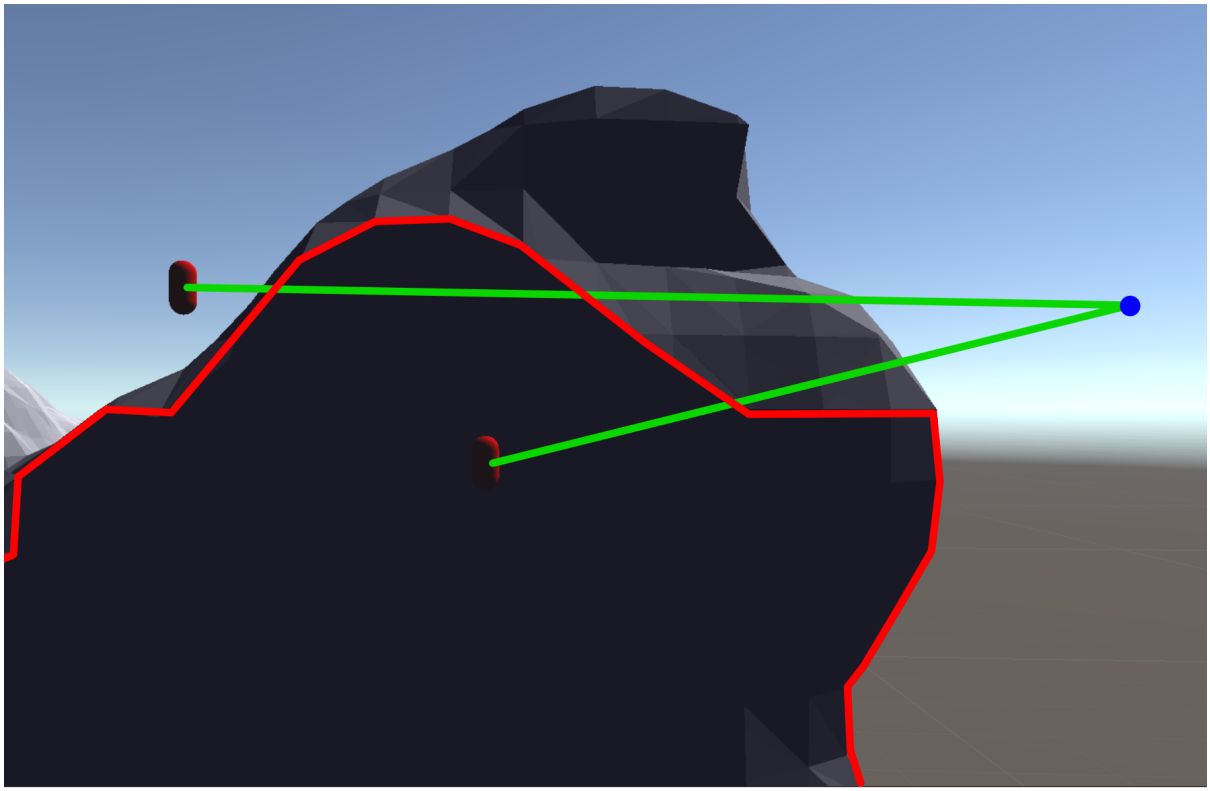


Figure 34. Agent inside terrain. (Red line is the terrain surface, Blue dot is the arbitrary point outside of the terrain, Green lines are the raycasts that count the number of times the surface was hit.)

To prevent the agents from falling off the terrain, when an agent reaches the edge of the terrain they are oriented back to the center of the terrain. The choice of having them re-orient themselves to the center was made to minimize the chances of agents getting stuck, as bouncing them off the edge or teleporting them to the opposite side would result in agents getting stuck at the edges of the terrain.

This was previously done with a distance from the center rather than a distance from the edge, which would cause the agents to stay within a certain radius of the center as seen in Figure 35. However, this caused agents to have a tendency to circle around this edge, impacting their behavior. It also prevented them from utilizing the full size of the terrain.

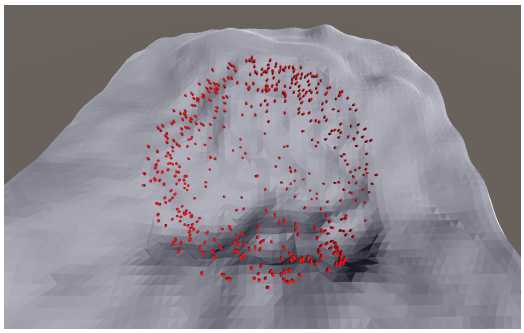


Figure 35. Agents not leaving the 'homesick' radius

Collision

Because the agents don't have any shape or size, raycasts can not be used to check for collisions with other agents. Instead, this is done by the 'AgentManager' class. This collision check simply checks if the distance from any other agent is within a predefined 'collision distance'. If that is the case, then the agents collide, which is currently implemented as bouncing off each other in opposite directions. There is also an 'avoid distance' which is larger than the collision distance and causes the agents to turn in order to avoid collisions with the other agents. In larger swarms, checking for all the other agents becomes increasingly inefficient, as the time complexity of every agent checking every other agent is O(N^2) where N is the amount of agents. Meaning computation time increases quadratically with the size of the swarm as visualized in Figure 36. To reduce this effect, the agents are first sorted into a grid. This grid is a cube structure, where the size of the cube is larger than the 'avoid distance' and 'collision distance'. By doing so, the other agents only have to check the agents in their own grid cube and the ones in adjacent cubes (Figure 37), instead of all the agents.

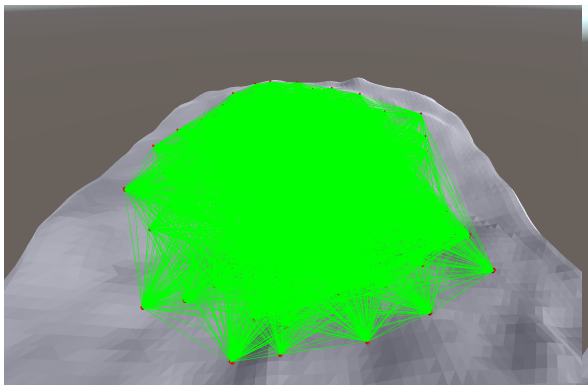


Figure 36. Visualization of proximity checks performed by every simulation step if the agents are not sorted beforehand. 100 agents, each green line is a proximity check. This shows that 10.000 proximity checks are made to check for collisions.

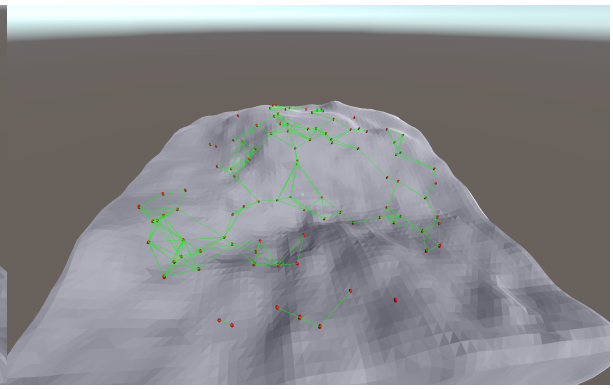


Figure 37. Visualization of proximity checks performed every simulation step if the agents are sorted beforehand. 100 agents, each green line is a proximity check. This shows a greatly reduced number of checks is necessary to check for collisions.

Code structure

The agents are specialized realizations of the abstract class 'Agent'. The Agent class provides functions such as StepForward, DigAt, SecretePheromone, and GetPheroneAt for the realization to choose from and decide when to use them. An example Agent realization is the PillarBuilder class which is described in further detail in Section 3.4. This is illustrated in Figure 38. To customize the behaviour of the swarm more than just changing the variables, one can create their own class that realizes the 'Agent' class.

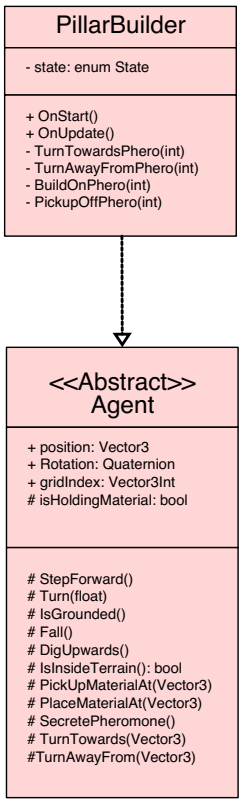


Figure 38. Class diagram of the PillarBuilder class

2.5. Pheromones

Pheromones are a key mechanism in the stigmergic communication between termites. Every termite will dispense pheromones many times during a simulation. Therefore, it is imperative that the performance of the chosen implementation is good.

2.5.1. Options

Terrain texture

One option is to implement the pheromones as colors on top of the terrain mesh. This simply paints the color on the texture of the terrain after which termites can measure the RGB values in front of them to sense the pheromones.

Pros:

- Good representation of the liquid stage of pheromones.
- Unlimited detail, as pheromones can be added in continuous space.

Cons:

- Does not allow it to travel through air, which greatly influences the pheromones dynamics.

Individual objects

Utilizing the same sorting grid that agents use to check for collisions, the pheromones can be objects that are placed in the scene, and sensed by checking their distance.

Pros:

- Easy to implement, just instantiating new objects and checking the distance to those objects is already achieved in previous code.
- Would allow objects to be placed anywhere.
- Unlimited detail, as pheromones can be added in continuous space.

Cons:

- The number of objects could increase to such an extend that it hinders performance, assuming that every agent releases pheromones multiple times in the simulation.

3D grid

Much like the density map of the terrain, each pheromone could have their own density map.

Pros:

- Allows pheromones to be added anywhere.
- The performance of a 3D grid implementation is constant.
- Resembles the gas state of pheromones, and allows gas physics to be implemented.

Cons:

- Detail is limited to the resolution of the grid.

2.5.2. Implementation

The 3D grid method is the only one that can accurately simulate the gas state of pheromones which is the most important aspect of this choice. The implementation is similar to how the terrain is implemented and also contains 4 elements as can be seen in Figure 39.

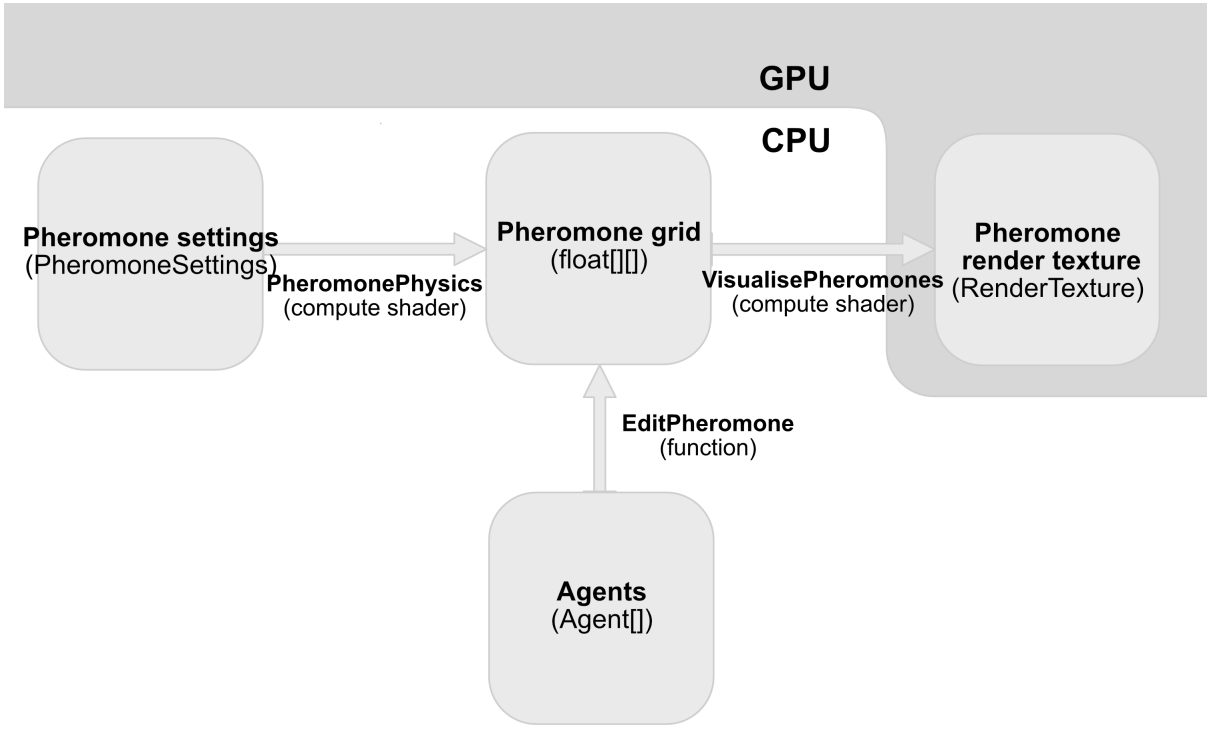


Figure 39. Pheromone flowchart. Showing the main parts of the pheromone implementation.

1. Pheromone grid: This is the density map with which the agents interact. Like the terrains density map, it is also a 3D scalar field, but its datatype is a 3D float array, unlike the terrains density map which is a RenderTexture. The use of a render texture was not possible for this implementation because the agents need to be able to both read and write individual points to the pheromone density map. A render texture only allows to be written to it. Each pheromone has its own pheromone density map that can be sampled independently.
2. Pheromone render texture: The density map by itself is visible to the agents because of their sampling function, but it is not visible to the user. To make the pheromones visible to the user, the information of the pheromone density maps is combined and translated into a RenderTexture. The RenderTexture is then sampled by a raymarching shader that samples the render texture and draws it to a 2D texture which is displayed on the users screen. The choice to first translate the pheromone density maps to a RenderTexture was for performance: the RenderTexture is stored on the GPU which makes sampling it with shaders, which do the calculations on the GPU, much faster.

3. Pheromone settings: These are the settings which determine the characteristics pf the pheromones, including decay rate, diffusion rate, and if they are effected by wind. These options are described in further detail below. These settings are used to run the pheromone gas simulations once every few simulation steps.
4. Agents: The agents simply read and write to the pheromone density map. They can add any amount of pheromone to any point in the grid, and can sample any position which will return the value of the closest pheromone grid point.

Pheromone physics

The gas physics are computed on each point on the pheromone density map grid separately, using a compute shader. As this process is also quite computationally expensive, there is an option to only do these computations once every N amount of simulation steps, with N being a variable in the optimization settings called ‘phero update delay’. By using this, the effect of the pheromone gas simulation is increased so its dynamics will still be similar over a larger time frame. Increasing the ‘phero update delay’ improves performance, but will result in agents working with slightly outdated pheromone data.

The gas physics include:

- Decay, which decreases the pheromone concentration by a constant amount each execution. Useful to prevent pheromone build-up where the ambient pheromones keep increasing because more pheromones are added than removed. It also prevents old pheromone trails from staying around forever.
- Diffusion, which proportionally diffuses pheromones, causing a gradient of pheromones. Useful for guiding agents to the centre of a pheromone source.
- Wind: which diffuses pheromones into a cardinal direction also including up or down. The updraft in the termite mound caused by the ventilation is likely to impact the mounds architecture. This is made to simulate that.

In the current version the pheromones permeate terrain. This allows pheromones to move through the walls of the royal chamber or covered galleries, which is not realistic. Therefore it is not an accurate gas physics simulation. This is further described in Section 4.1.

Interpolation

Although the pheromones are in a grid, interpolation can be used to get more detail from this grid. When placing pheromones at a certain location, the easiest method is to take the nearest grid point and add the pheromones to that point. Another approach is to calculate the distance between the point of placement and the nearest grid points and distribute the pheromones over those grid points as a function of how close the placement location is to the grid points. This difference can be seen in Figures 40 and 41.

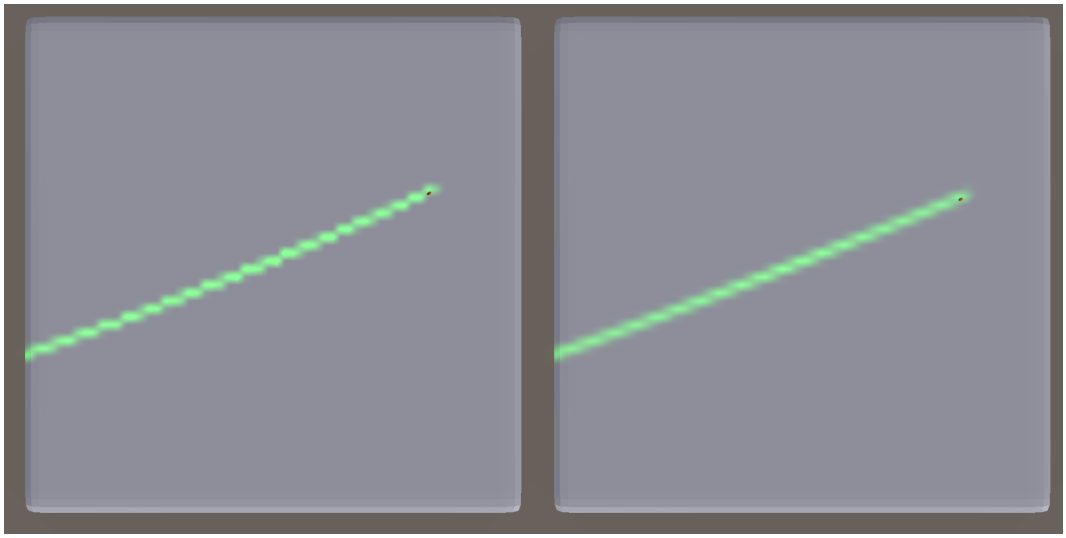


Figure 40. A single pheromone trail (green) without pheromone interpolation. The trail clearly shows the discrete steps of the pheromone grid.

Figure 41. A single pheromone trail (green) with placement interpolation. The trail is much smoother and the steps of the grid are much less pronounced.

Whether or not pheromones are placed into the grid also changes affects the resulting behaviour of the agents. In Figures 42 and 43, 100 trail laying agents move around. These agents continuously secrete the trail pheromone, and also turn towards that trail pheromone if it is in front of them. This causes a positive feedback-loop from which trails emerge. This behaviour is based on the findings in Bruinsma's thesis as described in Section 1.4.2.

When agents don't use interpolation to place the pheromones, the trails that are formed have a tendency to follow the cardinal directions of the grid. This is mitigated by using interpolation, but trails created with interpolated pheromone placing are much wider as a result of the agents placing the pheromones over a larger area. Because the latter is an unintended side effect, placement interpolation can be turned of in the settings menu which is described in more detail in Section 2.6.

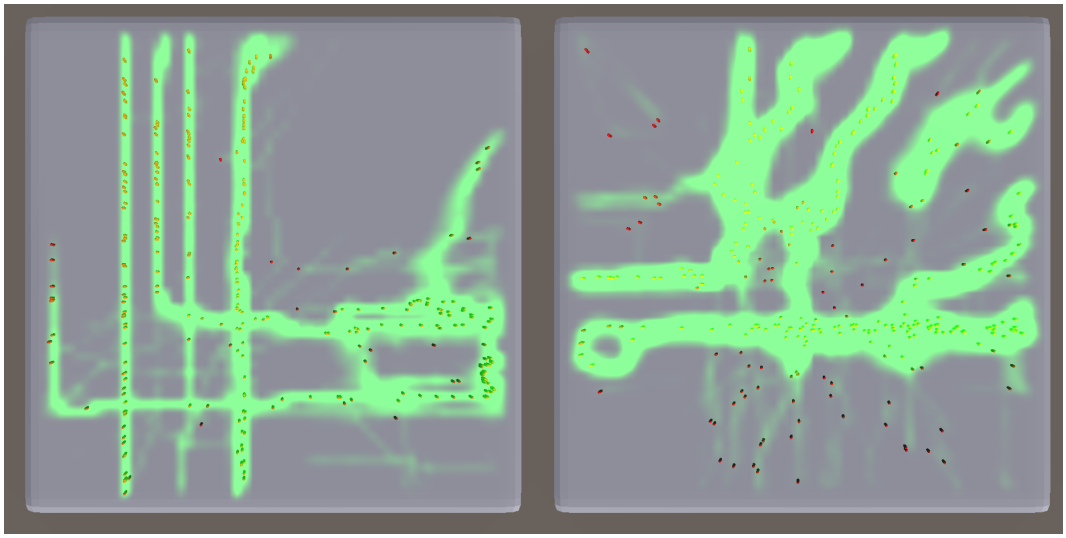


Figure 42. A swarm of 100 trail placing agents (red) creating a trail (green) without pheromone placement interpolation. The trails have a tendency to follow the cardinal directions of the cubic pheromone grid.

Figure 43. A swarm of 100 trail placing agents (red) creating a trail (green) with pheromone placement interpolation. The tendency to of the trails to follow the cardinal directions of the pheromone grid is much lower.

Besides using interpolation to place pheromones, it can also be used to sample pheromones at a location, instead of returning the pheromone concentration of the nearest grid point. The process of this is the reverse of placement interpolation. For sampling interpolation, the pheromone concentration of the nearest grid points are sampled, weighted based on distance and then combined to a single value that more accurately represent the pheromones in the location of sampling. The effect of this is also that the trails that emerge have a lower tendency to follow the cardinal directions of the grid as seen in Figures 44 and 45.

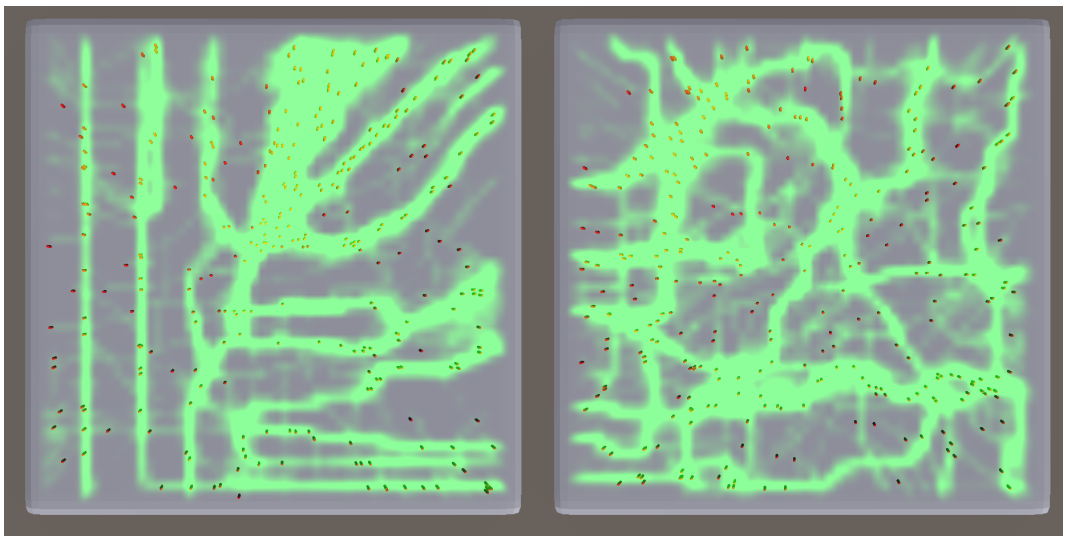


Figure 44. A swarm of 300 trail placing agents creating a trail without pheromone sensing interpolation. The trails have a tendency to follow the cardinal directions of the cubic grid in which they are placed.

Figure 45. A swarm of 300 trail placing agents creating a trail with pheromone sensing interpolation.
The tendency to of the trails to follow the cardinal directions of the grid is much lower.

Lastly, both placement interpolation and sampling interpolation can be used to completely mitigate the trails tendency to follow the cardinal directions of the pheromone grid. Thereby, creating a much more organic looking trail, which is likely a better approximation of real termite trail forming (Figure 46). The interpolation of placing and sampling is slightly more computationally expensive, but it still constant time (now O(8) with interpolation instead of O(1) without interpolation) and does not significantly impact performance.

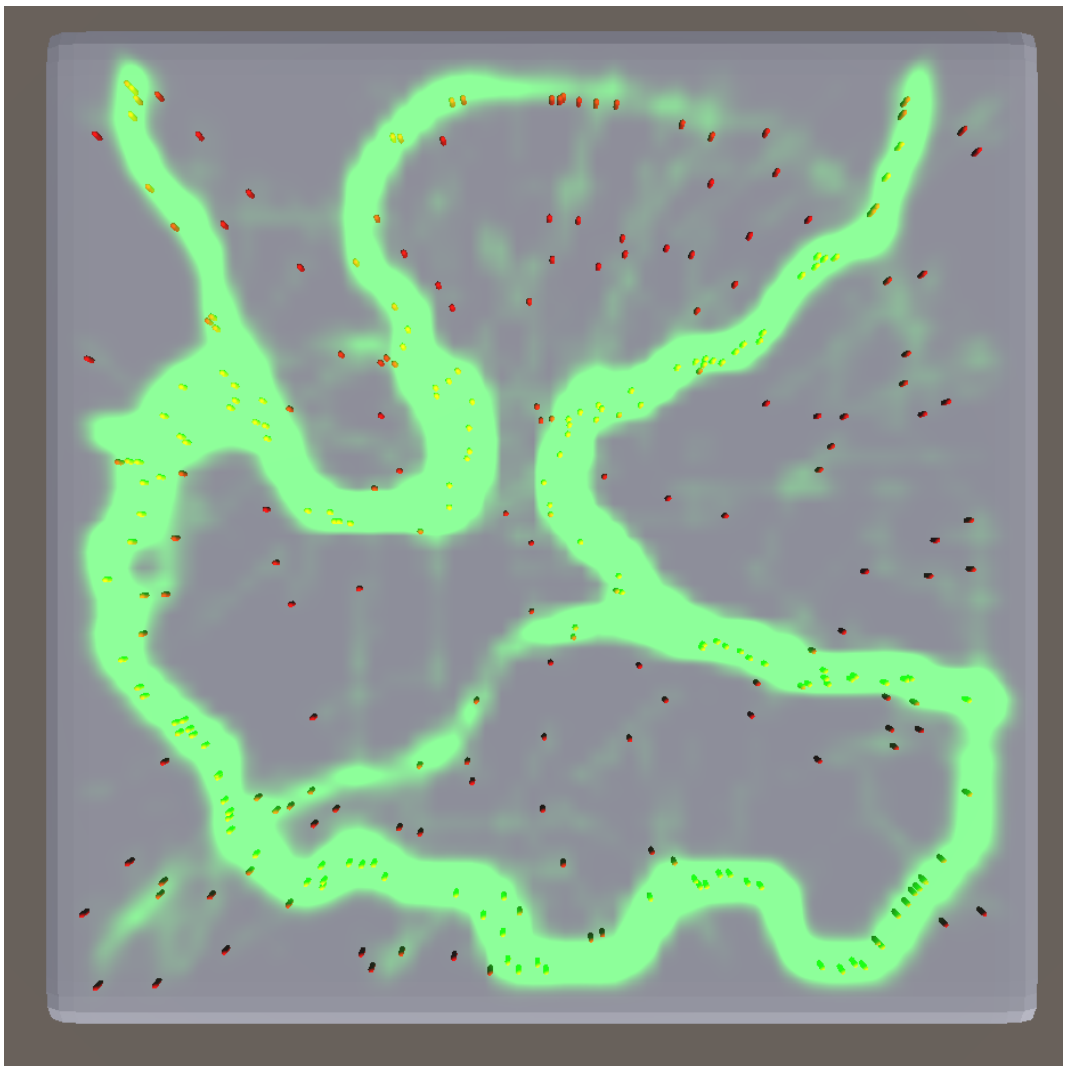


Figure 46. A swarm of 300 trail placing agents creating a trail with both pheromone sensing and placement interpolation. The tendency of the trails to follow the cardinal directions of the grid is completely mitigated.

2.6. Interface

The interface is the link between how users interact with the system. It is used to create or load a simulation, and adjust the parameters of the simulation as seen in Figure 47. The interface consists of two parts: the header which is used to control the simulation, and the the settings menu which is used to tune the parameters of the simulation.

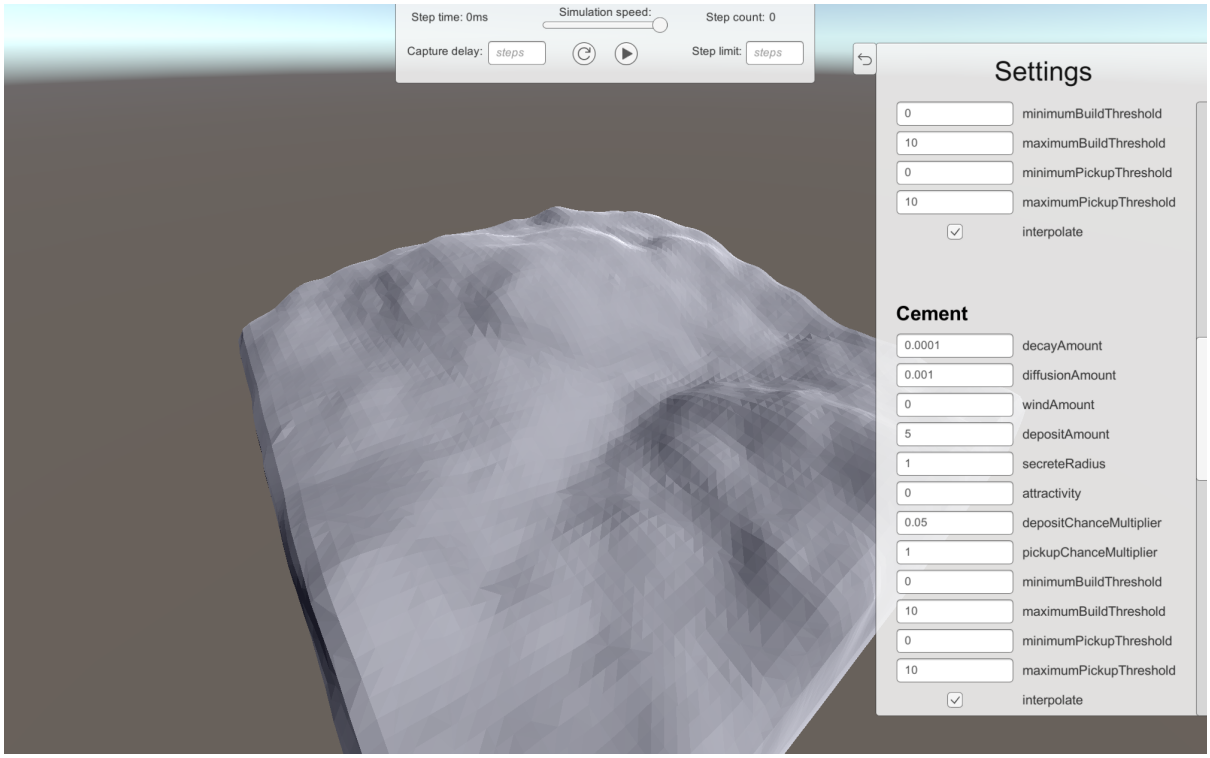


Figure 47. Interface with the header on top and the settings menu opened on the side.

2.6.1. Implementation

Settings menu

The settings are lists of variables/parameters that other scripts refer too. The settings menu is automatically generated based on the references the settings manager holds and the variable fields that those references classes have. This allows future development to simply add a variable to any of the settings, for which corresponding value selector will then be automatically generated in the settings menu during run time. These settings are stored in ScriptableObjects which are a specialized Unity datatype that is persistent outside of running simulations. Meaning that changes made to the settings are saved and not reset upon the start of a new simulation. The settings menu can be collapsed by pressing the arrow button on the left side of the menu.

The settings are separated based on the four distinct types of settings:

- The terrain settings: These are the parameters that are passed to the perlin noise function that ultimately influences the characteristics of the initially generated terrain.
- Pheromone settings for each of the pheromones: the rate at which pheromones diffuse, decay, and move with the wind.

- Agent settings: which include the behavioral as well as the hardware settings such as movement speed and pheromone sense distance. Tweaking these settings can result in different behaviour within the same control strategy.
- Optimization settings: changes that can affect performance. Such as in how many chunks the terrain is divided.

In the code, settings are referenced through the settings manager which is a mediator design pattern which holds references to all the different settings. This allows the settings to be easily swapped out both outside and during run time.

Header

At the top of the screen is the header as seen in Figure 48. The header holds the option that control the simulation.

- The play button is used to start, pause, and unpause the simulation.
- The reset button resets the simulation to its starting point. This clears any changes made to the terrain, removes all pheromones, and places all the agents back to where they started.
- The speed slider is used to slow the simulation down. This allows the user to look at each step individually in case the simulation is too fast. If the simulation speed slider is completely to the right, the simulation will run as fast as the host computer allows it.
- The step time indicator simply states how long each simulation step takes to compute.
- The step counter indicates on which time step of the simulation it is currently on.
- The step limit is used to automatically pause the simulator after the step limit is reached. This is especially useful for changing settings during the simulation. For instance, if the queen pheromone needs to diffuse before the agents are allowed to build, the user can set the step limit to 1000 and change the settings when the simulator is automatically paused.
- The simulator has a camera which can automatically take pictures of the simulation. The capture delay sets the interval on which the camera takes pictures. These pictures are saved in assets/Captures

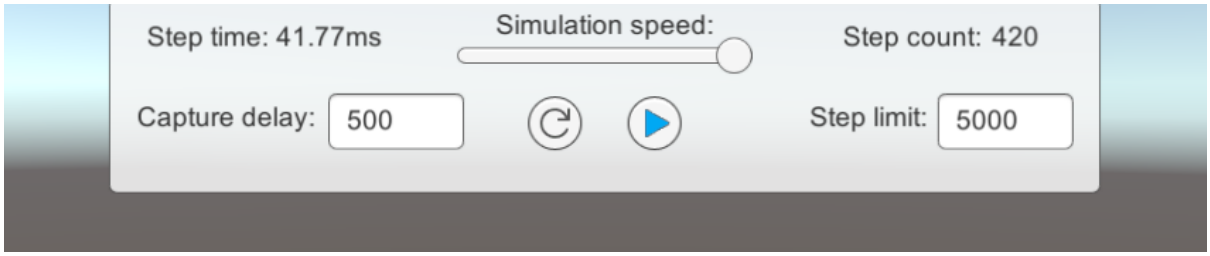


Figure 48. The header while running a simulation showing that the interface looks like with the settings menu collapsed.

2.6.2. User tests

In this section feedback was gathered to suggest improvements for the interface. A good description of what is required of the usability is described in a quote from Dorigo m., et al. 2021 from the literature review 'Swarm Robotics: Past, Present, and Future': *"Simulations should be highly configurable to respond to the needs of a diverse research community. At the same time, setting up a new simulation configuration should not require expert knowledge of the inner working of the software."*

There are two facets on which the simulation can be configured:

1. The relation of what input causes what output, which is the control strategy. In TermiteSim the control strategy is defined in code, which gives it the freedom for any type of behaviour to be created. However, this code has to be compiled for it to work in the simulator. This is not possible within TermiteSim itself and requires Unity to compile the code and rebuild the project. An example of a research question that can be answered with this functionality is: Can Greens theory of termite aggregation at excavation sites also cause the emergence of pillars?
2. The parameters that describe the extend to which those relations occur. These parameters are adjustable in the settings menu from within TermiteSim. An example of a research question that can be answered with this functionality is: Does a slower pheromone decay rate increase the height of the pillars formed from aggregated deposition?

Method

To evaluate this a user test will be conducted with a researcher outside of the project. The success of the evaluation will be concluded from the time it takes for them to perform a number of tasks and the number of times they require assistance outside of the initial explanation.

The participant was given the following description:

Termites use pheromones to communicate. One of those is the cement pheromone which causes aggregated deposition. When a termite places a soil pellet, that soil pellet is infused with the 'cement' pheromone. When other pellet carrying termites are in the presence of the cement pheromone, it causes them to place that soil pellet, which also contains cement pheromone. This causes a positive feedback loop. Over time equally spaced deposition zones emerge which, over time, turn into pillars. That later form the walls of the termite mound.

TermiteSim is a simulator that can simulate termite-like building behaviour. It does this by simulating a swarm of termite-like builders, which are called agents. These agents can move around, pickup and place material, and secrete pheromones. In this simulation the agents run a simplified version of the aggregated deposition behaviour. It causes them to move around in an area, pick up material and place it down with 'cement' pheromone. Placing down a soil pellet happens randomly, but the chance to place a soil pellet is greatly increased in the presence of the cement pheromone.

The participant was then presented with TermiteSim and was asked to try to replicate pillar formation from aggregated deposition in 3 steps:

1. *Your goal is to try to get those deposition zones to appear by changing the parameters of the simulation.*
2. *Now try to get pillars to form from those deposition zones.*
3. *Try to increase the height of the pillars.*

To complete these steps the participant will need to perform the following actions:

1. Identify the settings menu
2. Scroll the setting menu
3. Find the 4 relevant parameters for this task
4. Change the parameters where necessary
5. Identify the start button to start the simulation
6. Use the camera controls to inspect the structure
7. Interpret whether the change led to the desired result.
8. Identify the reset button to reset the simulation
9. Iterate on tuning the behaviour

Result

The participant took 15 minutes to complete the 3 tasks and asked 8 questions during the task. Afterwards they spend another 25 minutes on trying to implement their own ideas with the simulator.
During the test, the participant made the following suggestions:

1. Many of the parameters in the settings menu relate to each other. Perhaps nesting the parameters in sub sections makes this relation implicit.
2. The settings menu contains many parameters, of which not all are equally relevant. I suggest that you make a distinction between the commonly used parameters, and advanced parameters, and then hide the advanced option by default. If a user wants to use the advanced settings, there should be an option to show advanced settings to unhide them.
3. Some of the names of the parameters are not self explanatory. I suggest you add tooltips to the parameters with a description of their function.
4. In order to evaluate the resulting structure, it would be useful to be able to export the terrain.

Discussion

During the test, 6 out of the 8 questions that were asked were on the meaning or effect of individual parameters. Once the participant became familiar with the relevant parameters for the completion of the tasks, the participant was able to autonomously complete the tasks. The effect of the parameters does not necessarily need to be explained, as it can be understood through changing the values and seeing the results. That, with the naming of the parameters should allow users to gain an understanding if the effect of the parameters without being explicitly told.

It is interesting that the participant kept using the simulator for 25 minutes of their own volition after the test had concluded. The participant was told multiple times that the test had ended, yet kept changing the parameters and seeing if the behaviour improved. This indicated a willingness to use the simulator more. The participant also made a remark on the pheromones not disappearing on the reset of the simulation, but only after starting the simulation after a reset. This was a bug and has since been fixed. Lastly, the participant was surprised that the simulator did not stop by itself. This resulted in the addition of the step limit to the header, as described in Section 2.6.1.

Conclusion

The suggestions made by the participant would be great additions to the simulator. The current state of the interface offers functionality, but could improve in usability. The participant asked 8 questions for the 9 actions that needed to be performed. However, 6 of these question were about only 1 of the action. The action in question was finding the relevant parameters. The other actions were performed with little to no help required. The interface is sufficiently usable given that the user is familiar with the parameters.

2.7. Program structure

A simplified UML class diagram can be seen in Figure 49. Both the GeneratedTerrain and AgentManager class inherit from Unity's MonoBehaviour class. Meaning these classes will be attached to a Unity GameObject in the scene and their Start and Update functions will be called on the Start of the program and on every Update/frame, respectively. The GeneratedTerrain class generates the terrain based on the settings it receives from the SettingsManager. It also holds the reference to the mesh as a private variable and any changes made to the terrain will be handled by the GeneratedTerrain class. Similarly for the pheromones, who also can only be accessed through here.

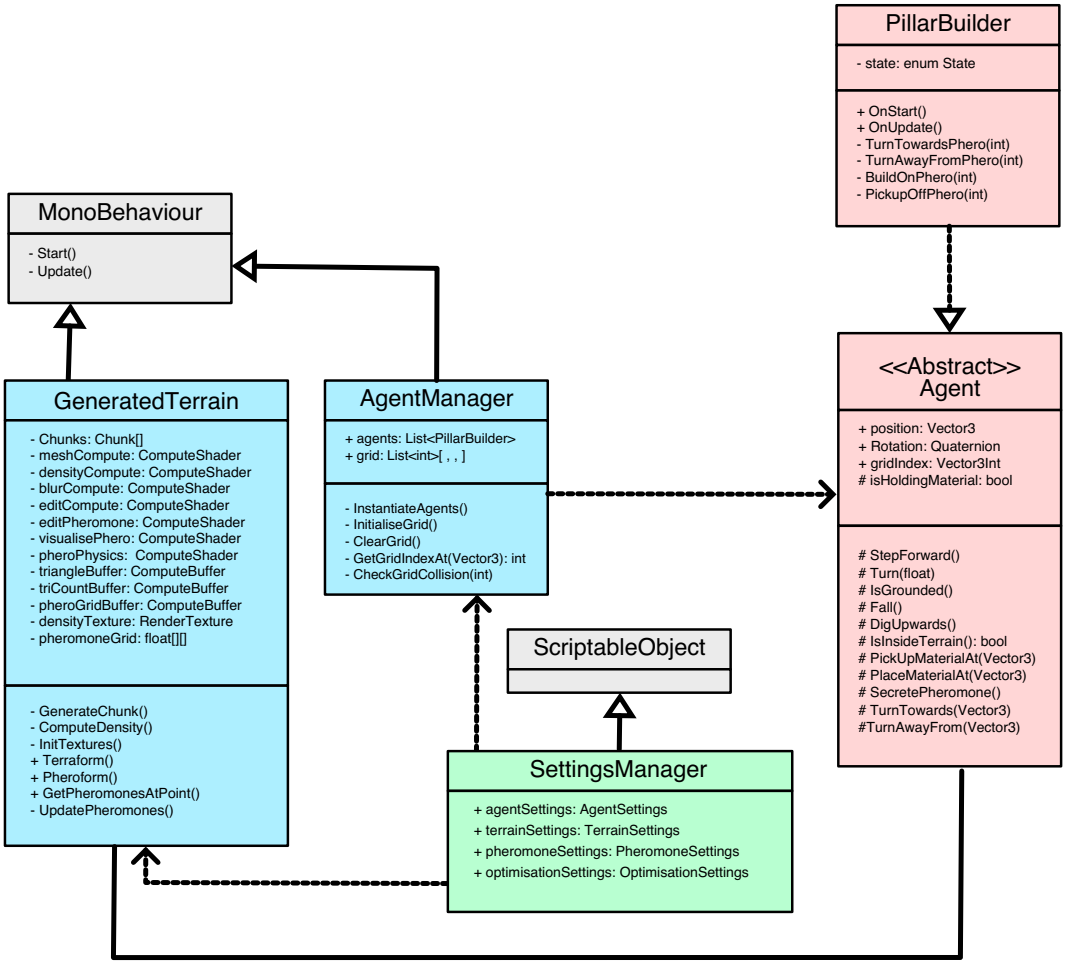


Figure 49. Simplified Class diagram of TermiteSim

The AgentManager class instantiates the PillarBuilder class and manages interaction between the members and thus has a dependency relation with the PillarBuilder class. It follows the Manager deign pattern.

The PillarBuilder class is the implementation of the abstract Agent class. The PillarBuilder class is just an example implementation of the abstract Agent class. Other implementations that don't exhibit termite-like behavior could also be created by creating a new realization of the Agent class.

The Abstract Agent class is the foundation of any Agent realization and provides its realization with the ability to move and interact with the terrain and pheromones.

The SettingsManager is holds static references to the different types of settings that are used to fine tune the behaviour. Using the mediator design pattern, it reduces chaotic dependencies and allows the settings to be saved and swapped more easily. This class also inherits from Unity's ScriptableObject class which gives it persistence, allowing its values to be saved.

2.8. Conclusion

TermiteSim now has a 3D environment with a terrain that is able to replicate the organic shapes of termite-like building, simulate a swarm of hundreds of agents that can perform the tasks as described by Bruinsma, simulate a multiple pheromones, and finally, an interface to easily change the behavioral parameters. Now that all these functional requirements have been met, TermiteSim can now be used to test whether this functionality does indeed give it the capability to simulate termite-like building behaviour.

3. Evaluation

Now that the simulator as described in Chapter 2 meets the requirements stated in Section 1.6, it can now be used to replicate the observed termite building behavior as described in the literature (Section 1.4). To do so, TermiteSim will be used to recreate trail forming behaviour (Section 3.1), Royal chamber construction, (Section 3.2), Covered gallery construction (Section 3.3), and Pillar construction (Section 3.4). Then dynamically formed trails are used to create entrances in the royal chamber construction in Section 3.5. Lastly, a simple simulation of a cleaning robot swarm will be done in Section 3.6 to demonstrate that TermiteSim can potentially also simulate outside the scope of termite-like behaviour.

To evaluate if the required functionality does indeed give the simulator the capability to reproduce termite-like building behavior, the resulting structures of Sections 3.2 and 3.3 will be compared in their respective sections to the resulting structured produced by similar simulations of Ladley and Feltell as described in Section 1.5.

3.1. Trail forming

As described in Section 1.4.2, one of the pheromones that termites use is the trail pheromone. This pheromone is used in creating trails which attract other termites to follow those trails, thus creating a positive feedback loop of trail forming.

TermiteSim includes a control strategy that models this behaviour by having the agents follow 3 rules that they execute every step:

1. Agents move forward.
2. Agents turn towards the highest pheromone concentration they sense with their antennae.
3. Agents place trail pheromone at their location.

Through iteration a set of parameters is found that consistently produces the desired behaviour of trail forming. The set of parameters for this simulation can be seen in Table 4.

The progression of trail forming with these parameters can be seen in Figures 50 to 58. Trails start out as separate low concentration trails left by individual agents, but they gradually merge together to form strong, high concentration trails.

Number of agents	300
Inter-agent collisions	Off
Decay	0.001
Diffusion	0.005
Deposit amount	0.02
Attractivity	20
Antennae length	2

Table 4. Relevant parameters used for the simulation of dynamic trail forming.

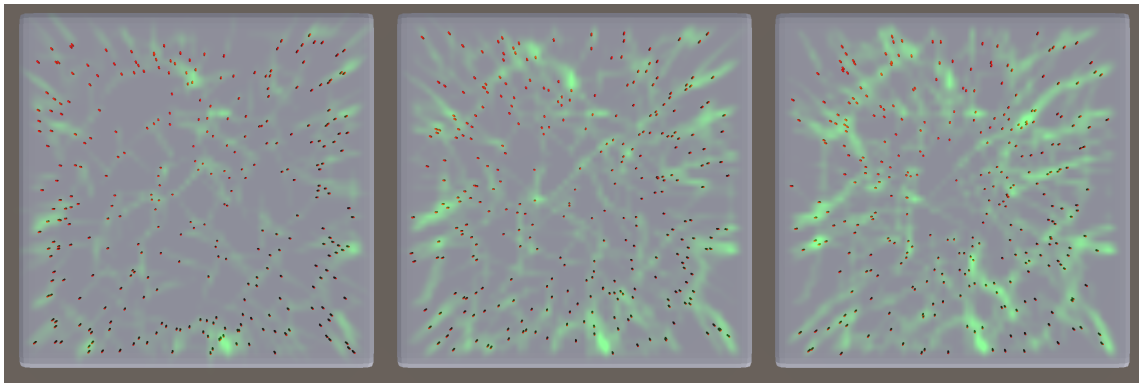


Figure 50. T = 50. The agents randomly move around.

Figure 51. T = 100. Some areas with slightly higher pheromone concentrations appear.

Figure 52. T = 150. The first indications of trails start to appear.

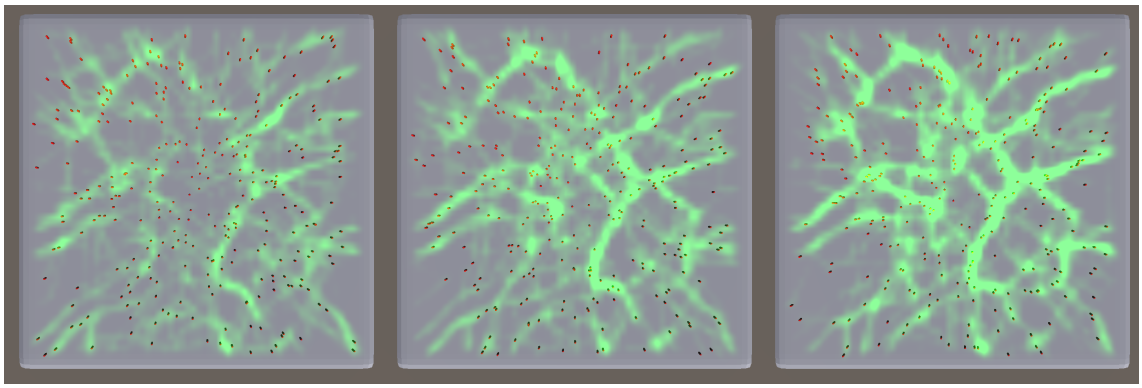


Figure 53. T = 200. Trails become more defined.

Figure 54. T = 300. The initial trails reach the maximum concentration in some areas.

Figure 55. T = 400. The trails strengthen and become wider.

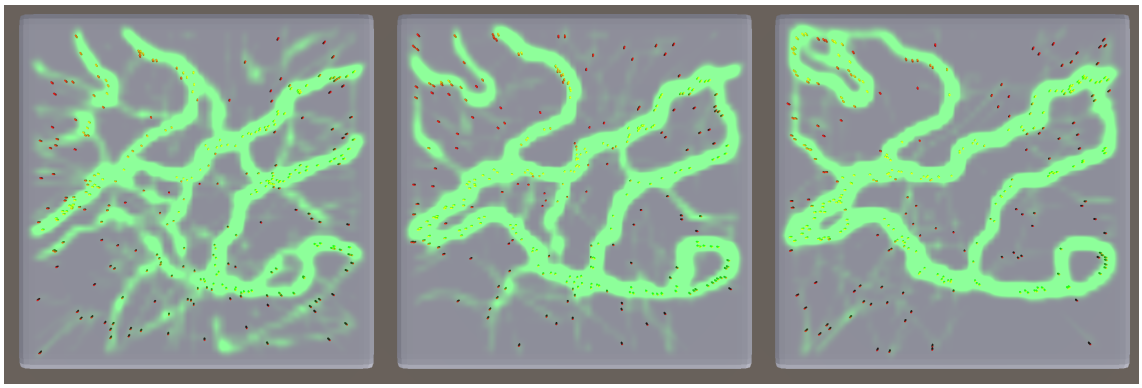


Figure 56. T = 800. The main trails are now clearly defined.

Figure 57. T = 1600. The number smaller trails reduce, while the main trails become more pronounced.

Figure 58. T = 2400. The trails have reached a stable state. Most of the agents travel within the trails.

3.1.1. Parameter influences on trail forming

TermiteSim allows the tuning of parameters through the settings menu. These parameters affect the behaviour of the agents, and pheromones. In this section, several parameters are individually changed and the resulting pheromone trails are compared to the baseline trail that was generated in the section above using the parameters in Table 4.

Antennae length in trail forming

Agents turn towards pheromones they are attracted to, and turn away from pheromones they are repelled by. They do this by sampling the pheromone concentrations at the ends of their antennae. The antennae length determines the distance from which the agent will be able to sense pheromones. The larger the antennae length, the further away from their position pheromones will be sampled. In the base settings the antennae length is set to 2. To see the effect increased antennae length has on trail forming, the antenna length was changed to 10 and the simulation was done again. As can be seen in Figure 59, the trail that is formed is much smoother and rounder. One explanation of this is that when agents approach a sharp corner in the trail, the agents start turning earlier because of the increased antennae length. This causes them to stray of the trail and 'cut corners' and create a new trail which is less sharp as is illustrated in Figure 60.

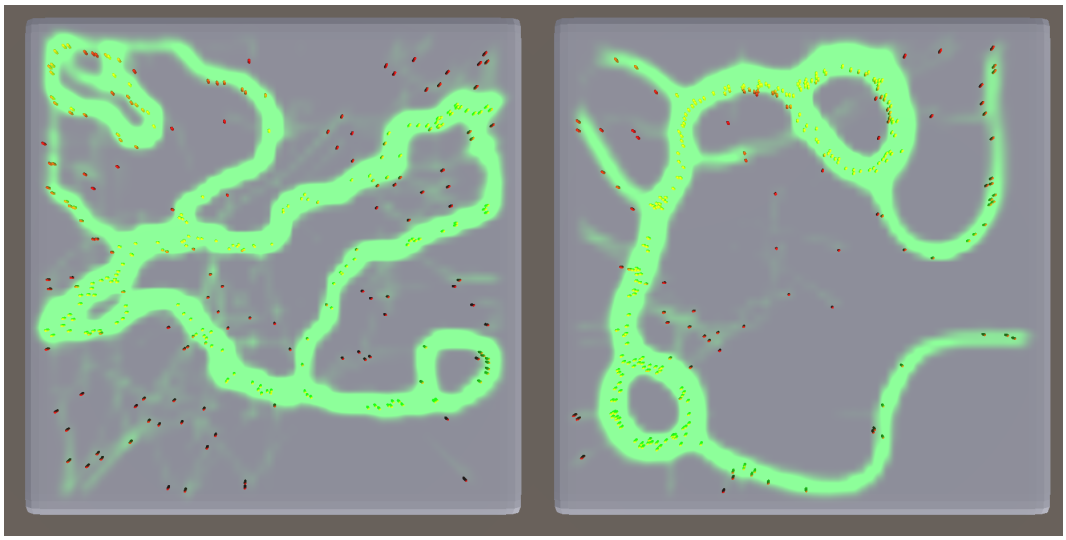


Figure 59. Comparing increased antennae length (Right) to the base settings (Left) at T = 2400. Smoother and rounder trail forming.

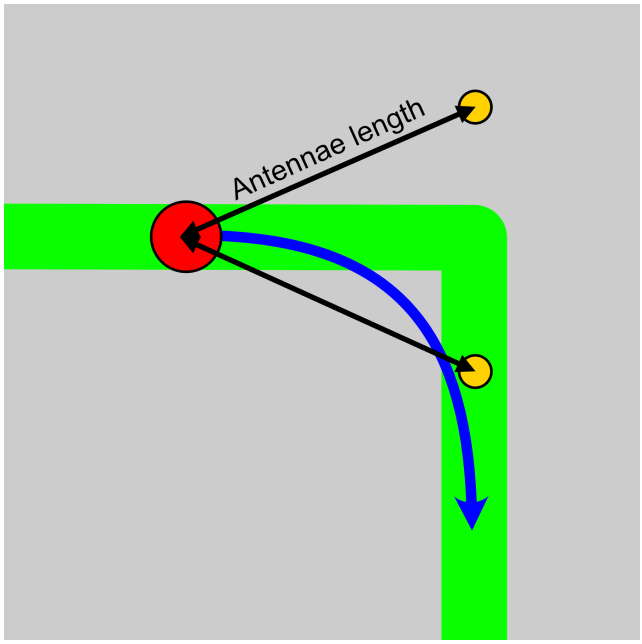


Figure 60. Illustration of corner cutting due to increased antennae length. (Red is an agent, Yellow is the antennae sample points, Green is the existing trail, blue is the path the agent will follow.

Attractivity in trail forming

When an agent senses a pheromone, how much it turns towards that pheromone is determined by the attractivity. The higher the attractivity, the sharper the turn towards the pheromone. A negative attractivity value will result in the agent being deterred by the pheromone.

The effect of attractivity on trail forming can be seen in Figure 61. With decreased attractivity, the agents don't follow the trails, and the the resulting trails are poorly defined. An increase in attractivity leads to less agents straying from the trail, and trails being better defined.

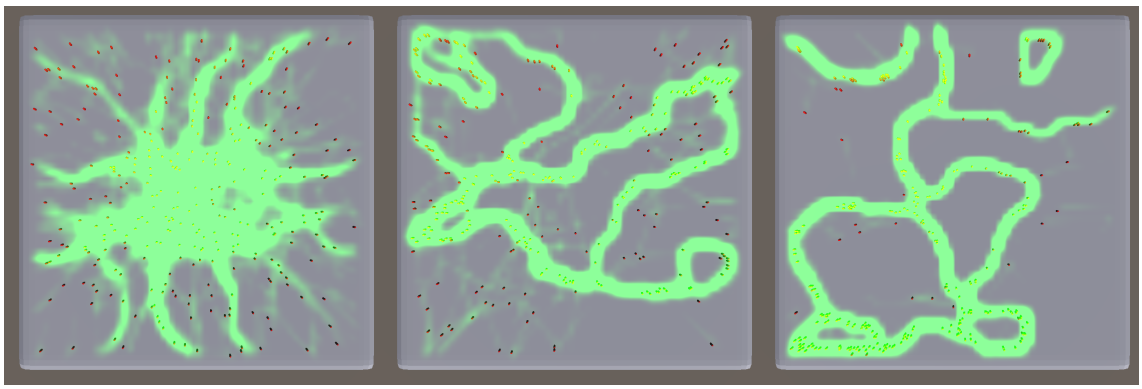


Figure 61. Comparing decreased attractivity (Left) and increased attractivity (Right) to the base settings (Centre) at T = 2400. Decreased attractivity leads to less agents staying on the trail and a fuzzier, less defined trail. Increased attractivity leads to more agents staying on the trail and a more defined trail.

Pheromone diffusion in trail forming

Once pheromone is placed, it diffuses through the space. How much the pheromone diffuses influences how long trails will persist after placement, and how large the gradient at the edges of the trail will be.

As can be seen in Figure 62, decreased diffusion leads to more trails, which are thin but well defined. There is also a tendency for the trails to follow the cardinal directions. This is likely due to the fact that the pheromone is placed into a cubic grid. Increasing the diffusion causes the trail to become less defined, and only one trail was formed during the simulation. This is because most trails disappear before they become strong enough to attract enough agents to maintain the trail.

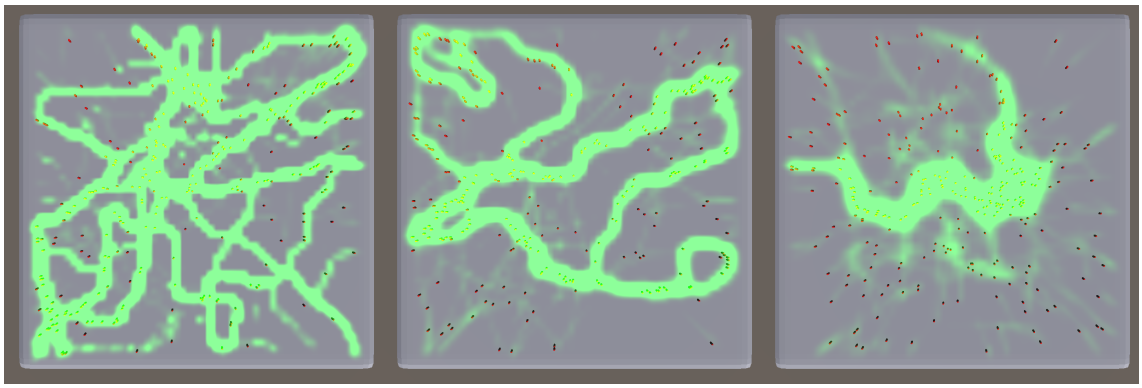
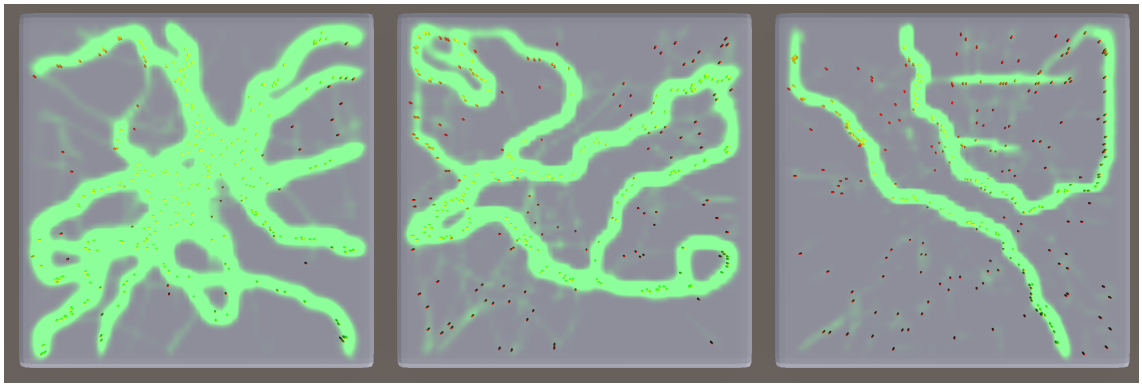


Figure 62. Comparing no diffusion (Left) and increased diffusion (Right) to the base settings (Centre) at T = 2400. Decreased diffusion leads to many more trails with a higher tendency to follow the cardinal directions of the grid. The trail that emerges with increased diffusion is less defined and does not reach far from the center.

Pheromone decay in trail forming

Pheromones decay at a constant rate. This makes that pheromones fade over time and eventually disappear. Without decay, pheromone trails will persist forever and with new pheromone being added, the amount of pheromone in the simulation will only increase until the simulation space is completely filled with pheromone. Decreasing pheromone decay leads to more and wider trails being constructed, where more agents stick to the trails, as seen in Figure 63. Increasing pheromone decay leads to fewer and thinner trails being created with fewer agents on the trail.



Pheromone trails showing the effect of decreased decay with 300 trail following agents at T=2400. More and wider trails emerge. Most agents stay on the trail.

Pheromone trails with the base settings with 300 trail following agents at T=2400. The emergence of a strong, clearly defined trail.

Pheromone trails showing the effect of increased decay with 300 trail following agents at T=2400. Fewer and thinner trails emerge.

Figure 63. Comparing decreased decay (Left) and increased decay (Right) to the base settings (Centre) at T = 2400. Decreased decay causes to more and wider trails emerge. Most agents stay on the trail. Increased decay leads to fewer and thinner trails with less agents on them.

Moving trails

During these tests, one unintended behavior was observed. The trails, once solidified, don't remain in place. They shift, and change shape over the course of a few thousand simulation steps, as can be seen in Figures 64 to 72. The cause of this, and its implications are further described in Section 4.1.

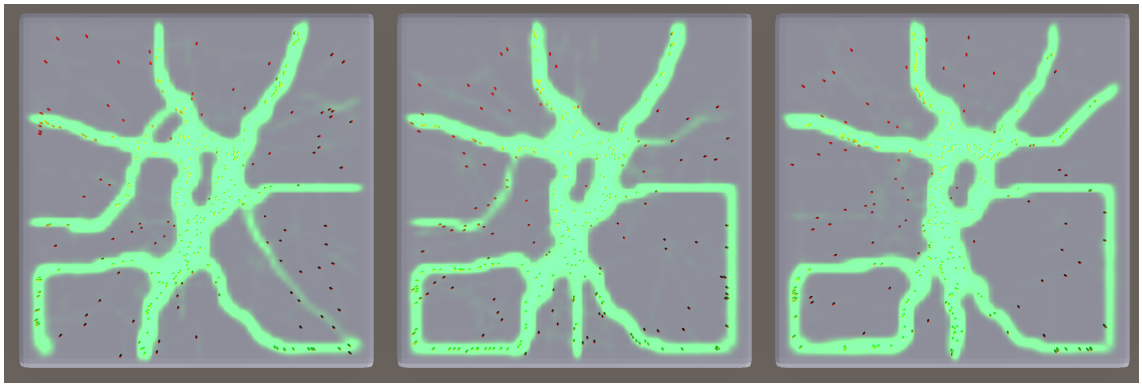


Figure 64. Shifting trails at T = 2000.

Figure 65. Shifting trails at T = 4000.

Figure 66. Shifting trails at T = 6000.

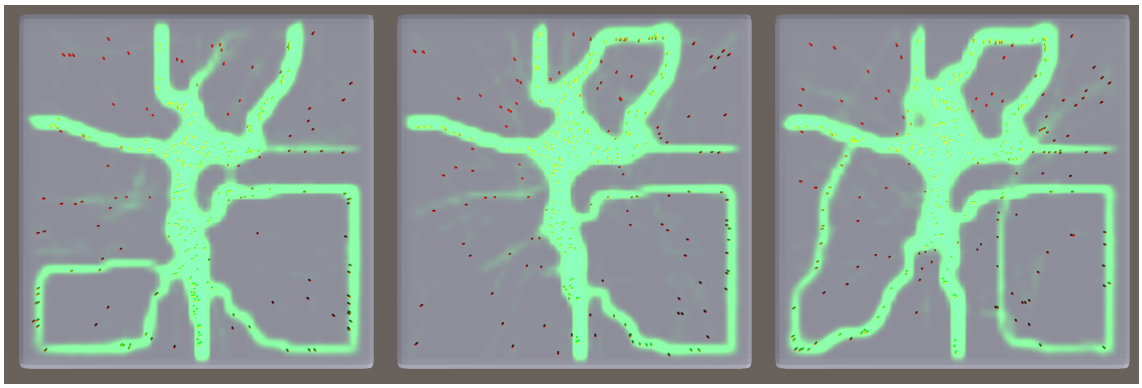


Figure 67. Shifting trails at T = 8000.

Figure 68. Shifting trails at T = 10000

Figure 69. Shifting trails at T = 12000

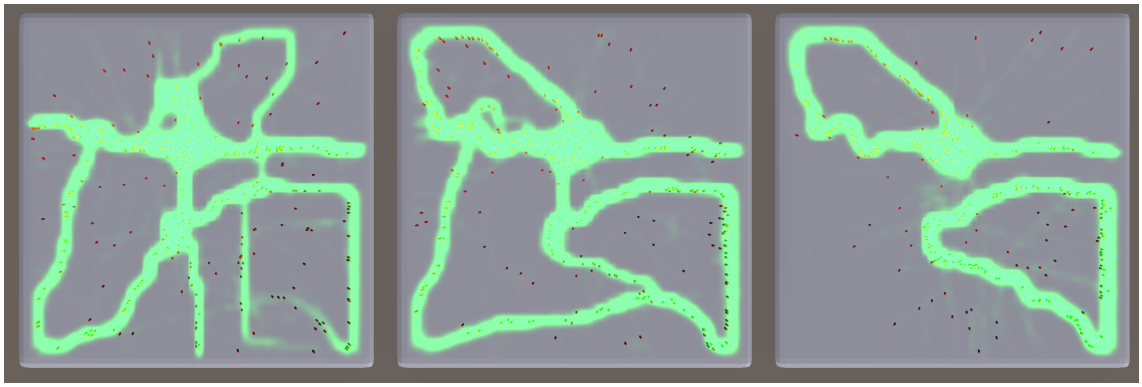


Figure 70. Shifting trails at T = 15000

Figure 71. Shifting trails at T = 18000

Figure 72. Shifting trails at T = 20000

3.2. Royale chamber construction

As described in Section 1.4, observation in bruinsma's thesis indicates the presence of a queen pheromone that stimulates termites to deposit material around the queen. The proposed mechanism is that the queen constantly emits the queen pheromone which triggers agents to build within a certain range of pheromone concentration. This was replicated in simulations by Ladley and Feltell was the emergence of the royal chamber as seen in Figures 9 and 13.

To replicate this in TermiteSim, agents have 2 states. Each state behaves differently. The first state is the gathering state. Here they look for material to pick up, and do so by executing these rules every simulation step:

1. The agents move forward.
2. The agent moves away from the queen pheromone, dependent on the queen pheromone's attractivity.
3. If the pheromone concentration is within the pickup range, then it will have a chance of picking up material, and going into the building state.

The second state is the building state, in which the agent searches for a spot to place material it picked up. The rules it executes each simulation step are:

1. The agents move forward.
2. The agent moves towards the queen pheromone, dependent on the queen pheromone's attractivity.
3. If the pheromone concentration is within the deposit range, then it will have a chance of depositing material underneath it, and going back into the gathering state.

This simulation was done using the parameters seen in Table 5. The progression of this simulation can be seen in Figure 73 to 81. Notable is that for this simulation the attractivity was set to 0, so the queen pheromone did not affect the movement of the agents. Effectively removing rule number 2 for both states.

Decay	0.000001
Diffusion	0.1
Deposit Amount	5
Attractivity	0
Antennae length	2
Minimum Deposit Threshold	0.05
Maximum Deposit Threshold	0.1
Minimum Pickup Threshold	0
Maximum Pickup Threshold	0.01
Deposit Chance	0.01
Pickup Chance	0.01

Table 5. Relevant parameters used for the simulation of royal chamber construction.

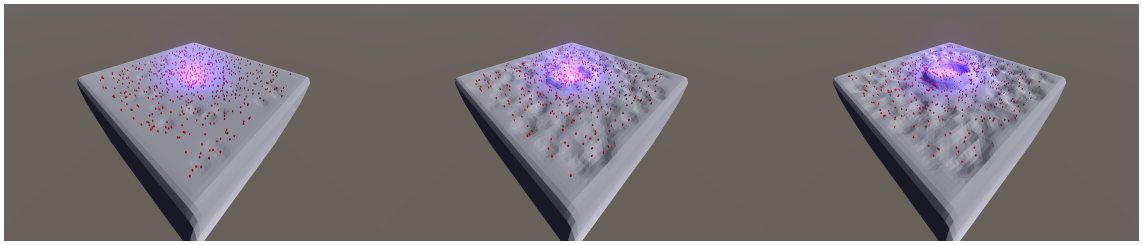


Figure 73. Royal chamber formation at T = 0. The queen pheromone has spread out to a smooth gradient. Construction has not yet started.

Figure 74. Royal chamber formation at T = 100. The initial construction of the royal chamber begins in the shape of a ring around the queen.

Figure 75. Royal chamber formation at T = 200. The ring increases in height.

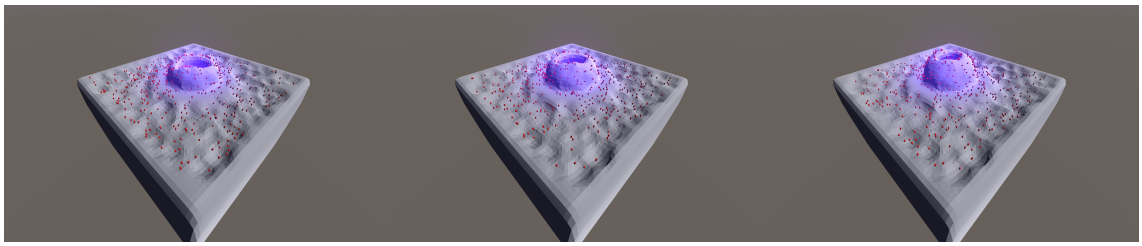


Figure 76. Royal chamber formation at T = 400. A clear, flat area around the ring shows the area where the agents neither deposit not pick up material.

Figure 77. Royal chamber formation at T = 600. The ring starts to be built inwards to close the opening in the top.

Figure 78. Royal chamber formation at T = 800. Building progress slows down as the area for deposit has decreased in size, and less agents stumble across it.

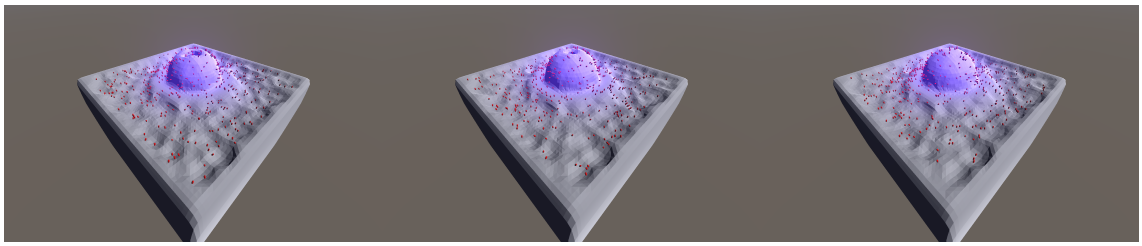


Figure 79. Royal chamber formation at T = 1000. Minor building progress

Figure 80. Royal chamber formation at T = 1200. The hole is almost sealed.

Figure 81. Royal chamber formation at T = 1500. The hole at the top has been sealed and the royal chamber is complete.

Another simulation was done with attractivity to the queen pheromone. This increased building efficiency and halved the number of simulation steps required to form a complete royal chamber. The result can be seen in Figure 82.

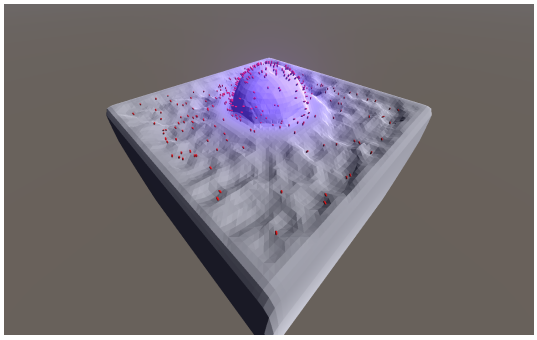


Figure 82. A completed royal chamber at T=800. Construction of the royal chamber with attractivity to the queen pheromone decreases the time it takes to construct a completed royal chamber.

Laminar air flow

To make a case for the theory that pheromones were the main mechanism behind the construction of the royal chamber, Bruinsma introduced a laminar air flow to disrupt the pheromones during construction. This caused the royal chambers shape to be elongated with the flow of air, proving that pheromones played a part in the construction. Ladley and Feltell also recreated this experiment in his simulation, the results of which can be seen in Figures 10 and 14. TermiteSim also supports the introduction of airflow, to affect the pheromones. This allows that experiment to also be recreated in TermiteSim. The result of that simulation can be seen in Figure 83, and a section view of the royal chamber can be seen in Figure 84. This same experiment was recreated with an upwards draft. This led to a vertically lengthened royal chamber, which can be seen in Figures 85 and 86.

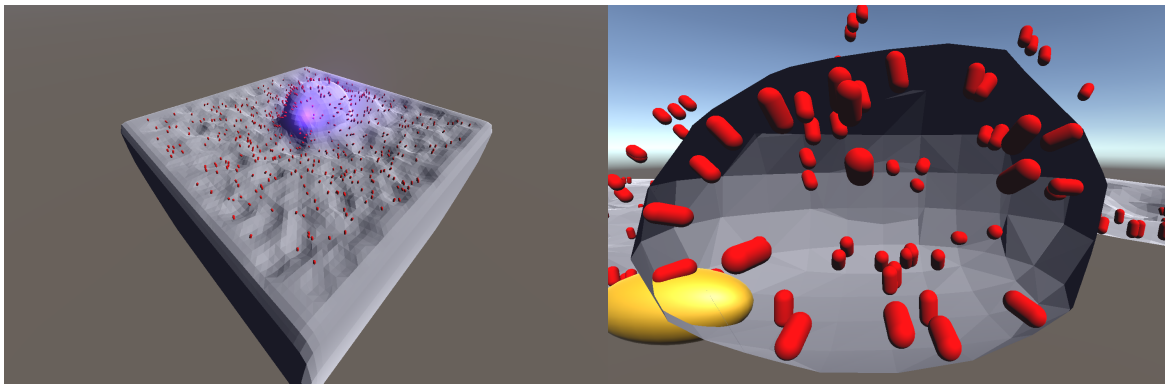


Figure 83. Royal chamber construction under the influence of lateral airflow. (Red are agents, purple is the queen pheromone)

Figure 84. A view inside the royal chamber constructed under the influence of lateral airflow. (Yellow is queen, red are agents)

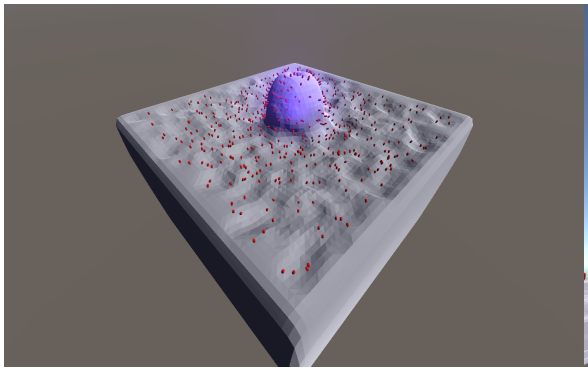


Figure 85. Royal chamber construction under the influence of an updraft. (Red are agents, purple is the queen pheromone)

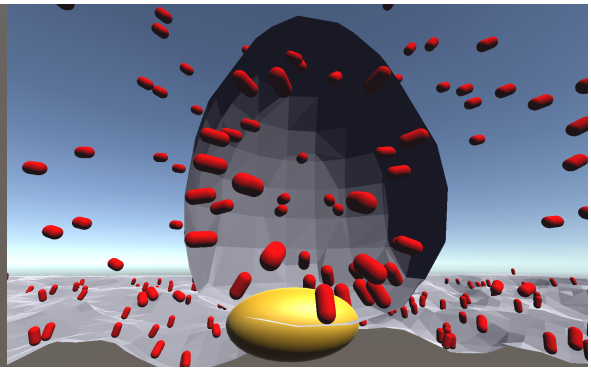


Figure 86. A view inside the royal chamber constructed under the influence of an updraft. (Yellow is queen, red are agents)

Following queen shape

Another observation Bruinsma made was that the construction of the royal chamber closely followed the shape of the queens body. Feltell was able to reproduce this in his paper as seen in Figure 15. This was also replicated in TermiteSim by placing multiple queens in an L shape and reducing their pheromone deposition, so that the total remains equal. Then 300 templating agents were added to the scene and build a royal chamber around the L-shaped queen as can be seen in Figures 87 to 89. As seen in Figure 90, the shape of the royal chamber does not follow the shape of the queens body closely. But it is affected by the queens shape, resulting in a somewhat triangular chamber. The shape of the resulting chamber can likely be adjusted by tuning the used parameters.

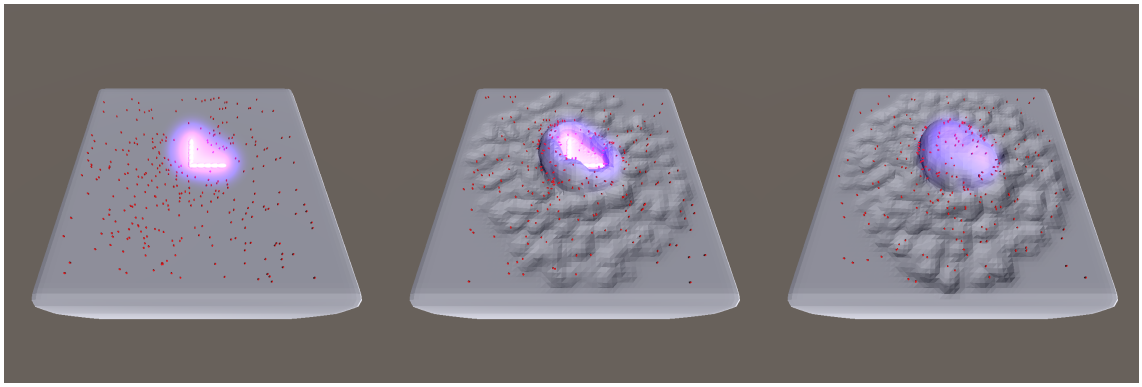


Figure 87. T = 1000. Showing the pheromone gradient, before the agents start building.

Figure 88. T = 1500. The agents deposit material around the queen, following her shape.

Figure 89. T = 2500. Completed royal chamber around an L-shaped queen.

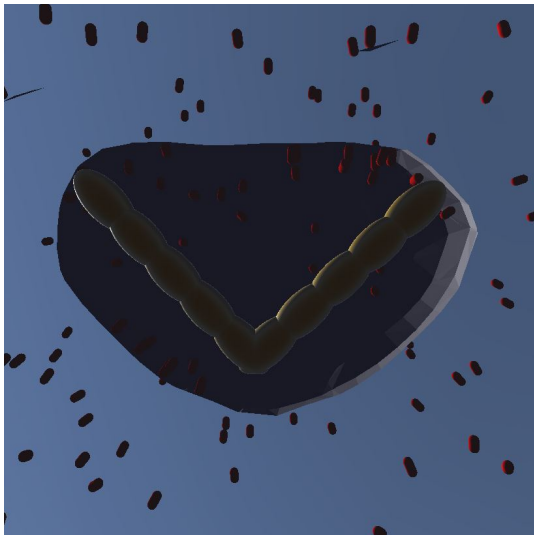


Figure 90. Inside the triangular royal chamber around the L-shaped queen. Showing that the royal chamber does not follow the queens shape closely, but is influenced by the queens shape.

3.3. Covered galleries construction

3.3.1. Covered galleries around a trail template

Another behaviour that was observed is the deposition of material at the edges of strong trails, which over time leads to the creation of covered galleries. Ladley simulated this by introducing a set of trail agents that create a single trail from one side of the simulation to the other and added builder termites that build along the edges of the trail seen in Figure 11 and 16 for Feltells recreation. Feltell then did another simulation with two crossing trails as seen in Figure 17. These simulations are described in more detail in Section 1.4.5. These simulations were also recreated in TermiteSim. However, currently, TermiteSim only supports one type of agent. Therefore, the trail in the following simulations is placed as a template, instead of dynamically placed by moving agents. To place this template trail, the stationary queens from royal chamber construction are placed in the shape of the trail and repurposed to continuously place trail pheromone instead of queen pheromone. The agents in the simulations adhere to the following rules:

1. Agents move forward.

2. Agents turn towards the highest trail pheromone concentration they sense with their antennae.

3a. If the agent is in the gathering state: If the pheromone concentration is within the pickup range, then it will have a chance of picking up material, and going into the building state.

3b. If the agent is in the building state: If the pheromone concentration is within the deposit range, then it will have a chance to deposit material, and going into the gathering state.

Using the parameters in Table 6, the first simulation can be seen in Figures 91 to 93. For the first 1000 simulation steps, the agents aren't allowed to build to allow the pheromone gradient to reach a stable state. The construction progresses faster in the centre of the covered gallery, as the density of agents is higher there due to the agents returning to the centre after reaching the edge of the terrain. After 10000 steps, the construction of the covered gallery is completed.

Decay	0.0001
Diffusion	0.2
Deposit Amount	0.1
Attractivity	150
Antennae length	2
Minimum Deposit Threshold	0.05
Maximum Deposit Threshold	0.15
Minimum Pickup Threshold	0
Maximum Pickup Threshold	0.04
Deposit Chance	0.01
Pickup Chance	0.01

Table 6. Relevant parameters used for the simulation of covered galleries construction around a template trail.

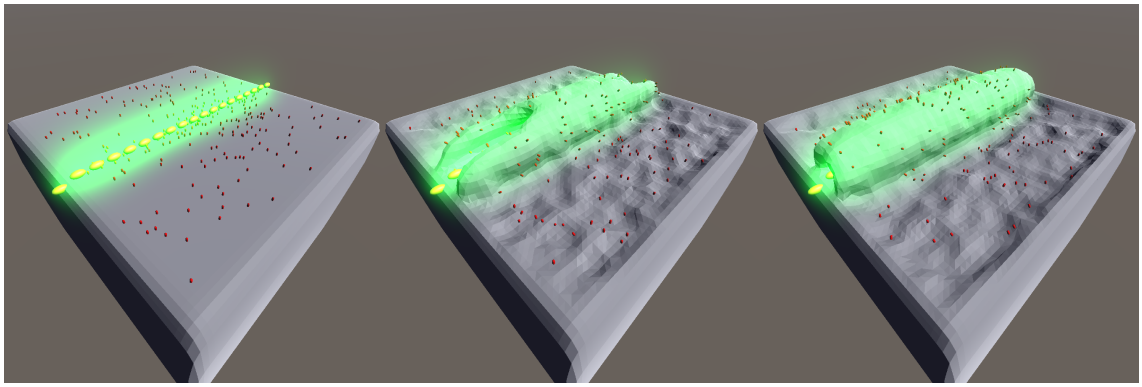


Figure 91. Covered gallery at T = 1000. First allowing the pheromones to reach a stable state before allowing the agents to build.

Figure 92. Covered gallery at T = 5000. The agents are constructing covered gallery. The centre is more developed as the density of agents there is higher.

Figure 93. Covered gallery at T = 10000. The finished covered gallery.

Using the same parameters, another trail was added perpendicular to the first trail. The simulation can be seen in Figures 94 to 96. The construction progresses significantly slower. This is likely because the area where agents can pick op material has reduced, and more material is needed as there are more walls to build. However, even after 100000 steps the construction is not finished. There is a small chance that agents get stuck at the edge of the terrain. Usually this is is insignificant, but for extremely long simulations, such as this one, it can cause many agent to get stuck over time. In this simulation, at T = 100000, 95% of the agents got stuck, which caused building progress to almost completely stop.

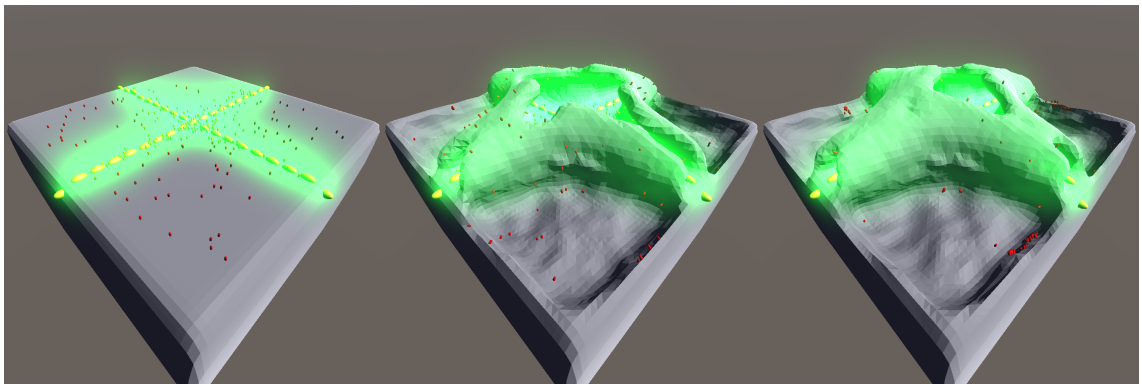


Figure 94. Covered gallery intersection at T = 1000. Showing the pheromone gradient, before the agents start building.

Figure 95. Covered gallery intersection at T = 45000. Building slows down as agents start to get stuck and cant find material to pick up.

Figure 96. Covered gallery intersection at T = 100000. Building has almost completely come to a halt as many agents are stuck.

Another simulation was done, where the parameters were adjusted to those in Table 7. This was done to decrease the amount of material required to build the covered galleries, by decreasing the width of the trails and decreasing the amount of material agents pick up and place as can be seen in Figures 97 to 99. However, due to the decreased size of the galleries, the agents get stuck inside the narrow ends of the tunnels before construction could be completed.

Decay	0.0003
Deposit Chance	0.1
Deposit radius	2
Deposit weight	5

Table 7. Relevant parameters changed for the simulation of covered galleries construction around two crossing template trails.

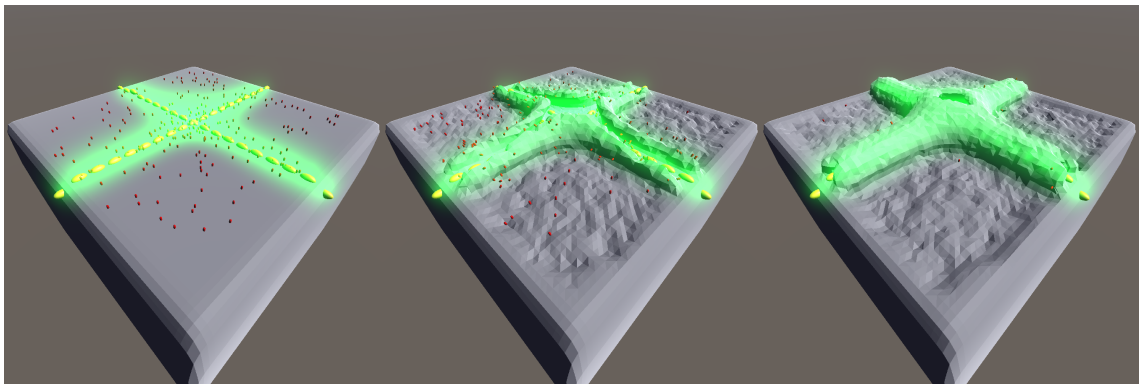


Figure 97. Covered gallery intersection at T = 1000. Showing the pheromone gradient, before the agents start building.

Figure 98. Covered gallery intersection at T = 8000. Showing thinner and slimmer walls forming around the trail.

Figure 99. Covered gallery intersection at T = 20000. The construction comes to a halt, as agents get stuck inside the narrow tunnel.

The parameters were adjusted once more to the values in Table 8, to make the galleries wider, while maintaining thin walls. This time the construction of the covered galleries around the intersecting trails was successful as seen in Figures 100 to 102.

Decay	0.0003
Diffusion	0.3
Deposit Amount	0.2
Attractivity	50

Table 8. Relevant parameters changed for the simulation of covered galleries construction around a template trail.

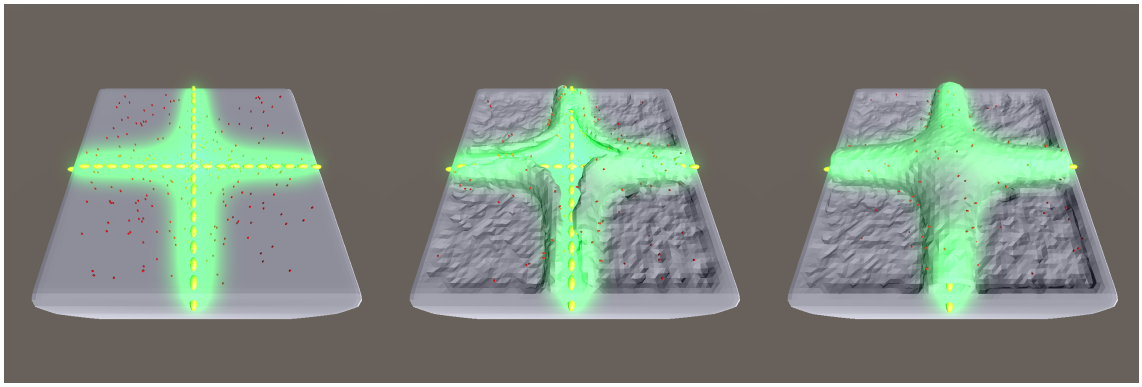


Figure 100. Covered gallery intersection at T = 1000. Showing the pheromone gradient, before the agents start building.

Figure 101. Covered gallery intersection at T = 10000. The galleries walls are thinner, but the width is similar to the single trail simulation.

Figure 102. Covered gallery intersection at T = 45000. The completed intersecting galleries.

3.3.2. Covered galleries around dynamic trails

As demonstrated in Section 3.1, agents are able to dynamically form trails. By combining the trial forming control strategy with the template builder control strategy, a new type of agent is created that both lays trails and and build ad the edges of those trails. This agent is called the Gallery builder agent. The rules it follows are:

1. Agents move forward.

2. Agents turn towards the highest trail pheromone concentration they sense with their antennae.

3. Agents place trail pheromone at their location.

4a. If the agent is in the gathering state: If the pheromone concentration is within the pickup range, then it will have a chance of picking up material, and going into the building state.

4b. If the agent is in the building state: If the pheromone concentration is within the deposit range, then it will have a chance to deposit material, and going into the gathering state.

The following simulation was done with the parameters seen in Table 9, and a swam of 300 gallery builder agents.

The desired behaviour is for agents to only pick up material once the a trail has consolidated. Then to pick up material if the trail pheromone concentration is high, and place it at the edge of the trail. This should cause some trails to be established before construction begins. During the simulation seen in Figures 102 to 105, trails are are formed but as soon as those trails start to consolidate they are blocked of by material being placed on them. This prevents the trail from being followed, and causes it to become neglected after which it decays and disappears.

One explanation of the trail blocking behaviour is that it occurs because as trails consolidate, their pheromone concentration increases. And as it increases, it has to first pass through the concentration range which stimulates deposition, which causes the agents that were traveling the trail to deposit on top of it. This kills the trail before it can surpass the pheromone concentration range that stimulates deposition.

Decay	0.001
Diffusion	0.005
Deposit Amount	0.02
Attractivity	20
Antennae length	2
Minimum Deposit Threshold	0.1
Maximum Deposit Threshold	0.5
Minimum Pickup Threshold	0.5
Maximum Pickup Threshold	10
Deposit Chance	0.01
Pickup Chance	0.01

Table 9. Relevant parameters used for the simulation of covered galleries construction around dynamic trails.

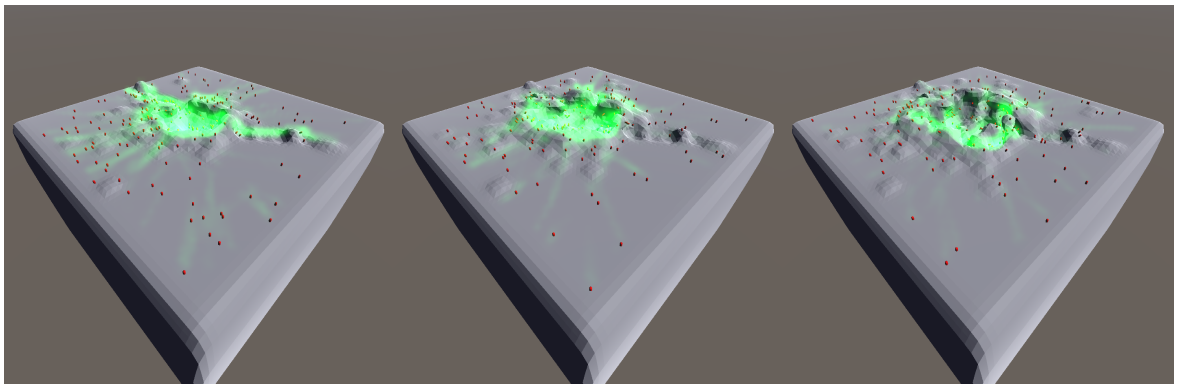


Figure 103. Covered gallery around dynamic trails at T = 1000. Formed trails are being blocked off. Only the centre trail is constant.

Figure 104. Covered gallery around dynamic trails at T = 1500. Blocked off trails, are neglected and disappear.

Figure 105. Covered gallery around dynamic trails at T = 5000. No progress in building covered galleries, only a crater around the centre.

Multiple changes to the parameters were made in an attempt to get covered galleries, but none were successful. Some of these attempts can be seen in Figures 106 to 108. Perhaps there is a set of parameters that results in the desired behaviour, but could not be found at this time. One factor that might potentially play a role in thee inability to construct covered galleries around dynamic trails is the tendency of trails to change over time, first described in Section 3.1, later elaborated in Section 4.1. As the trails change shape, the construction around the trails doesn't which causes misalignment. This can be compared to construction following a blueprint, and then subtly changing the measurements on the blueprint.

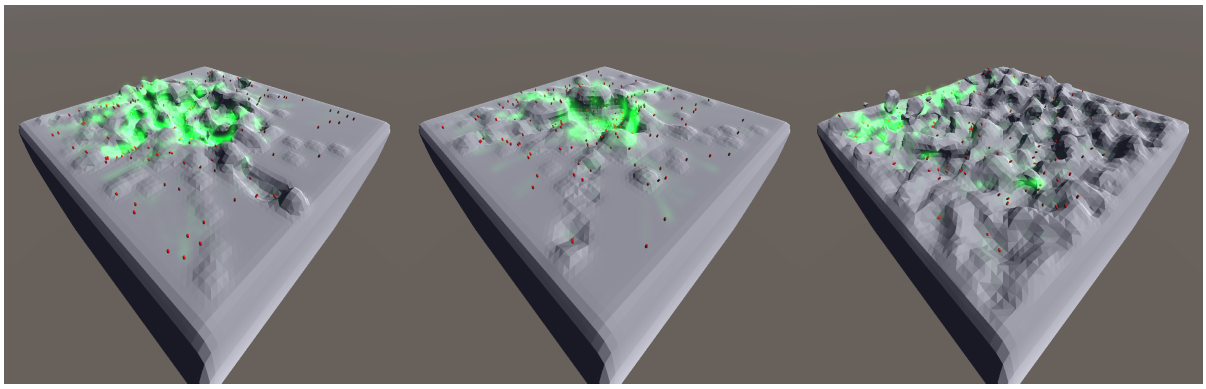


Figure 106. Covered gallery around dynamic trails at T = 5000, diffusion = 0.01. Increasing the diffusion in an attempt to create less, but wider and stronger trails.

Figure 107. Covered gallery around dynamic trails at T= 5000. deposit chance = 0.001. Decreasing the deposit chance to reduce the chance of a trail being blocked before consolidation.

Figure 108. Covered gallery around dynamic trails at T=10000. Adjusted pickup and deposit thresholds. Lowered the deposit range to 0.001 - 0.1, and lifted pickup restrictions.

3.4. Pillar building

One of the first behavior described by Bruinsma is aggregated deposition by the cement pheromone as described in Section 1.4.2. At its core it just causes the agents to deposit material on top of the cement pheromone, which is secreted when they deposit material, causing a positive feedback loop of them piling soil pellets on top of soil pellets. In the Bruinsma's observation he describes a emergence of equally spaced pillars. There are no other simulations that reproduce this behavior.

In order to reproduce this behaviour, the pillar builder agent type is created. Pillar builders have 2 states. The first state is the gathering state. Here they look for material to pick up, and do so by executing these rules every simulation step:

1. The agents move forward.
2. The agent moves away from the cement pheromone, dependent on the cement pheromone's attractivity.
3. If the cement pheromone concentration is within the pickup range, then it will have a chance of picking up material, and going into the building state.

The second state is the building state, in which the agent searches for a spot to place material it picked up. The rules it executes each simulation step are:

1. The agents move forward.
2. The agent moves towards the cement pheromone, dependent on the cement pheromone's attractivity.
3. If the cement pheromone concentration is within the deposit range, then it will have a chance of depositing material underneath it, and going back into the gathering state.
4. If there is no cement pheromone at the agents location, it has a small chance to deposit material, creating a new deposition zone.

The idea is to have a deposition zone with a large pheromone gradient around it. This large pheromone gradient has 2 functions, attracting agents from a large distance, and preventing the creating of new deposition zones within the gradient. Multiple simulations where done to attempt to create equally spaced pillars. However, none where successful. In order to get a large pheromone gradient, the diffusion rate has to be high. But a high diffusion rate would also cause the pheromones of a deposition to dissipate before it could turn into a deposition zone. Reducing the pheromone diffusion resulted in some pillars being formed, but those merged together over time as seen in Figures 109 to 111. A potential cause is that as the pheromone concentration increases, the area of the deposition zone that has the minimum required concentration for deposition becomes larger, thus expanding the deposition zone. This causes agents to also deposit at the edge of that zone, further increasing its size. If this is true, then lowering the deposition chance should cause deposition zones to merge sooner, and lowering the deposit chance should cause deposition zones to become pillars. The result of those changes lead to the simulations seen in Figure 112 and 113. Although Figure 113 does show a pillar, it is only one, and not multiple equally spaced pillars. It would be interesting to see what would happen if those settings were used on a larger world size. Perhaps multiple of those pillars would arise.

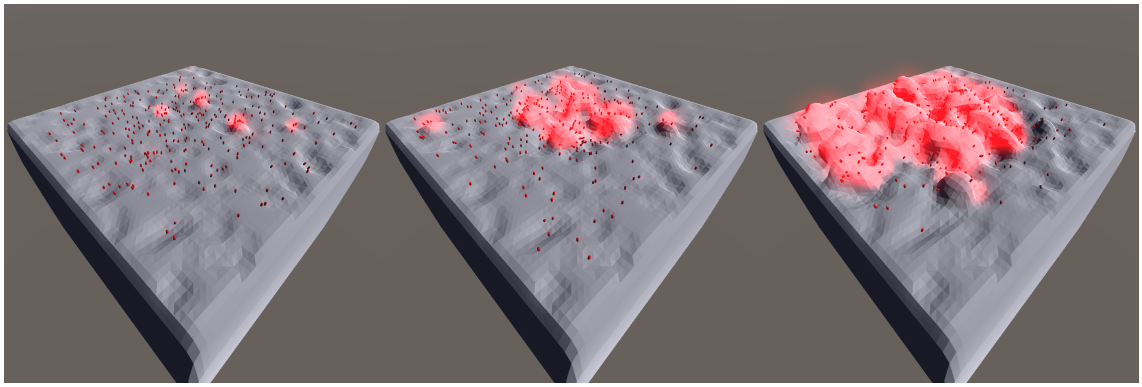


Figure 109. Attempt at pillar formation at T = 400. Initial deposition zones appear.

Figure 110. Attempt at pillar formation at T = 800. Separate pillars form.

Figure 111. Attempt at pillar formation at T = 2400. All pillars have merged together.

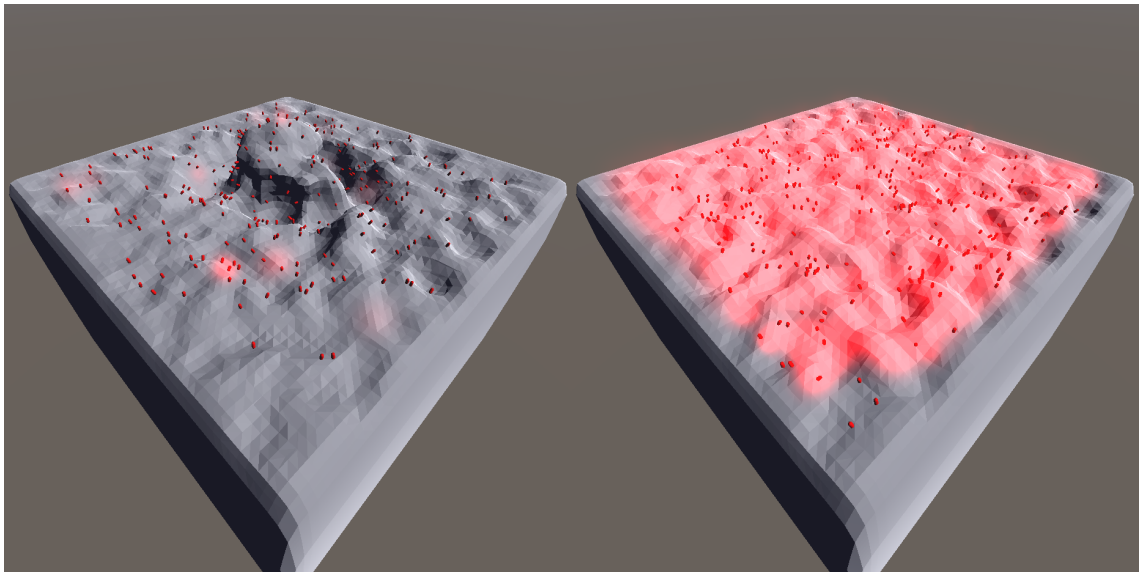


Figure 112. Simgular pillar formation at T = 25000. Decreased deposition chance leads a singular pillar in the centre of the world.

Figure 113. Increased deposition pillar formation at T = 1000. Increased deposition chance leads to rapid expansion of deposition zones.

There might be a set of parameters for this control strategy that results in the desired behaviour of pillar building, but it was not found. However, as described in Section 1.4.2, trail pheromone may inhibit deposition. Perhaps what supports the emergence of equally spaced pillars is not a large gradient of cement pheromone, but the trails between the deposition zones that prevent the deposition zones from expanding vertically. Therefore, a new control strategy is created that causes agents to also lay trail pheromone, which inhibits deposition. Multiple variations on parameters were run with this control strategy of which one result can be seen in Figure 114. Although the trails reduced the merging of deposition zones, it does not prevent it and over time, the formed pillars still merge.

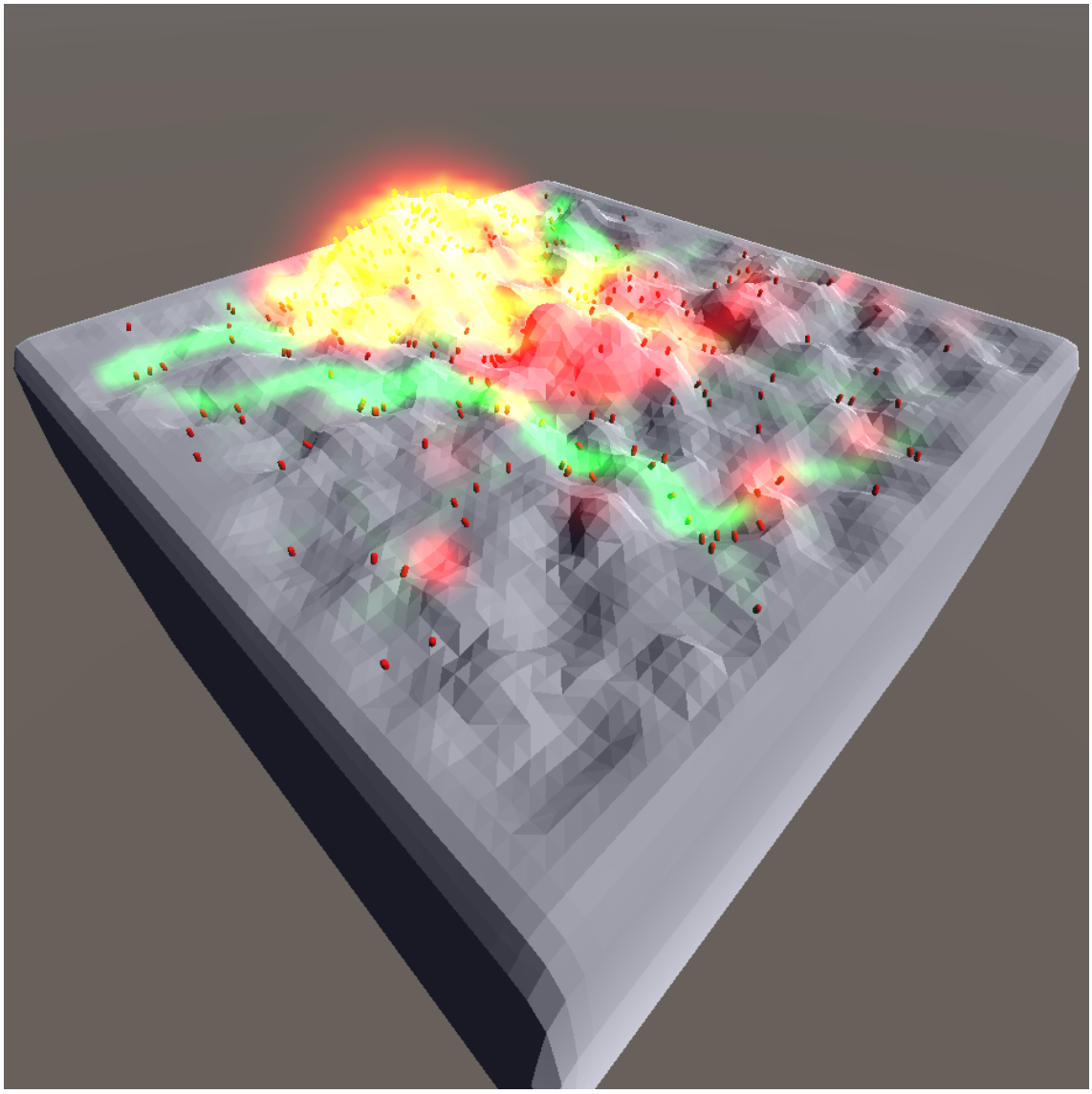


Figure 114. Attempt at pillar formation with dynamic trails at T = 3000. Pillar building with trail pheromone that inhibits deposition. Does not completely prevent merging of pillars, but does reduce it.

3.5. Mound Building

The last behaviour that was replicated from previous simulations is the construction of the royal in the presence of trail pheromones as seen in Figures 12 and 18. This was done by using the Mound builder control strategy in TermiteSim. This control strategy is made to experiment with the behaviour that occurs when combining all previous control strategies. Agents following this control strategy will be affected by the queen, cement and trail pheromone. The effects can be configured to alter the deposit and pickup chances, and can also limit the agents to only perform those actions in a certain range of pheromone concentration. These affects can be configured individually for each pheromone type. For these simulations the pickup and deposit chances were 0.01 and 1 respectively and not affected by pheromones. This means that the builders have a 1% chance to pick up new material and a 100% to place it down if the conditions allow so. By setting the minimum and maximum thresholds to 0 and 10 respectively, there will be no restrictions on that behaviour.

Pheromone	Queen	Trail
Decay	1E-5	0.002
Diffusion	1	0
Deposit Amount	5	0.02
Attractivity	50	20
Minimum Deposit Threshold	0.08	0
Maximum Deposit Threshold	0.1	0.01
Minimum Pickup Threshold	0	0
Maximum Pickup Threshold	10	10

Table 10. Relevant parameters used for the simulation of royal chamber construction in the presence of dynamic trails.

First a simulation was done that combines dynamic trail forming with royal chamber construction with the parameters set as shown in Table 10. Here the trail inhibits deposition by only allowing agents to deposit material if the trail pheromone concentration is between 0 and 0.01. The results of this simulation can be seen in Figures 115 to 118, and a closer look at the final structure can be seen in Figure 118. In the first 1000 simulation steps, agents are prevented from picking up material to allow the pheromone trails and the queen pheromone template to reach a stable state. In Figure 116 can be seen that equally spaced pillars appear that are increased in height and connected in Figure 117.

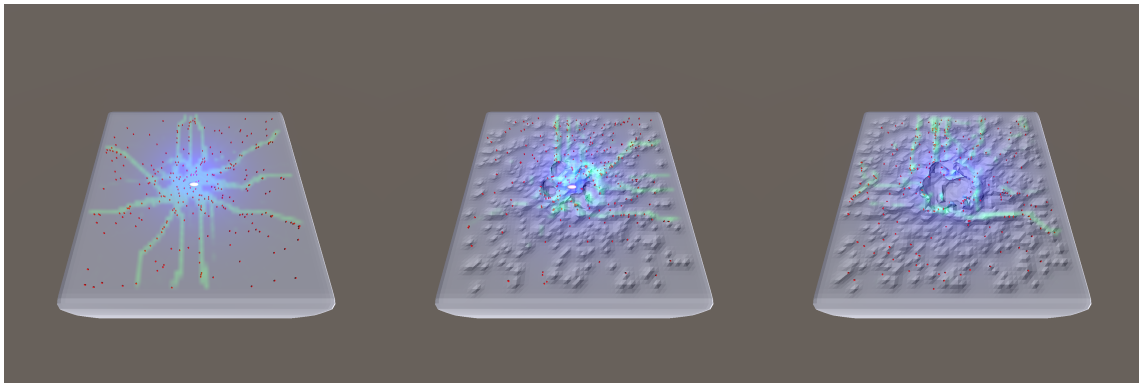


Figure 115. Royal chamber with trails at T = 1000. Showing the pheromone gradients, before the agents start building.

Figure 116. Royal chamber with trails at T = 4000. Showing the construction of equally spaced pillars around the queen.

Figure 117. Royal chamber with trails at T = 15000. Showing the complete royal chamber with entrances at ground level.

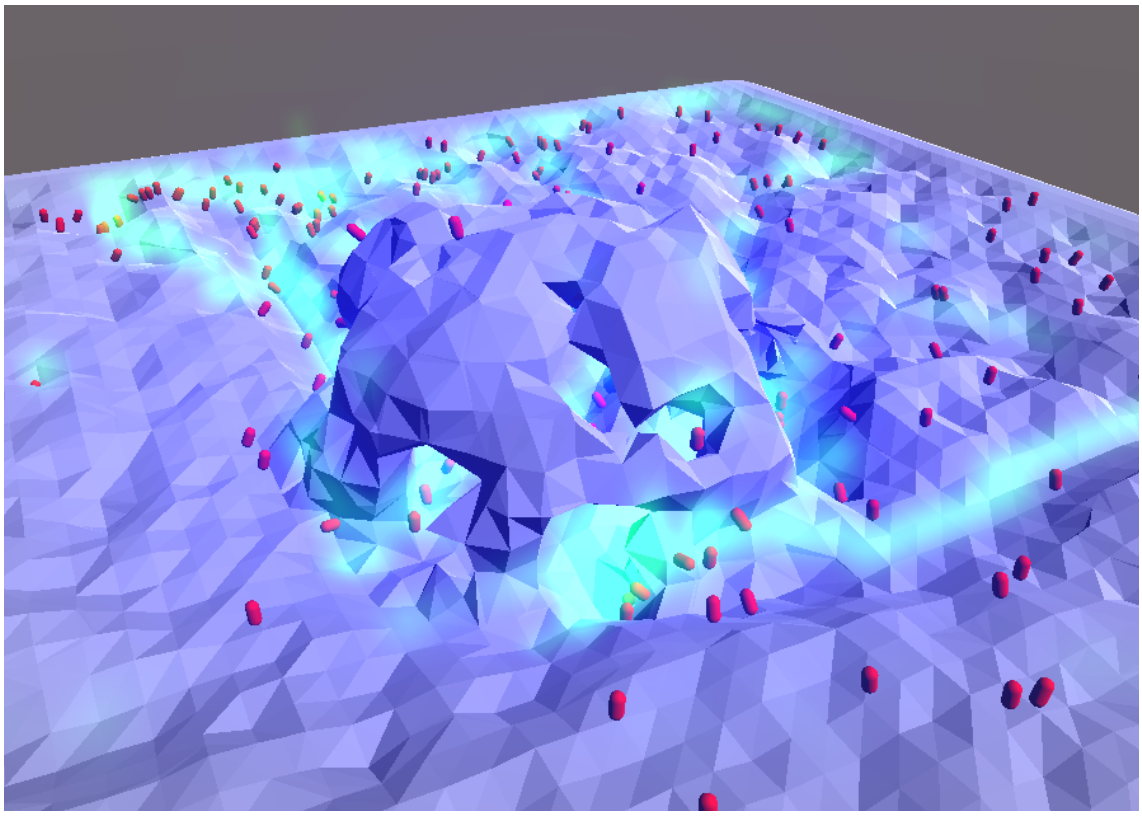


Figure 118. Construction of the royal chamber in the presence of trail pheromone. Showing the creation of entrances at the location of where trails intersect with the royal chamber.

Note that the royal chambers roof remained unfinished. This is likely due to the fact that most agents stay close to the trails, which were not formed on top of the chamber. This lead to very few agents reaching the top of the roof, lowering the frequency of construction on the roof.

In an attempt to increase building efficiency, the cement pheromone was added with the parameters presented in Table 11. The effect of the cement pheromone was only to attract agents that carry material to place, by increasing its attractivity. The cement pheromone did not influence the chance for deposition as this change was already 100% as stated above. This lead to the simulation seen in Figures 119 to 122. The resulting structure showed that only 1 individual pillar emerged, and instead

walls were constructed. In Figure 122 can be seen that these walls were later turned into pillars by the agents removing material after the walls reached a certain height. These pillars were thinner than the resulting structure from the simulation without cement pheromone.

Pheromone	Cement
Decay	1E-5
Diffusion	0.01
Deposit Amount	1
Attractivity	100
Minimum Deposit Threshold	0
Maximum Deposit Threshold	10
Minimum Pickup Threshold	0
Maximum Pickup Threshold	10

Table 11. Relevant parameters of cement pheromone used for the simulation of royal chamber construction around dynamic trails and cement pheromone.

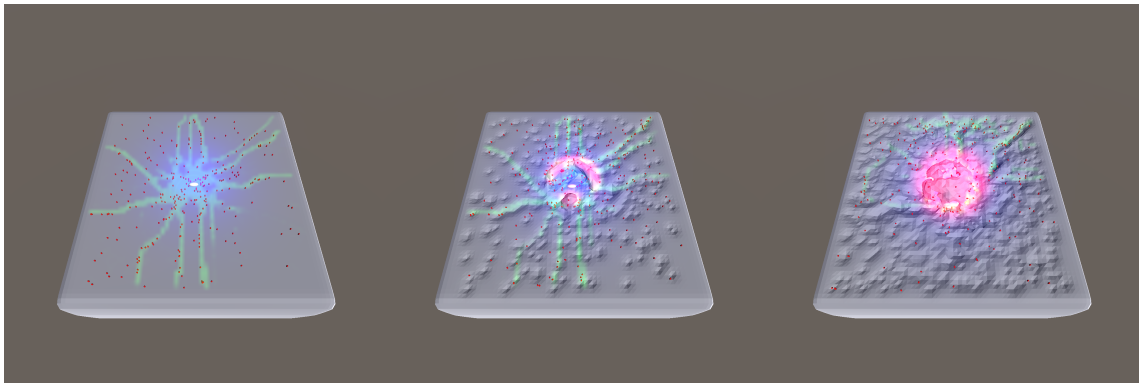


Figure 119. Royal chamber with trails and cement pheromone at T = 1000. Showing the pheromone gradients, before the agents start building.

Figure 120. Royal chamber with trails and cement pheromone at T = 1600. Showing the creation of a large wall and 1 pillar around the queen.

Figure 121. Royal chamber with trails and cement pheromone at T = 10000. Showing the completed royal chamber with entrances at ground level.

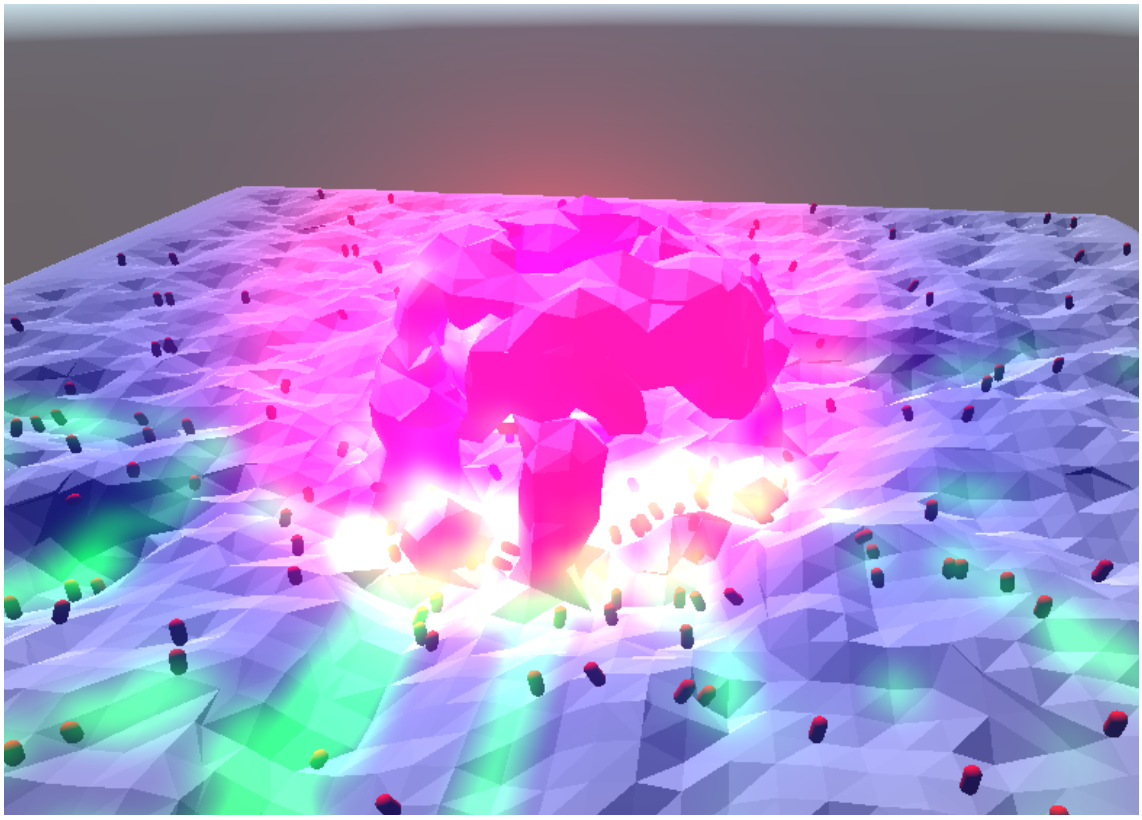


Figure 122. Construction of the royal chamber in the presence of trail pheromone and cement pheromone. Showing the side from which walls were initially constructed that now have been turned into pillars with wide gaps for trails.

3.6. Non termite behaviour

As a proof of concept, a simple example of non-termite behaviour is also replicated in TermiteSim. This is a simulation of 20 cleaner bots spreading a cleaning substance over a flat area. To model this cleaning substance, trail pheromone with no decay nor diffusion is used. The relevant parameters used for this simulation can be seen in Table 12. The cleaner bot agents follow 3 rules:

1. Agents move forward.
2. Agents turn away from the cleaning substance.
3. Agents place cleaning substance at their location.

Decay	0
Diffusion	0
Deposit Amount	0.2
Attractivity	-500
Number of agents	20

Table 12. Relevant parameters for the simulation of cleaner robots.

As can be seen in Figures 123 to 125, the bots first move straight forward, causing the area to be sectioned into blocks. They then start filling these areas until they are surrounded by cleaning substance, at which point they cross over to a new section. This repeats until the entire area is filled.

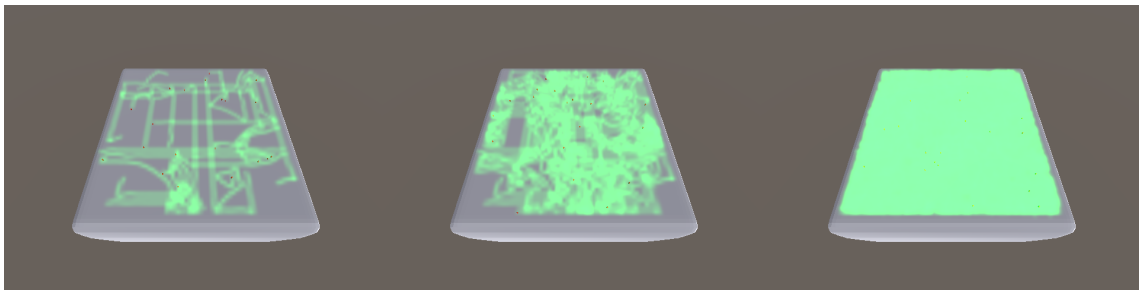


Figure 123. Cleaner bots at T = 150. Cleaner bots section the area.

Figure 124. Cleaner bots at T = 450. The sections are filled.

Figure 125. Cleaner bots at T = 1500. The entire area is covered equally.

Another simulation was done where the agents move randomly, without being affected by the cleaning substance, to compare the efficiency for covering the area which can be seen in figures 126 to 128.

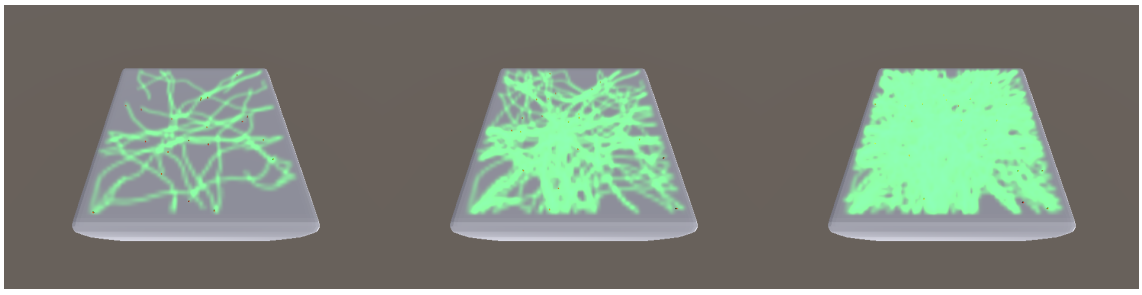


Figure 126. Random bots at T = 150. Agents moving randomly.

Figure 127. Random bots at T = 450. Some areas are frequented, while many remain untouched.

Figure 128. Random bots at T = 1500. Many areas are not yet cleaned.

3.7. Conclusion

As demonstrated in this chapter, TermiteSim is able to replicate similar constructions as shown in previous simulations. This proves that the required capabilities for further research of termite-like building behaviour through simulations have been met as described in Section 1.6. TermiteSim also demonstrates it can simulate the formation of trails and change the characteristics of those trails by tuning the parameters of the simulation. Lastly, TermitesSim creates a simple proof of concept for simulating non-termite-like behaviour within TermiteSim. Demonstrating that TermiteSim can be used outside the scope of termite research, in order to design the behaviour of robot swarms.

4. Discussion

We looked at the potential of swarm robotics in the construction industry, and identified that more research needs to be done (Chapter 1). For this research a simulator was needed, and a list of requirements for such a simulator was determined (Section 1.6). This was list compared to currently available simulators, but it was found that none of them suffice (Section 1.7). Therefore, this project created a new simulator, named TermiteSim, following the requirements (Chapter 2). This functionality is then demonstrated by using TermiteSim to recreate the observed behaviours described in the literature in Section 1.4.

As shown in section 3.3.1, small changes to the parameters can drastically change the resulting behaviour. In this these, most of the simulation presented were the ones that succeeded in demonstrating coherent behaviour. But many of the parameter settings tried during the project did not result in the desired behaviour. For instance, the balance between decay and pheromone deposition is very fine. To little decay and the entire terrain gets covered, too much decay and pheromones disappear before they can affect the swarms behaviour. However, the fine balance between parameters might also be due to the control strategy that utilizes those parameters. It would be interesting to compare the robustness of range threshold based control strategies as implemented in Chapter 3, to response-threshold function based control strategies.

Interesting to note is the explanation for smoother trail forming due to increased antennae length, as illustrated in Figure 60. This proposes that due to the antennae implementation, agents will always cut corners when they are attracted to a pheromone. If that is indeed the case, then this might be an explanation for the shifting trails described in section 3.1.1 and shown in Figures 64 to 72. As stated before, these shifting trails might have a large impact on emergent behaviour, and be the reason that no covered galleries could be constructed around the dynamically formed trails as described in Section 3.3.2. This might also influence agent behaviour in other was that are not yet observed.

The second behaviour that was demonstrated is the construction of the royal chamber (Section 3.2). This behaviour was successfully replicated the structures from previous simulations (Section 1.4.5).

As described in Section 1.4.2, the queen pheromone also stimulates the agents to pick up material. This was not included in the simulations presented in Chapter 3, but was briefly experimented with. Bruinsma (1979) describes that the queen is slowly lowered due to this effect, but in the simulations this lead to a rapid descent that created a large hole. In one case this would even lead to the queen not descending because the pickup chance was so high that agents would pick up material just within the walls of the royal chamber. This resulted in a sort of island in the centre of the royal chamber on which the queen was seated. This could also be described as a throne or a pedestal. This is also one of the behaviours that would be interesting to research further.

The third behaviour was the construction of covered galleries (Section 3.3). In the first set of simulations the construction was created around a stationary pheromone trail. The balance of the parameters for this simulation much less forgiving as a small change could prevent the structure from being completed. The larges cause of this was agents getting stuck in various geometries of the terrain. This highlighted some limitations of the simulator as a result of bugs. If these bugs are fixed, then the simulation could affectively run much longer, allowing for much larger structures to be built. Then simulations were done in an attempt to recreate those same covered galleries construction around dynamic trails placed by agents. This proved to be more difficult than recreating the previous behaviours. This could be due to the fragile balance between the parameters, or the shifting trails discussed above. It would be interesting to further attempt to replicate covered galleries around dynamic trails, as this would potentially allow the mound created in Section 3.5 to extend past just the royal chamber, which in turn could lead to new emergent structures.

In Section 3.6 A very simple proof of concept of a cleaner robot swarm is simulated in order to demonstrate that TermiteSim can simulate outside the scope of just termite behaviour. However, these simulations are done on a flat surface, where a the dynamic approach that is simulated is not optimal. Where the value of this dynamic approach really lies is in uneven geometries. It would be interesting to perform that simulation again on uneven terrain. This should be possible by simply adjusting the terrain settings and running the vacuum bot control strategy.

In the simulation that were done, all of them stayed relatively close to the surface of the terrain. There is some literature described in Section 1.4 that suggest the possibility of termites creating excavation sites. It would be very interesting for a whole new control strategy to be created that causes these excavation sites to emerge, and see what emergent behaviours these cause. Both isolated and in combination with other pheromones as this could lead to the emergence of underground structures.

To conclude, these simulations show that there are many different balances of parameters and control strategies that can lead to termite mound-like structures. And that there is a balance in the parameters that must be found through iteration. Though this balance can be found if given enough time to iterate. It is difficult to conclude wether a control strategy does not follow the behaviour of termites, or that the balance of parameters is very delicate for it to succeed. And that even if a desired behaviour is observed, the control strategy does not necessarily reflect the underlying mechanisms that termites use. That being said, it is incredibly interesting to experiment with the emergent behaviour and structures by just changing the parameters. Although many hours of simulations have been simulated using TermiteSim there is a sense that it has been only the surface of what could be done with it.

4.1. Limitations of the evaluation

Despite offering the required functionality, there are still several limitations to TermiteSim.

Feltell demonstrated the use of a response-threshold function instead of the window range method applied by Ladley as described in Section 1.4.5. The control strategies demonstrated in TermiteSim in Section 3.2 Royal chamber construction and 3.3 Covered galleries construction use a window range, and Section 3.4 Mound construction uses a linear relation between drop chance and pheromone concentration. Which is a different relation between pheromone concentration and pickup and deposit chances than the literature suggests. This possibly lead to different results as the parameters for pheromone diffusion could not be compared to those of Feltell.

One decision that was made was to have agents turn back towards the centre when they reach the edge of the terrain as described in Section 2.4.2. This was done to prevent agents from getting stuck near the edges. However, in Section 3.3.1 this still lead to agents getting stuck near the edge and preventing the simulation from completing the structure. This chance is very small, but when running long simulations (e.g. 100.000 simulation steps) a termite swarm of 300 can have 250 agents getting stuck, reducing construction efficiency at the later stages of the simulation. Thus effectively limiting maximum duration of a simulation done in TermiteSim. Besides this, this implementation also causes the density of agents to be artificially inflated in the centre of the terrain if there is a lack of attractive pheromones. Although this is a limitation, it was used in Section 3.5 to get the trails to form around the queen. By increasing the trail pheromone's attractivity, the trails would not intersect with the construction of the royal chamber, and thus not leading to the spaced pillars, nor the entrances to the royal chamber. Without utilizing the high density of agents in the centre due to the terrain edge implementation, none of the attempts to create entrances to the royal gallery were successful. This could be an indication that the balance for this behaviour was to intricate to find, and could mean that the control strategy that was used not representative of the process termites use to create and maintain entrances to the royal chamber.

Besides pheromones the tactile feedback of spacial heterogeneity and air humidity are also likely a stimulus for deposition as described in Section 1.4.3 Other Stimuli. Such stimuli are currently not included in TermiteSim. The implementation of these stimuli in the simulations could have a major impact on the emergent behaviour.

In the simulation in Chapter 3, the swarms consist of only one type of agent. Other simulators use separate termite types such as: Foraging termites, nursing termites, and builder termites which all have their own behaviour. Instead, TermiteSim's agents have multiple states they can change between. The composition of the swarm, with different types of agents might result in the emergence of different structures. For instance, in Section 3.5, the cement pheromone was added which attracted the agents that were carrying material to place. The goal of this was to guide agents to the deposition areas to increase building efficiency. However, these agents also lay trails. This meant that the larger influx of agents would cause more deposition inhibiting trails to be placed on the deposition zone. This in turn stoped any agent from repositioning there. This caused the opposite of the intended effect, completely preventing construction.

Contrary to real world observation, pheromones in TermiteSim only have a gas state. For instance, the cement pheromone of actual termites is mixed with the soil as a liquid. This liquid pheromone than evaporates over time. This causes pheromones to persist for longer durations of time, despite a high diffusion rate. In TermiteSim's pheromone implementation, either the diffusion rate is high to increase the pheromone gradient to attract more workers to the deposition site, or the diffusion the is lowered to extend the time it takes before a deposition site is abandoned. But a trade of between pheromone gradient size and lifetime is still present. Preferably, both would be independent of each other, as the current balance for pillar building is too fine where no effective set of parameters has been found yet. Pheromones lacking a liquid state potentially also affects trail forming, as emergent trails are shown to not maintain stationary over longer simulations. This in turn is detrimental for the construction of covered galleries around those emergent trails as shown in Section 3.3.2.

Lastly, despite the increase in realism compared to previous simulations with the space being continuous and the terrain allowing for the construction of organic shapes. The simulation is still a simplification of real termites and the control strategies have yet to be tested on a physical robot swarm.

4.2. Further development

All of the limitations named in the previous section can be overcome with further development of the simulation. However, there are also alternative design decisions which were not limiting to the scope of the project, but could add value for future work in simulations.

Gravity in terrain

Currently, material can only be placed on the surface of the terrain. This prevents placing material in midair. Similarly removing material can also only be done at the surface of the terrain. However, there is no check for wether removing material results in parts of the terrain being unsupported. Thus leading to floating chunks of terrain in some cases. Although rare, this is an unrealistic occurrence that should preferably be prevented.

Halting

Studies have shown that overcrowded areas result in halting, where termites slow down, like in a traffic jam. Currently, collisions are prevented by the agents turning away from each other, while maintaining a constant movement speed. This results in more turning in case of overcrowding. An option could be added to implement halting instead of turning to avoid collisions.

Drunkards walk

Research shows that termites tend to do a drunkard walk as described in Chapter 1. In termite sim, all the turning is a result of pheromones, the terrains geometry, and agents reaching the edge of the terrain. There is no random turning performed in any of the simulations in Chapter 3. A variable could be added that adds this drunkard walk behavior to the current behavior of the control strategies.

DOTS

An alternative implementation of the agent representation would be to use Unity's Data Oriented Technology Stack (DOTS). This allows a large number of Unity GameObjects to be converted into Entities as referred to in DOTS. This would allow each agent to still make use of colliders. Allowing for a physics-based collision system. During the project, there was an attempt to implement DOTS, but the conclusion was that there were too many reasons to not implement it:

1. Much of the work done on the project would have to be scrapped and re-done.

2. There was no guarantee that it would work as it was still quite new and deemed experimental at the time of the project. There were no projects that had attempted something similar to this one using DOTS and I there would be a high chance of it becoming a time sink to even just get the same results as what the project had reached at that time.

3. The added functionality is optional: For self-assimilating emergent behaviour, inter-agent collision would be crucial. But for terrain forming builder robots, collisions would make some interaction more realistic, but are not necessary for emergent behaviour research.

4. It would not improve performance, as the current implementation is already very performant.

Evaluation functions

Currently, TermiteSim supports the experimentation of creating and running a simulation but is not able to evaluate the swarm's performance. It would be a great addition to the project to include several built-in fitness functions that would give the user some metrics to evaluate performance. For example, a fitness function that rates how well a shelter creating agent is at staying in the shade. To have a high fitness it would be able to quickly create suitable shelter.

Multiple agent types

As described above, TermiteSim is currently limited by its inability to simulate multiple types of agents in the same simulation. Steps have been taken to support this but it is not implemented in the interface, and is thus inaccessible within the simulator.

Separate variable settings for each agent type

There is a balance between usability, configurability and ease of implementation as described in Section 1.6. The choice was made to separate the parameters of the different pheromones to allow agents to interact with each one differently purely through parameter configuration. However, agents can have multiple states, and these different states require different interaction with the parameters. This is also true for different agent types. A better balance for the interface could be designed and implemented that would offer this functionality.

Functionality to user interface

TermiteSim has been designed to be configurable. However, the method to configure all the options have not all been implemented into the interface. Thus, limiting the functionality of TermiteSim. These options can be accessed by downloading and installing the open source Unity project. A few examples of these options are: Changing the colours of the pheromones, Adding more pheromone types, Changing the rules in the control strategies (requires coding), Creating new control strategies (requires coding), Loading and saving presets of parameters, Changing the look of the agents and the terrain (can be seen in Appendix Section 6.2).

4.3. Reflection on project approach

At the start of this project it was decided to spend one week performing a project sprint, which is to attempt quickly perform a watered down version of all the facets of the project. This lead to a very poorly optimized simulator, but served as a learning experience which has helped me gauge the difficulty of certain aspects of the simulator and forced me to define the interface between the different parts of the simulator. But most of all, it was helpful in igniting a mindset of working towards and minimal viable product, without getting stuck on the details. This has been especially useful for dealing with perfectionism helped plan out the rest of the project. However, doing a sprint at the start of a project is perhaps most useful in projects where a sprint MVP can be reached within a small fraction of the total project time. The insights gained from the sprint also lead to the decision to iteratively work on the separate parts, instead of fully finishing one with all the functionality described in Section 2.1. This approach was highly effective in getting a complete working simulator early on in the project. The sprint didn't help with planning the evaluation, in neither the method of evaluation and the planning for it. In hindsight there are many more things that would have been incredibly interesting to simulate, but they did not fit in the planning.

4.4. Conclusion

TermiteSim is a state of the art simulator that goes beyond the capabilities of previous simulators in regards to the resolution of the simulation, accessibility of starting new studies through a simple user interface, and adaptability to allow for more freedom in designing the swarms behavior. It is able to replicate previous simulations as shown in Chapter 3, with room for further research to be done using TermiteSim. Its functionality is verified by successfully replicating termite-like building behaviour in Sections 3.2, 3.4 and 3.5, but is not limited to it. TermiteSim can also be used to simulate a broader range of robot swarms as shown in Section 3.5 with the recreation of a swam or vacuum robots. This allows the findings of termite-like building behaviour to be implemented in more practical applications, and potentially disrupt the field of swam robotics in general. This increases the relevancy of swarm robotics as a method for automated construction and improves the rate at which swarm robotics can become a viable solution in real world applications. In Section 2.6.2 TermiteSim is also proven to be at least usable by users who have little to no knowledge of termites, and no knowledge of the inner workings of TermiteSim. Lastly, there is a large range of potential research to be done using TermiteSim as discussed in Chapter 4.

5. References

1. Oxford Economics. (September 2021) Future of Construction: A global forecast for construction in 2030. Oxford Economics. https://www.oxfordeconomics.com/resource/future-of-construction/)
2. K-team. (2012, June 28). Kilobot. http://www.k-team.com/mobile-robotics-products/kilobot
3. J. L. Deneubourg, “Application de l’ordre par fluctuations a la descrip- tion de certaines e´tapes de la construction du nid chez les termites,” Insectes Sociaux, vol. 24, pp. 117–130, 6 1977.
4. Bonabeau E, Theraulaz G, Deneubourg JL, Franks NR, Rafelsberger O, Joly J, Blanco S. 1998 A model for the emergence of pillars, walls and royal chambers in termite nests. Phil. Trans. R. Soc. Lond. B 353, 1561–1576. (doi:10.1098/rstb.1998.0310)
5. S. P. Sane, S. S. Ramaswamy, and S. V. Raja, “Insect architecture: structural diversity and behavioral principles,” Current Opinion in Insect Science, vol. 42, pp. 39–46, 12 2020.
6. Bruinsma OH. 1979 An analysis of building behaviour of the termite Macrotermes subhyalinus (Rambur). Thesis, Landbouwhogeschool, Wageningen
7. Calovi DS, Bardunias P, Carey N, Turner JS, Nagpal R, Werfel J. (2019). Surface curvature guides early construction activity in mound-building termites. Phil. Trans. R. Soc. B 374: 20180374. http://dx.doi.org/10.1098/rstb.2018.0374
8. B. Green, P. Bardunias, J. S. Turner, R. Nagpal, and J. Werfel, “Exca- vation and aggregation as organizing factors in de novo construction by mound-building termites,” Proceedings of the Royal Society B: Biological Sciences, vol. 284, 6 2017.
9. Ockoa S., Heydeb A., and Mahadevanb L. (2019). Morphogenesis of termite mounds. https://doi.org/10.1073/pnas.1818759116
10. J. Werfel, K. Petersen, and R. Nagpal, “Designing collective behavior in a termite-inspired robot construction team,” Science, vol. 343, pp. 754– 758, 2014.
11. Y. Mitakaand, T. Akino,“Areviewoftermitepheromones:Multifaceted, context-dependent, and rational chemical communications,” Frontiers in Ecology and Evolution, vol. 8, 1 2021.
12. A. Perna and G. Theraulaz, “When social behaviour is moulded in clay: On growth and form of social insect nests,” Journal of Experimental Biology, vol. 220, pp. 83–91, 1 2017.
13. A. Khuong, J. Gautrais, A. Perna, C. Sba¨ı, M. Combe, P. Kuntz, C. Jost, and G. Theraulaz, “Stigmergic construction and topochemical information shape ant nest architecture,” Proceedings of the National Academy of Sciences of the United States of America, vol. 113, pp. 1303–1308, 2 2016.
14. J. F. Boudet, J. Lintuvuori, C. Lacouture, T. Barois, A. Deblais, K. Xie, S. Cassagnere, B. Tregon, D. B. Brückner, J. C. Baret, H. Kellay, (2021) From collections of independent, mindless robots to flexible, mobile, and directional superstructures. Sci. Robot. 6, eabd0272. DOI: 10.1126/scirobotics.abd0272
15. Dorigo, Marco & Floreano, Dario & Gambardella, Luca Maria & Mondada, Francesco & Nolfi, Stefano & Baaboura, Tarek & Birattari, Mauro & Bonani, Michael & Brambilla, Manuele & Brutschy, Arne & Burnier, Daniel & Campo, Alexandre & Christensen, Anders & Decugnière, Antal & Di Caro, Gianni & Ducatelle, Frederick & Ferrante, Eliseo & Förster, Alexander & Guzzi, Jerome & Vaussard,

Florian. (2012). Swarmanoid: A Novel Concept for the Study of Heterogeneous Robotic Swarms. IEEE Robotics & Automation Magazine. 20. in press. 10.1109/MRA.2013.2252996.

16. D. Ladley and S. Bullock, “The role of logistic constraints in termite construction of chambers and tunnels,” Journal of Theoretical Biology, vol. 234, pp. 551–564, 6 2005.

17. Feltell, D., Bai, L. and Jensen, H.J. (2008) ‘An individual approach to modelling emergent structure in termite swarm systems’, Int. J. Modelling, Identification and Control, Vol. 3, No. 1, pp.29–40.

18. Dorigo, M., Theraulaz G., Trianni V. (2021, July) Swarm Robotics Past, Present, and Future, Proceedings of the IEEE | Vol. 109, No. 7, July 2021, DOI: 10.1109/JPROC.2021.3072740

19. Unity. (2023) Retrieved from unity.com

20. Sebastian Lague. (2021, May 29). Coding Adventure: Terraforming [Video]. YouTube. https://www.youtube.com/watch?v=vTMEdHcKgM4

21. Sebastian Lague. (2021, June 19). Terraforming [Source code]. Github. https://github.com/SebLague/Terraforming

6. Appendix

6.1. Project brief

TU Delft

Personal Project Brief - IDE Master Graduation

An accessible simulation of termite-like builder robot control strategies — project title

Please state the title of your graduation project (above) and the start date and end date (below). Keep the title compact and simple. Do not use abbreviations. The remainder of this document allows you to define and clarify your graduation project.

start date 18 - 10 - 2022 — 20 - 03 - 2023 end date

INTRODUCTION **

Please describe, the context of your project, and address the main stakeholders (interests) within this context in a concise yet complete manner. Who are involved, what do they value and how do they currently operate within the given context? What are the main opportunities and limitations you are currently aware of (cultural- and social norms, resources (time, money,...), technology, ...).

By themselves, termites are simple creatures that make seemingly straightforward decisions based on their immediate surrounding. However, termites do not work alone. They are part of a decentralised swarm that works together to survive. To do that the termites, specifically Macrotermitinae, build a mound that includes a royal chamber, fungal gardens, nursery galleries, cellars and ventilation shafts as seen in image 1. Somehow these simple creatures cooperate to create this remarkably complex structure. There is no architect or manager termite that has the grand plan of this mound and directs the others, so how do they achieve this? Through emergent behaviour. Emergence is the side effect of bringing together a combination of capabilities. Which, in this case, leads to a complex structure.

Inspired by their biological counterparts, swarms of termite-like builder robots could one day be used to additively manufacture large-scale structures on construction sites, in disaster areas, or even on Mars. But first we must learn how to influence, design and exploit emergent behaviour in autonomous multi-agent systems.

There are many strategies for controlling the individual and emergent behaviour. A control strategy based on stigmergy can be simple at first, but as rules are added the resulting behaviour becomes exponentially more complicated. Therefore, experimentation is required to adjust the control strategy until the outcome is desirable. This is best done through simulation in order to get the roughly the right outcome before physical testing.

However, current robot simulators are not build for the type of building termites do, where they source material locally and use that to additively manufacture a structure. For this a smooth terrain is required to build the ramps, slopes and other geometries required to navigate the structure itself during construction. In currently available simulators this could be achieved by building the world out such a large number of cubes that it would drastically impact performance to the point that simulating a swarm of a thousand agents would be infeasible. Instead of a discrete terrain made out of cubes, a non-discrete terrain, represented as a surface mesh, would allow for highly detailed terrain geometry while being much less computationally expensive. That is why a new simulator specifically for termite like builder robot swarms is required for further research in this field.

This project aims to create an open-source simulation platform of a termite-like builder robot swarm for experimenting with different control strategies and adjusting them accordingly. In other words: a tool for researchers to design desired behaviour of a builder robot swarm. An illustration of such a simulation can be seen in image 2.

space available for images / figures on next page

introduction (continued): space for images

image / figure 1: Cross section of a termite mound. Earthlymissions. https://earthlymission.com

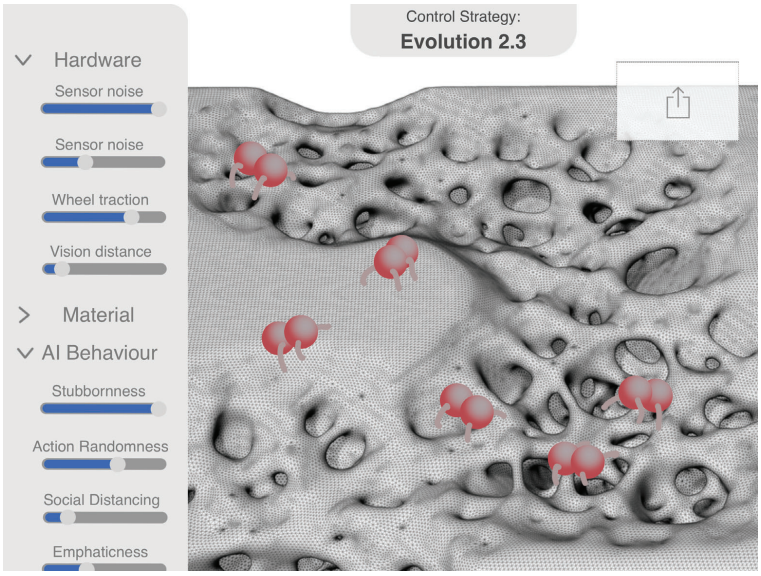


image / figure 2: Interface draft of termite-like builder robot simulation.

PROBLEM DEFINITION **

Limit and define the scope and solution space of your project to one that is manageable within one Master Graduation Project of 30 EC (= 20 full time weeks or 100 working days) and clearly indicate what issue(s) should be addressed in this project.

Automation in the form of swarm robots is an increasingly feasible opportunity for solving complex physical problems. However, how such a swarm is to be instructed and controlled is one of the many topics that requires further research. The problem is that, currently, there are no robot swarm simulators that are suitable for simulating termite like building.

This graduation project aims to provide a tool in the form of an open-source simulator to improve our understanding of emergent behaviour in termite-like multi-agent systems under distributed control, and allow researchers to evaluate different approaches that instruct the desired behaviour into such systems.

This will include the development of the virtual agents, the terrain and their interaction with an appropriate level of realism. The creation of a physics engine is outside of the scope and instead one that is readily available will be used. Several control strategies from literature will be implemented and their performance evaluated and compared to each other. Furthermore, a user interface to adjust the control strategy and environmental settings and review relevant performance data of the swarm will be created and tested.

Optionally the program could be hosted on a webserver and accessible via a web browser in order to negate the need to download the simulator and store it locally. Possible experimentation could be done with deep neural networks controlling the individual agents and/or using an evolutionary algorithm to optimise control strategy settings. Furthermore, a way to export the built structure is also a possibility.

The use or creation of a physical robot is outside the scope of the project because of the time investment. A visual interface for creating new control strategies is also outside of the scope.

ASSIGNMENT **

State in 2 or 3 sentences what you are going to research, design, create and / or generate, that will solve (part of) the issue(s) pointed out in "problem definition". Then illustrate this assignment by indicating what kind of solution you expect and / or aim to deliver, for instance: a product, a product-service combination, a strategy illustrated through product or product-service combination ideas, ... In case of a Specialisation and/or Annotation, make sure the assignment reflects this/these.

In this project I will make and test an open-source termite-like builder robot swarm simulator as a tool for researchers to and designers to experiment with control strategies, environmental factors and hardware limitations.

The simulator is a digital product which I plan to build in Unity, utilising the Nvidia PhyX engine for physics interactions. The service of hosting the simulator on a server to allow for persistent multi-user simulations is optional and secondary to the goal of this project. Tests will be conducted with users where they use the system and evaluate the interface and generate sets of control settings. I also intend to get a conference/journal paper submission about this project.

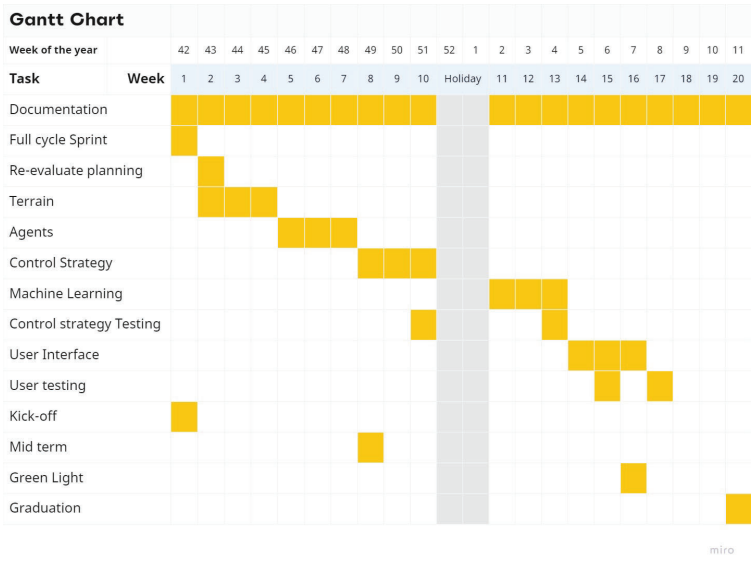
Initials & Name J. L. Lieveloo Student number 4567382

Title of Project An accessible simulation of termite-like builder robot control strategies

PLANNING AND APPROACH **

Include a Gantt Chart (replace the example below - more examples can be found in Manual 2) that shows the different phases of your project, deliverables you have in mind, meetings, and how you plan to spend your time. Please note that all activities should fit within the given net time of 30 EC = 20 full time weeks or 100 working days, and your planning should include a kick-off meeting, mid-term meeting, green light meeting and graduation ceremony. Illustrate your Gantt Chart by, for instance, explaining your approach, and please indicate periods of part-time activities and/or periods of not spending time on your graduation project, if any, for instance because of holidays or parallel activities.

start date 18 - 10 - 2022 20 - 3 - 2023 end date



The simulation can be divided into different sections that need to work independently. Therefore, I have my divided my planning into these segments after the initial sprint.

Full cycle sprint: In the first week, directly after the kick-off I will build a MVP demo by hacking together different parts of the project from (online) resources. This is meant to give me an idea of how the different parts will interface and where there will be friction.
Re-evaluate planning: Based on the knowledge and experience gained in the Full cycle sprint, I will re-evaluate my planning and adjust it here necessary. I will also redefine, add or remove features based on that my learnings.
Terrain: the first mayor part of the simulation I plan to tackle is the terrain, the world in which the agents will be able to move, pick up and place material.
Agents: In week 5 I plan to work on the agents, and give them the features required to sense and manipulate the world through simulated sensors and actuators.
Control Strategy: Starting week 8, the agents will become autonomous through the implementation of control strategies from literature. These can then be configured using their settings. These control strategies will then be evaluated in the simulation as seen in Control strategy testing.
Machine learning: After the control strategy is completed an evolutionary algorithm can be used to optimise the settings for the given context. The performance of the control strategies with these settings will then be compared to the performance of the settings used in the previous section.
User Interface: For the simulation to be accessible the control strategy settings will need to be easily accessible and changeable. For this the user interface will be developed and tested in User Testing.

MOTIVATION AND PERSONAL AMBITIONS

Explain why you set up this project, what competences you want to prove and learn. For example: acquired competences from your MSc programme, the elective semester, extra-curricular activities (etc.) and point out the competences you have yet developed. Optionally, describe which personal learning ambitions you explicitly want to address in this project, on top of the learning objectives of the Graduation Project, such as: in depth knowledge a on specific subject, broadening your competences or experimenting with a specific tool and/or methodology, ... Stick to no more than five ambitions.

This project interests me. The aspect of autonomy combined with inspiration from nature is right up my alley. For this project a lot of coding is required, which I enjoy doing. This interest started when I was introduced to Arduino, and throughout my study I have taken every opportunity to do more programming. If it was helping other students with their projects, doing my own ardruino projects, learning new coding languages and taking all the courses that required coding (minor Robotics Bsc electives: Mechatronics, Software, Msc electives: Digital Materials, cognitive ergonomics for complex systems 1 and 2, Experimental formgiving of visual information). This has left me with skills in several coding languages, but I feel that the coding aspect of my designs is never evaluated properly and am therefore unsure of its quality compared to professional programmers. This is the direct reason I set up this project to be as it is.

1. The main motivation I had for setting up this project is that I want to be able to program on a professional level and prove that I can do so.
2. I want to be able to use the tools used in software development such as Visual Studio and Git.
3. I want to be able to present this project to anyone without losing them. And for that I will need to be able to boil down complicated topics and ideas into layman's terms and describe them in an engaging way (I want to be better at presentations, a good presentation regardless of topic).

FINAL COMMENTS

In case your project brief needs final comments, please add any information you think is relevant.

6.2. Additional visualization options

These options are only accessible in the open-source Unity project of TermiteSim

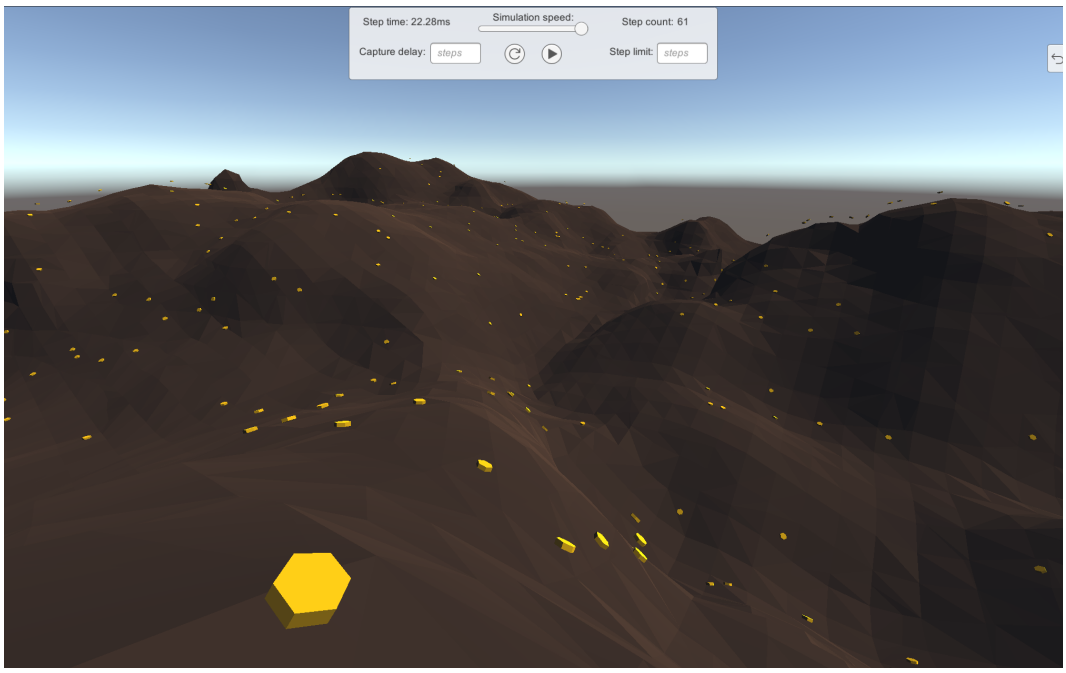


Figure A1. TermiteSim with different terrain color, agent color and agent shape.

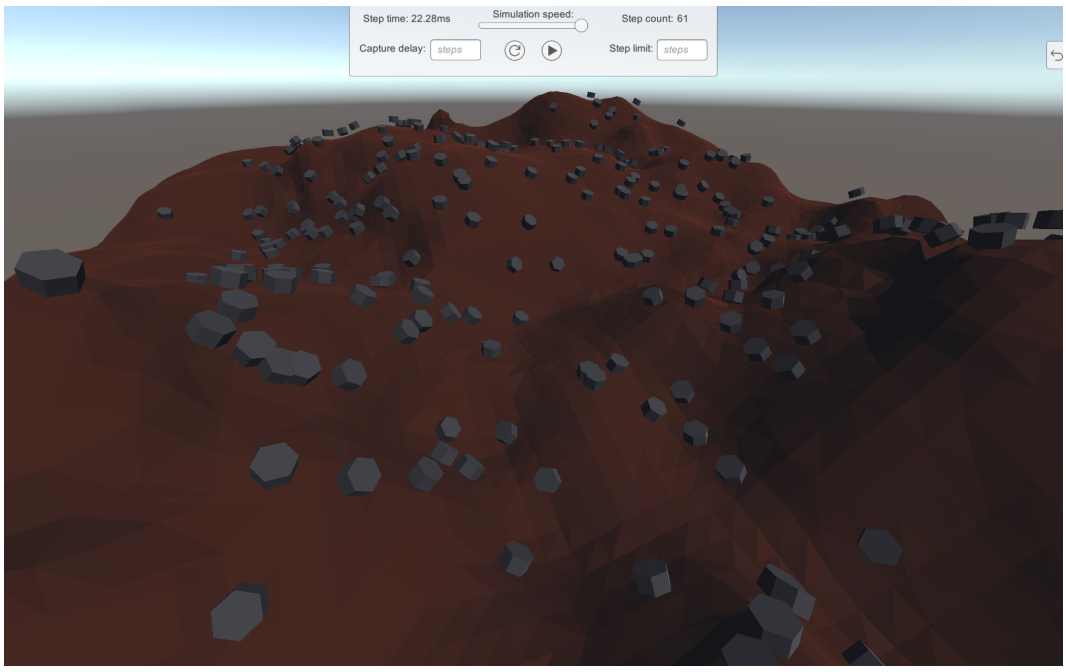


Figure A2. TermiteSim with different terrain color, agent color and agent shape.