Clinical Graphics iOS App
Bachelor project
Delft University of Technology

Authors:
Bert Bosch (4043405)
Fanny Lie (4088204)

Supervisors:
Dr. Anna Vilanova
Wynand Winterbach

June 17, 2014
Summary

The problem presented to the authors was to develop an iOS app for viewing 3D motion simulations on an iPad since Clinical Graphic’s current 3D PDF files cannot be opened on an iPad. For a more detailed description of the assignment, see appendix A [Project Assignment].

After defining a list of requirements followed by a research phase, the Scrum method was used for keeping track of this project’s process. This led to finishing the core features first and then adding more neat features as well. The agile approach requires a working app at the end of every sprint.

This resulted in an app as requested by Clinical Graphics to be improved by themselves and to be brought on the market. The app currently includes all core functions and some extra usability features. Optimisation and small additions can be done to improve the app before placing it in the App Store.

This report covers these subjects.
Preface

This report documents the development of the app to be created by order of Clinical Graphics. This project has been executed as a Bachelor project to conclude the Bachelor Computer Science at the Delft University of Technology. The project started on the 21st of April and concluded with a final presentation on the 24th of June.

We would like to thank the following persons:

- Dr. Ir. Peter R. Krekel from Clinical Graphics for the opportunity of this assignment and for providing us the resources needed for developing this app;
- Wynand Winterbach from Clinical Graphics for guiding and supporting us throughout the project, as well as being our contact person for Clinical Graphics;
- Dr. Anna Vilanova, our TU Coach, for supervising us and watching over our progress;
- The Computer Graphics and Visualization group at the Faculty of Electrical Engineering, Mathematics and Computer Science - Delft University of Technology, for answering our questions with regard to graphics when our knowledge fell short.
1 Introduction

Clinical Graphics is a company that creates interactive motion reports that doctors use to aid them in their treatment decisions. These reports include 3D models of CT scans in a 3D PDF. Based on the simulations that can be applied on these models, doctors can evaluate different possible situations. A request of the doctors was to view these 3D PDFs on an iPad instead of a regular PC.

Since there is no support for reading 3D PDFs on an iPad, the assignment of developing an iPad app was proposed to the authors. This app must be similar to the 3D PDFs: the app should make it possible for iPad users to view these 3D models and apply simulations on them. The detailed assignment description is included in appendix A Project Assignment.

This document provides the approach to building this app. The outline is as follows.

First, starting in chapter 2, the requirements of the software, in this case the iPad app, are defined and elaborated so that it becomes clear what the company wants and what the required functionalities of the app are.

This is followed by the project methodology in which the process planning and the used tools are discussed. This chapter gives insight in the project approach and process.

After that, the system design is presented and the design decisions are explained. It clarifies the overall structure and user interface and discusses several aspects of the product.

Different testing methods and results as well as feedback received from the Software Improvement Group, SIG, is discussed in chapter 5.

Finally, a list of possible recommendations are listed and elaborated for further improvement of the app.
2 Requirements

Before starting discussing the requirements, the current system and which features of that system have to be implemented on the iPad app is described. For this project a basic description was given, which is elaborated in 2.2. In final section of this chapter, the basic description is extended to a set of more detailed requirements.

2.1 Current System

Clinical Graphics provides the user with a 3D PDF. The PDF consists out of a report part that contains the description of their solution and a part that contains a 3D simulation of the concerning body part.

![Figure 1: Example of a 3D object in Clinical Graphic’s 3D PDF.](image)

Figure 1 shows the current interface of the 3D simulation included in the 3D PDF. In the 3D simulation, the body part is shown. With a mouse action the object can be rotated. Different plans can be selected: this alters the look of the object. 3D models of body parts are acquired by computed tomography (CT) scans. The CT mode shows the current state of the object. The resection modes show different solutions to the problem. Every resection differs in the way the object is altered. The differences are shown with coloured spheres. Every mode has specific animations. If an animation has a collision point, this point is highlighted with coloured spheres.

Their current system works with javascript. This is incorporated in the PDF. Their programma generates a PDF for every single job. The 3D viewer only works with the native program of Adobe, Adobe Reader. Other viewers are only able to show the regular (non-3D) part of the PDF.
2.2 Global Requirements

Clinical Graphics had minimal requirements defined for the app they had in mind. These requirements must be met in order to meet the company’s needs. These requirements are extracted from [A Project Assignment](#).

- Is suitable for running on an iPad2, iPad3, iPad4 and iPad mini
- Can be compiled and run on a physical local development iPad (for testing)
- Has a PIN/password that needs to be entered when the app is opened
- Can communicate with an external (Clinical Graphics) data server in such a way that each communication session is authenticated with a user’s unique ID + password
- (Securely) store a user’s login and password, and then to automatically supply it as needed when accessing Clinical Graphics’ servers. This personal information will be secured in a similar way as offline reports are secured: with encryption that is unlocked with the app’s pin/password
- Can download individual job reports (e.g. 3D scenes, each consisting of at least 2 3D models) from the external data server (the data size is assumed to be between 5MB and 15MB per report)
- Can locally save the reports so the reports can be viewed without the need for an internet connection to view them.
- Encrypts or locks the locally saved reports in some way so that it is difficult to access when the app’s pin / password is unknown.
- Allows a user to browse and view the locally saved reports
- Has a 3D render window that can display a the 3D objects and playback the transforms of one of these objects.
- Allows the user to delete job reports from the device

Most of these requirements define a high level description of the main functionalities of the app. For more details on the functionality a more detailed set of requirements must be given, which is defined in the following section.

2.3 Detailed Functionality and Quality Requirements

In addition to the requirements defined by the company, the following requirements have been set up to define a more concrete app.

2.3.1 Detailed Functionality

Additional features for the app which has not been included in 2.2 are defined in this section. These requirements have been acquired by deliberating with the Clinical Graphics and by reflecting the app concept on its usability.

- The user must be able to search through a list of jobs, both in local files and remote files on the server.
• The server will provide each job with a zip file, which contains a PDF report without
the 3D model and the 3D model objects. So the app must be able to display both.

• The user must be informed when there is no internet connection.

• The user must be informed of the download progress of a job.

• The user must be informed that the 3D model is loading.

• The possibility to change the PIN code.

2.3.2 Quality Requirements
To guarantee a certain quality level of the product, quality requirements have been defined
as follows.

• When displaying a 3D model, the frame rate should be at least 30 fps.

• The app may not crash.

• The app must operate even when there is no internet connection so the user can
open stored files.

• The app should not freeze when downloading a job.

• The 3D viewer should render the objects correctly.
3 Project Methodology

In this chapter the project methodology used for this project is explained. The original planning and what actual was used are enlightend. At the end of the chapter an overview of all the tools that were used to build the app is presented.

3.1 Process Planning

For this project the choice was made to use the agile approach by using Scrum. This section specifies how Scrum was used to plan the project and what really happend.

Scrum

Scrum is an iterative and incremental agile software development framework for managing product development. Each cycle starts with a meeting to discuss which features are added. At the end of each cycle the progress is evaluated. Because our team was a small team of two, this was done quickly and we could adjust accordingly. The cycles where defined hand in hand with the phases of the project. These phases were set by the course TI3800 - Bachelor Project. We integrated that planning into our own planning which resulted into the following table.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Deliverables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td>Project Plan</td>
</tr>
<tr>
<td>Research</td>
<td>Research Document</td>
</tr>
<tr>
<td>Development Part 1</td>
<td>Browser</td>
</tr>
<tr>
<td>Development Part 2</td>
<td>3D viewer, Code Evaluation 1(SIG)</td>
</tr>
<tr>
<td>Final Improvements</td>
<td>Final App, Code Evaluation 2(SIG)</td>
</tr>
<tr>
<td>Final Report</td>
<td>Final Report</td>
</tr>
<tr>
<td>Final presentation</td>
<td>Final Presentation</td>
</tr>
</tbody>
</table>

When having a look at the phases, the development of the app is divided into three parts: Development part 1, Development part 2 and Final Improvements. The Scrum method mainly applied for these three development phases.

Initial Planning

At first we divided the features over the weeks appropriately. In the weekly Scrum meetings the features could be shifted between weeks so that the planning would be adjusted accordingly to what has ben done and hat has yet to be done, since some features may have took less time than initially thought, and others might have taken more. The initial weekly planning is shown in Figure 2.

![Figure 2: Initial weekly planning](image)

When using Scrum, different sprints were created with the initial planning in mind.
At the start of every sprint the main features for that sprint were discussed and selected.

**Final Planning**
The initial planning was not suitable for the actual progress. Development of the browser went faster than expected so part two of the development has been moved up in the planning. This part went slower than expected and was done at the middle of the last part of the development.

For each week a list activities where described.

**Week 1 Planning**
- Meet with Clinical Graphics.
- Acquire a list of requirements.
- Write the Project Plan.
- **Deliverables** Project plan (See [Project Plan](#))

**Week 1-2 Research**
- Research the requirements
- Setup all the tools to be used for the project
- Execute the iOS development tutorial to get acquainted with Xcode
- Plan presentation day
- Book room for presentation
- **Deliverables** Research Report (See [Research Report](#))

**Week 2-3 Browser**
- Implement the Browser
- Login screen
- Pin lock screen
- **Deliverables** Demo browser

**Week 4-5 3D Viewer**
- OpenGl setup
- Obj-Loader
- Test Browser and obj Loader
- Process the SIG evaluation 1
- **Deliverables** Demo of 3D View

**Week 6 Final improvements**
- Process the SIG evaluation 2
• Clean up code.

**Week 6-7 Final Report**

• Write the final report

• Prepare demo for final presentation

• **Deliverables** Final Report

**Week 8**

• Prepare final presentation

• **Deliverables** Final Presentation

### 3.2 Tools

To develop the app for the iPad a variety of tools were needed. Choosing the right tools is important to keep the development on the right track. The chosen tools are mostly straightforward.

**Xcode & GitHub**

Apple has provided the development community with a program to develop apps for iOS called Xcode. This integrated development environment, IDE, is only available for Mac OS X. Thus, the development had to be done on a Mac. Other payed IDEs are available but expensive. The downside of Xcode is that it is not very complete in terms of features like code coverage. However, the version of Xcode used for this project (version 5.1) does support the use of GitHub. GitHub can be used by an interface which is integrated Xcode. This allows conveniently switching between branches and keeping track of commits.

**XCTest**

Xcode provides the developer with the framework XCTest for unit testing code. XCTest is a framework for the unit testing of Objective-C code. The simulator runs the code and executes all written tests on the app. Continuous integration is supported by Xcode. However, a Mac server has to be used to run the tests, which was not available for this project due to limited resources.

**iOS Simulator & iPad**

For user testing, the app must run. To run the app a simulator can be used. The simulator is automatically installed when Xcode is installed. The simulator supports different types of iOS devices. Since mouse events are more limited than real life touch events on an iPad, several touch inputs can be simulated. On the simulator touch events can be simulated by mouse clicks. Dragging is possible as well. When the alt-button is pressed a two finger event can be simulated.

In the assignment was stated that the app had to run on an actual iPad. The main difference between the simulator and an actual iPad was the speed of the devices. The simulator uses the available processing power of the Mac whereas the iPad is limited to the processing power of an iPad. Therefore testing the app on the simulator is faster usually than on an iPad.

**Google Drive & ShareLatex**

The planning and other textual documents were shared between the team members via
Google Drive. The final report has been made with sharelatex, this is an online version of latex where multiple people can make adjustments in one document.
4 System Design

This chapter gives insight into design decisions made in advance of and during the development of the app. The design can be divided in three sections. First the system architecture is described. Next, the graphical user interface (GUI) is illustrated. Finally, the serveral components in the system are explained.

4.1 Architecture

The app is divided in components. By using a Model-View-Controller model combined with a Client-Server model, our app is represented the way it is intended.

Model View Controller

Apple suggests using the Model-View-Controller (MVC) model to build apps for the iPad. Controller objects are a conduit through which views learn about changes in the data model, and vice versa. Views are notified of changes in model data through the apps controllers, and controllers communicate user-initiated changes to model objects. Whether they respond to user actions or define navigation, controllers implement an apps behavior. This model is recommended by Apple, and the program to be built can be achieved by using this model.

Server-Client model

For the app to fully function it has to receive data. This data is acquired by communicating with a server. To model this function, a server-client model is preferred.

Combined

The app itself has a MVC architecture although on a more global level it has a server-client architecture. The app calls the server for its information and the app
4.2 GUI

A important part of the app is the interface. The user interacts with the app by touch. This differs greatly in comparison with an application for a PC or Mac. The use of your hands instead of a mouse affects the design of the interface. Apple has certain guidelines how to make a design for an app. These guidelines are integrated in the parts that can be used to make the interface, for instance the layout of the views and how the buttons are represented.

The interface is created in a flowchart type manner. Every screen has a representative. See the figure in Appendix [II]. By creating links between screens segues can be made. A segue is a switch between screens. This segue can be executed by pressing a button or by completing a number of commands.

**Login screen**

When the app is launched for the first time, the user has to login to Clinical Graphics’ server. The login screen shows the user two text fields to fill in: the user id and the associated password. Entering text is done by the keyboard that appears when the user taps one of the text fields. If the user is using an external keyboard on their iPad, this physical keyboard is used for text input.

![Login Screen](image)

**Figure 5: Login Screen**

**PIN Lock**

As stated in the requirements, a pin lock had to be added to the system. The use of a PIN code is simple. The app asks the user to set a PIN code when the user logs in for the
first time and saves this PIN on the iPad. Every time the user is asked for the PIN, this PIN code is compared to the PIN code stored on the iPad.

However, entering the PIN code had yet to be designed. The regular way of entering a PIN code is using a keyboard for input. This could be done by showing the iPad keyboard, but the buttons would be placed inconveniently. That is why a new design was added by putting buttons in the center of the screen to represent the numbers so the iPad keyboard would not need to pop up. This is shown in Figure 6.

![Figure 6: Pin Lock view](image)

The user may choose not to set a PIN code. In this case, the user automatically agrees to the Terms and Conditions so that Clinical Graphics is not to be held responsible for unwanted leakage of data to third parties.

**Navigation Browser**

After logging in and setting the pin lock - or not, if the user has chosen not to - the main view is shown. This is composed of two parts: the navigation browser and the PDF/3D viewer. Initially, in the PDF viewer a PDF with instructions is shown. The navigation browser allows the user to search and select jobs to retrieve more info concerning this selected job. The *Local Files* tab represents all the jobs stored on the iPad. The *Search* tab can be used to search for a new job on the server. A search bar has been added to both the local and server files for searching through the jobs more efficiently. Finally, the navigation browser includes a settings tab. These setting give control over the app. For example, the user can change the PIN, the user can choose to use this PIN lock or not.
and the user can log out.

Figure 7: Local files view
Figure 8: Search Server view
Figure 9: Settings view

3D View
When a 3D model and animation is selected the models and animations are loaded and rendered in the 3D view. The view shows the objects in the middle of the screen with buttons surrounding it. The layout of the app’s 3D view is similar to Clinical Graphics’ original 3D PDF view to give the user a sense of recognition. The buttons for the animations had to be included in the view. If the buttons were placed on a different part in the interface, there would be less space left to view the model.
4.3 Component Design

Figure 11 shows the structure of the software. In this section, the components of this software are arranged on their functionality. These components are mainly divided into two parts: the 3D view and the browser. The login and PIN lock screens are discussed separately.

4.3.1 3D View

The 3D view has two main components: loading and rendering. The loading is done with information that is selected by the user in the browser. The object’s paths are given to the 3D view controller which loads all the different parts for the simulation of this object.
Figure 11: File structure
When the parts are done loading, the controller switches to rendering.

**Overall data format**

Data is downloaded from Clinical Graphics’ server. The browser sends the path of the selected data to the 3D view controller. Before the data is loaded, there has to be specified what has to be loaded. For this purpose, two files have to be present to specify what has to be loaded. One file is for the objects. The format is as follows [9]:

**Total 34**
Load 0 1 2 3 4 5 6......
Neutral 0 1 ..
Mode1 2 3 12 ...
Mode2 2 3 24 ...
Mode3 2 3 33 ...
static 1 3 6 ...
dynamic 0 2 ...
collisionPoints 34 35 ...

This specifies what objects are what and which objects are visible in different modes. If this file is not available, nothing is loaded.

For each animation there is a different file. Every animation has different names. For different plans, different animations names are present. So a file has to be delivered with the names of the animations. The file format is as follows:

11
flexion
abduction
internal
internal_50abduction
internal_90flexion
......

The first line is the number of buttons to be created and how many animations are to be loaded. The names of the animations have to be the same as the name of the folders where the animation files should be present.

**Objects**

When the files that specify which data has to be loaded, the objects are loaded. This is done by a .obj loader because the file extension supported by the app is .obj. Obj files have a specific format that is developed by Wavefront technology. The format is as follows:

`# List of Vertices, with (x,y,z) coordinates.
v 0.123 0.234 0.345 v ...
...
# Normals in (x,y,z) form.
vn 0.707 0.000 0.707
vn ...
...
# Face Definitions (see below)
f 3//1 4//2 5//3
f 6//4 3//5 7//6`

The vertices are points in the 3D space. Normal vertices are used for the lighting of the
model. Faces are the actual polygons to be rendered. Each face is a group of three vertices and three normal vertices. The format is parsed and the appropriate data parts are filled in.

After all the faces and vertices are loaded the .mtl file is loaded. This file specifies what kind of material the object is made of. The format is the following:

```
newmtl ourSpecialMaterial
Ka 0.00000 0.00000 0.00000
Kd 0.90000 0.85000 0.70000
Ks 0.17000 0.16000 0.13000
```

Ka is specifies the ambient, Kd the diffuse and Ks stands for the specular of the object.

A special kind of object are the collision points. These objects have to be semitransparent. Because the mtl file does not specify the alfa at this point, the alfa is set manually.

**Animations**
The app supports animations of objects by rotating and translating different objects. In the preload files the dynamic objects can animate. The animation consists of an array of values that are used for the rotation and translation. The format of animation files are as follows:

```
-13.9862,-13.0748,-0.4616,0.0006,0.1667,-0.1667,-0.7730,36.2374
-0.4709,0.0354,0.0015,-0.1666,5.8316,60.8861,0.1926,0.0527,-0.0001,-0.1666,12.4322,85...........
```

The first tree numbers represent the translation of the object and the following three the rotation. This is repeated until the end of the file.

**Rendering**
When all the objects and animations are loaded the controller starts to render the objects. For the rendering OpenGL ES 1.0 is used. Before the rendering can begin OpenGL ES 1.0 has to setup. Some of the main points are highlighted.

```
self.effect = [[[GLKBaseEffect alloc] init];
self.effect.light0.enabled = GL_TRUE;
self.effect.lightingType = GLKLightingTypePerPixel;

glCullFace(GL_BACK);
glFrontFace(GL_CW);
glEnable(GL_DEPTH_TEST);
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
```

GLKBaseEffect is used for the lighting and shadows. This a build in function of OpenGL ES 1.0. Only one light source is being used. The position of this light source is the user’s position. This allows the object to be visible at all times. After the lighting is set, some basic OpenGL settings are called. The glCullFace and glFrontFace are for setting how the faces are drawn, right side round or left side round. glEnable is used for depth test and blending. Depth test is used to determine which faces are visible and should thus be shown. If this is not enabled, the inside of the object would be visible instead of the outside. Blend is used for the alfa of the materials. When an object has an alfa less than the value: 1.0f, the object has to be drawn transparently.
After OpenGL is setup, the actual drawing can be done. The render function of each object, that has to be drawn, is called.

```c
− (void) render : (GLKBaseEffect*) effect : (GLKVector3) pos : (float) aspect{
    effect.material.diffuseColor = _diffuseColor;
    effect.material.ambientColor = _ambientColor;
    effect.material.specularColor = _specularColor;

    // setup modelMatrix
    GLKMatrix4 modelMatrix = GLKMatrix4Identity;
    if (_dynamic) {
        // animation
        modelMatrix = GLKMatrix4Translate(modelMatrix, _translate.x, _translate.y, _translate.z);
        modelMatrix = GLKMatrix4RotateZ(modelMatrix, _rotation.z);
        modelMatrix = GLKMatrix4RotateX(modelMatrix, _rotation.x);
        modelMatrix = GLKMatrix4RotateY(modelMatrix, _rotation.y);
    }

    // setup viewMatrix
    GLKMatrix4 viewMatrix = GLKMatrix4Translate(
        GLKMatrix4Identity, 0, 0, initialDistance);
    [self rotateMatrix:&viewMatrix];
    viewMatrix = GLKMatrix4Rotate(viewMatrix, -M_PI/2.0f, 1.0f, 0.0f, 0.0f);
    viewMatrix = GLKMatrix4Translate(viewMatrix, -pos.x, -pos.y, -pos.z);

    // construct Modelview
    GLKMatrix4 modelViewMatrix;
    modelViewMatrix = GLKMatrix4Multiply(viewMatrix, modelMatrix);
    effect.transform.modelviewMatrix = modelViewMatrix;

    // setup projectionMatrix
    GLKMatrix4 projectionMatrix = GLKMatrix4MakePerspective(
        GLKMathDegreesToRadians(65.0f), aspect, 0.1f, 1000.0f);
    effect.transform.projectionMatrix = projectionMatrix;

    [effect prepareToDraw];

    glEnableVertexAttribArray(GLKVertexAttribPosition);
    glVertexAttribPointer(GLKVertexAttribPosition, 3, GL_FLOAT, GL_FALSE, 0, _vertices);

    glEnableVertexAttribArray(GLKVertexAttribNormal);
    glVertexAttribPointer(GLKVertexAttribNormal, 3, GL_FLOAT, GL_FALSE, 0, _normals);
```
GL_FALSE, 0, _verticesN);  

glDrawArrays(GL_TRIANGLES, 0, _num_vertices);  
glDisableVertexAttribArray(GLKVertexAttribPosition);  
}

A few parts of this function are worth mentioning. Each object has a model matrix, this specifies how the object is positioned and how its rotation is. After that the view matrix is made, where the object is positioned in world space. The view matrix is then multiplied by the model matrix to rotate the object in world space. The projection matrix can be viewed as the camera of the user. After all the matrices are made and set the actual drawing is done. The faces and faces normals are set into the buffer and then drawn by using triangles. After the object is completely drawn, the buffer is disabled.

If the model is dynamic and an animation is in progress, the model matrix is translated and rotated. The order of this is of high importance due to the fact that matrix multiplications are not commutative. If the order is altered, the effect will break the animation.

### 4.3.2 Browser

The browser consists of three tabs: local files, remote/server files and settings. Once the user has saved a job, it is stored on the iPad and its PDF report and 3D model are available in the Local Files tab. For a job to be handled and presented correctly in Local Files, the server must provide a folder named after the job ID before it is compressed to a zip file. Within this job ID folder, a report PDF named job ID + side literal (either ‘L’ or ‘R’) and two folders, named ‘obj’ and ‘motion_data’, have to exist. If the job contains two sides, two reports are to be found, differing in the last literal only.

Downloading the zip file happens asynchronously so that the app does not freeze while downloading the file. A progress bar has been added to inform the user of the download progress. Unzipping the file is done by using the SSZipArchive library.

Via the Settings tab, the user is able to change the PIN lock code and even able to turn it on or off.
4.3.3 PIN Lock Login

To guarantee the privacy of patient data a lock can be set on the app. This can be done at the first setup of the app or afterwards in the settings menu. This safety is guaranteed by a two phase setup. A user has to enter the preferred code twice to set the code. The code is only stored if the two given inputs are equal. The PIN is saved on the app in the core memory system, which is encrypted by Apple’s own data encryption.

At every restart of the app and reactivation of the iPad from sleep mode, the app asks for the PIN code. The input is then compared to the data in the core memory. The lock screen unlocks when the PIN code matches the stored PIN code.

Clinical Graphics wants to give the user the option to use this lock, since entering a PIN code one after another is not user-friendly at all. The user however has to agree to terms that they themselves are responsible for the loss or leakage of data and patient data confidentiality so that the company cannot be held responsible.

4.3.4 Login Screen

Logging in is done at the first start up of the app. Without this information the app is of no use. No data can be downloaded form the server because no login data has been provided.

When the user enters his/her login information, the app asks the server if this user exists. Before this data is sent to the server, an authorization header is generated. This authorization header is in the format of a basic base64. Appendix [A][Authentication Header] presents the code that transforms the data to a base64string. When the authorization header has been made, the header is sent with the request to verify the user. The server only replies whether the data is correct. After confirming this data, it is stored in the core memory system of the iPad, just like the PIN code. At every request done by the app to the server, the app retrieves this data from the core memory.
5 Testing

When features are added, testcode is written for these features. Test code for apps are hard to write because most features are responses on touch events. Also most functions are private and hard to test from the outside. In response to this, all the public functions and imported functions are tested.

5.1 Automated Testing

Automated testing is used to keep track of all components and to check whether all functions still work after adding a new feature. After the feature is implemented, all the unit tests are run. If all tests succeed, the new feature is incorporated without breaking other functions. When a test fails, it means that the new feature broke an older feature, so adjustments have to be made.

5.1.1 Unit Testing

In this section the unit testing of the app is discussed. The unit testing is done with the XCTest framework. One of the most important and fragile function is the object loader, where a test object is loaded and all the different aspects of this object are tested.

5.1.2 Performance Testing

The 3D view was required to have a frame rate of at least 30 fps. This can only be tested by linking an iPad to Xcode. Xcode analyses the performance of the iPad on a few different aspects: memory, CPU and frame rate. A performance test was done by loading all models and then showing all models at the same time. This to be certain that the iPad could handle the maximum load. The results are promising and within the requirements of the project. The frame rate was 38 at minimum and the memory never surpassed 32% of the iPads memory. CPU however was in some point at the maximum. The object loader uses the full power of the CPU. The rest of the app does not use more than 98% of the 200% thats available.
5.2 Software Improvement Group Feedback

In this section the feedback of the Software Improvement Group is summarised. The feedback, which was given in Dutch, can be viewed in appendix C and D. There were two
moments of acquiring feedback: the first time was one and a half week before the end of
the project and after processing this feedback, the final submission took place at the end
of the project. First, the initial feedback is discussed. After that, the final feedback is
discussed. The feedback from SIG is mainly intended to improve the maintainability of
the app.

5.2.1 Intermediate Recommendations
The maintainability of the project was marked as above average. However, at some aspects
the code scored less than average. These aspects were Code Duplication and Unit Size.
The Component Balance was off as well. First a small explanation of these parts and what
can be done to improve it:

Duplication
Some functions have calls that are equal or very similar to another function. These calls
are categorised as duplicate. Often these lines could be abstracted to a new function so
that only one line of code is used to call all the other lines. This reduces duplication.
Other examples are that functions are the same but only differ in the variable that is
used inside the function. In that case, the functions can be abstracted by using an extra
parameter.

Unit Size
Unit Size represents the amount of lines inside a function. If a function has a lot of calls,
it can be hard to figure out what each part in the function exactly does. Comment lines
can be added but that would still be inconvenient. A better solution is to split the long
method into smaller methods which the main function calls. So the programmer can
view the main function and see what happens by the names of the smaller functions. If
he wants to change the function in some manner, he can edit the smaller function instead
of searching through the long function for that specific part.

Component Balance
Component balance can be view in how the project is divided. If a new developer opens
the project for the first time and all the class files are in the same folder, he does not
know with files belong to which component. The solution is to put the classes that belong
together inside a folder with the components name. Folders can be nested if that helps
dividing the components into a more understandable structure.

5.2.2 Improvements
Based on the feedback the code has been modified on the mentioned aspects. The biggest
improvements on unit size where made in the object loader and the 3D view controller.
The object loader parses the object files in a lineair manner. The code was also on long
piece of calls to fill the memory with the data. The loader has been split on the different
parts that were loaded. The 3D view controller had too many switch-statements. This is
improved by deleting them or by modifying them to make them shorter.
Code duplication was improved by making a master controller that had all the function that all the other controllers had. This deleted 4 functions in all the controllers.

The last thing that had to be improved improve was to structure the program. Instead of having all controllers in one folder, the controller folder items were divided into smaller folders. Each folder has a component name with the appropriate controller in them. If a different programmer opens the project he knows which controllers belong to which components.

5.2.3 Final Feedback

After the final features were added and all the recommendations were processed, the code was uploaded to SIG again for the final reading. The size of the project was increased with 12% and the score had increased as well. Most of the improvements were noticed except for two small points that remain to be improved. The project file was improved with a folder structure. However, if a programmer unfamiliar with the code were to open this project in another programming environment, the folder structure is gone. The folder structure has to be implemented in the folder and in the project. The other improvement was the absent of test code. The test code was not uploaded to SIG.

5.3 User Testing

In comparison with unit testing, user testing is used to test the functionality and usability of the app. In the development stage the functionalities were tested on the iPad simulator to speed up the process. Every sprint cycle we tested each other’s functions on the iPad simulator.

5.3.1 User Test Results iPad

At the end of the development stage we installed the app on a physical iPad and presented the app to different persons. Reactions on the app were mostly positive. However, there were some aspects that could be improved:

- Rotation of the models is too sensitive.
- Zoom function is too slow.
- Loading of the models is slow.
- After downloading a job, automatically opening the PDF or 3D model would be nice.
6 Conclusion

The objective of this project was to port the current system of Clinical Graphics’ 3D PDF to an iPad App. A research was done to find out if it was possible to even achieve this. After that the planning was made to ensure that the app functionalities were implemented in the right amount of time. The project development was done with the development method Scrum.

After the research and planning was made all the main features were specified. The result was that the app would follow a model view controller architecture. Besides that, the App had to be divided into different parts: the browser, PDF viewer and the 3D viewer.

Apple has given developers enough tools to develop an iPad app. Xcode was used to develop the App. This allowed the developers to write Objective-C and use a simulator to test the app.

Unit testing was done for the major parts of the system. It allows developers to keep track of the health of the system when new features are added.

Even though the app is complete in terms of functionality, some parts are not optimised yet. This became clear in the user tests. When a job file is opened, the 3D view takes a significantly long time to load all the objects. In addition, after downloading a job, automatically opening the PDF or 3D model was preferred.

For the main part every feature of the 3D PDF has been ported to the iPad app. Only some optimisation of features are necessary to completely mimic the original system and make it ready for deployment.
7 Recommendations

At the end of the project there are still some improvements to be made. A lot of the improvements are optimisation of different parts. A small overview and recommendations for those is give here.

ObjectLoader with raw data
When a 3D model is loaded, a string parser is used. This is slow when the model has a lot of faces/data lines. Every line is parsed and casted to the specific format that is necessary. An improvement would be to use raw data. So every part can be loaded directly instead of parsing it. Every 8 bytes would be a value. The only change that has to be made is that the first few lines should hold the info how many of what kind are after each other.

Animation with raw data
Although the animations are different from the 3D models, the same optimisation can be applied. The only difference is that the format of loading is already in the format it has to be, except that instead of using strings, bytes have to be used.

Unzip stream
When the data is downloaded from the server, the data is in the format of a .zip. After that the data is downloaded the zip is unpacked. This creates a map where all the unpack data is stored. The path for that file is than given to the appropriate paths in the system. However this creates more than double the data stored on the iPad. This can be overcome by using the zip as a folder, streaming the data of the zip. A different framework has to be found for that, or it has to be rewritten.

Fine tuning of the speed of animation and rotation
The 3D view has a given speed to rotate the object. This speed may be awkward for some users, the speed of rotation has to be fine tuned. A way of doing this can be "user field testing", which we did but without the end users of the system. Another recommendation on this aspect is a slider in the settings so that the user can edit the speed. The speed is not only for the rotation, but for the zoom-in, zoom-out function and the animation speed as well.
References


A  Project Assignment

This appendix presents the project assignment description given by Clinical Graphics.

An iOS App for Viewing 3D Motion Simulations

Clinical Graphics offers a motion analysis service for shoulder and hip surgery. Our clients (orthopaedic surgeons) send us CT-scan data, which we use to create 3D motion simulations. These simulations are used in diagnosis and pre-operative planning. For an example see the video on our website at www.clinicalgraphics.com. Currently, our 3D motion simulation results are sent to the surgeon by means of a 3D PDF file with javascript to playback simulations. We would like you to create an iPad version of this viewer application. Just as our 3D PDF files, the viewer should be capable of playing animations and visualizing the problem areas that prevent motion. The input for the viewer will be 3D models for the bones, a series of transformation matrices for the motion of the bones and point locations of where bones are colliding. The App should be easy to operate and have basic authentication functionality to get the appropriate simulation files for each user.

Clinical Graphics is a TU Delft Computer Science spin-off company. Were located in the Yes!Delft incubator, not too far from the campus area. Around since 2010, we provide an online scan analysis service: surgeons from around the world upload their scans to our website, which we then convert to 3D and analyze for the purpose of diagnosis and operation planning. We mainly perform motion simulations: given a scan of a hip or a shoulder, we can tell surgeons whether the joint has motion problems, whether patients need surgery to resolve those problems, and what should be done specifically. We put all this in 3D PDF reports and send it back to the surgeon. Our vision is that in 5 years time this will be standard practice in all Western countries. The potential of the simulations and analyses that we do consists of a significant increase in health care quality as a result of better diagnosis and operation preparations.
B Clinical Graphics iOS report viewer: Specification

Background Clinical Graphics creates interactive motion reports that doctors use to aid them in their treatment decisions. We currently focus on the analysis of femoroacetabular impingement (FAI) using CT, but also keep an open eye for opportunities involving other use cases and image modalities.

Goal To develop a user-friendly, attractive and functional iOS App with which doctors can retrieve and view Clinical Graphics motion simulation reports on their iPads.

Requirements Essential (minimum) requirements: An iOS App (source code + accompanying development environment) that

- Is suitable for running on an iPad2, iPad3, iPad4 and iPad mini
- Can be compiled and run on a physical local development iPad (for testing)
- Has a pin/password that needs to be entered when the app is opened
- Can communicate with an external (Clinical Graphics) data server in such a way that each communication session is authenticated with a user’s unique ID + password
- (securely) store a user’s login and password, and then to automatically supply it as needed when accessing Clinical Graphics’ servers. This personal information will be secured in a similar way as offline reports are secured: with encryption that is unlocked with the app’s pin/password
- Can download individual job reports (e.g. 3D scenes, each consisting of at least 2 3D models) from the external data server (the data size is assumed to be between 5MB and 15MB per report)
- Can locally save the reports
- Encrypts or locks the locally saved reports in some way so that it is difficult to access when the app’s pin / password is unknown.
- Allows a user to browse and view the locally saved reports
- Allows a user to browse and view the locally saved reports
- Has a 3D render window that can display a the 3D objects and playback the transforms of one of these objects.
- Allows the user to delete job reports from the device
C  SIG Evaluation Part I

Hierbij ontvang je mijn evaluatie van de door jou opgestuurde code. De evaluatie bevat een aantal aanbevelingen die meegenomen kunnen worden in de laatste fase van het project.

Deze evaluatie heeft als doel om studenten bewuster te maken van de onderhoudbaarheid van hun code en dient niet gebruikt te worden voor andere doeleinden.

Mochten er nog vragen of opmerkingen zijn dan hoor ik dat graag.

Met vriendelijke groet,
Eric Bouwers

Aanbevelingen

De code van het systeem scoort bijna 4 sterren op ons onderhoudbaarheidsmodel, wat betekent dat de code bovengemiddeld onderhoudbaar is. De hoogste score is niet behaald door een lagere score voor Component Balance, Duplication en Unit Size.

Wat opvalt bij het bekijken van de code is dat er geen duidelijke componenten-structuur zichtbaar is op het file-systeem. Dit maakt het voor een ontwikkelaar in eerste instantie lastiger om een algemeen beeld te krijgen van de functionaliteit die het systeem aanbiedt. Wij raden aan om kritisch te overwegen om de code in verschillende (functionele) componenten op te delen om zo een eerste indruk te geven van de high-level structuur van het systeem.

Voor Duplicatie wordt er gekeken naar het percentage van de code welke redundant is, oftewel de code die meerdere keren in het systeem voorkomt en in principe verwijderd zou kunnen worden. Vanuit het oogpunt van onderhoudbaarheid is het wenselijk om een laag percentage redundantie te hebben omdat aanpassingen aan deze stukken code doorgaans op meerdere plaatsen moet gebeuren. In dit systeem is er bijvoorbeeld duplicatie te vinden tussen de verschillende *Controller-classes. Het is aan te raden om dit soort duplicaten op te sporen en te verwijderen.

Voor Unit Size wordt er gekeken naar het percentage code dat bovengemiddeld lang is. Het opspitten van dit soort methodes in kleinere stukken zorgt ervoor dat elk onderdeel makkelijker te begrijpen, te testen en daardoor eenvoudiger te onderhouden wordt. Binnen de langere methodes in dit systeem, zoals bijvoorbeeld de 'GLModelViewCont roller.animation'-methode, is het mogelijk om het switch-statement te verwijderen. Het is aan te raden kritisch te kijken naar de langere methodes binnen dit systeem en deze waar mogelijk op te splitsen of te reduceren door het toevoegen van abstractie.

Over het algemeen scoort de code bovengemiddeld, hopelijk lukt het om dit niveau te behouden tijdens de rest van de ontwikkel fase. Als laatste nog de opmerking dat er geen (unit)test-code is gevonden in de code-upload. Het is sterk aan te raden om in ieder geval voor de belangrijkste delen van de functionaliteit automatische tests gedefinieerd te hebben om ervoor te zorgen dat eventuele aanpassingen niet voor ongewenst gedrag zorgen.
D SIG Evaluation Part 2

Hierbij ontvang je de resultaten van de hermeting van de door jou opgestuurde code. In de hermeting heb ik met name gekeken naar of/hoe de aanbevelingen van de vorige evaluatie zijn gemplementeerd. Ook deze hermeting heeft het doel om studenten bewuster te maken van de onderhoudbaarheid van hun code en dient niet gebruikt te worden voor andere doeleinden.
Mochten er nog vragen of opmerkingen zijn dan hoor ik het graag.
Met vriendelijke groet,
Eric Bouwers

Hermeting

In de tweede upload zien we dat zowel de omvang van het systeem (12%) als de score voor onderhoudbaarheid licht is gestegen.
De begeleidende e-mail wijst uit dat er wel degelijk componenten aanwezig zijn in de bron-code, maar dat deze niet op het file-systeem te zien zijn. Als het mogelijk is dat ontwikkelaars in de toekomst met andere editors gaan werken is het aan te raden deze twee component-indelingen nog gelijk te trekken.
Wat betreft de Duplicatie zien wij een stijging in de score, een deel van de duplicaten binnen de *Controller-classes zijn dan ook aangepakt. Ook wat betreft de Unit Size zien wij de score stijgen, de langste units binnen het systeem zijn op goede wijze aangepast.
Als laatste nog de opmerking dat ook deze aanlevering geen (unit)-testcode bevat.
Uit deze observaties en de begeleidende e-mail kunnen we concluderen dat de meeste aanbevelingen van de vorige evaluatie zijn meegenomen in het ontwikkeltraject. Het is goed om te zien dat bij een stijging van het volume ook de score voor onderhoudbaarheid is gestegen.
E Authentication Header

Function that takes the login data and transforms it into a Authentication header that is used to contact the REST API.

-(NSString *)stringByBase64EncodingWithString:(NSString *)inString {
    NSData *data = [NSData dataWithBytes:inString UTF8String]
        length:inString
        lengthOfBytesUsingEncoding:NSUTF8StringEncoding];
    NSUInteger length = [data length];
    NS MutableData *mutableData = [NS MutableData dataWithLength:
        ((length + 2) / 3) * 4];
    uint8_t *input = (uint8_t*)data bytes;
    uint8_t *output = (uint8_t*)mutableData mutableBytes];
    for (NSUInteger i = 0; i < length; i += 3){
        NSUInteger value = 0;
        for (NSUInteger j = i; j < (i + 3); j++){
            value <<= 8;
            if (j < length){
                value |= (0xFF & input[j]);
            }
        }
        static uint8_t const base64EncodingTable[] = "
            ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
            +/";
        NSUInteger idx = (i / 3) * 4;
        output[idx + 0] = base64EncodingTable[(value >> 18) & 0
            x3F];
        output[idx + 1] = base64EncodingTable[(value >> 12) & 0
            x3F];
        output[idx + 2] = (i + 1) < length ? base64EncodingTable
            [(value >> 6) & 0x3F] : '=';
        output[idx + 3] = (i + 2) < length ? base64EncodingTable
            [(value >> 0) & 0x3F] : '=';
    }
    return [[[NSString alloc] initWithData:mutableData encoding:
        NSASCIIStringEncoding];
}
F  Project Plan

This document function is to be a contract that holds the problem that will be addressed in one of the following sections. The project owner and the project members set the goals of the project in this document. The time planning and the approach is discussed as well. Some features and time planning can change overtime.

Summary

The projects goal is to build an iPad app that correctly shows 3D-models in PDFs. The greatest requirement of the project owner is a good framework that can be build upon. In 10 weeks time the project has to be completed.

Introduction

Clinical Graphics is a company that makes 3D models of CT scans and provide the doctors with a 3D PDF with info. This info contains approaches to deal with the problem at hand. A request of the doctors was to view the 3D PDFs on an iPad instead of a normal PC. This document provides the approach to building this app.

Motive

As stated in the introduction, the users would like to view the 3D PDFs on an iPad. A question of preference.

Approval and adjustments

If there are changes to the plan, it has to be discussed and approved by Clinical Graphics and the coach from the TU Delft.

Structure of the plan

First, a project description will be presented. The background, goal and other relevant information like demands, constraints and conditions will be clarified in this section. Second, the planning of this project will be discussed. This will be followed by the planning for this project. Finally, how quality will be preserved is stated in the final chapter.

Project assignment

In this chapter the project is further defined and specified. The project manager clarifies what the full requirements of the final product are.

Project environment

Since iPads are not able to correctly read and present 3D models in 3D PDFs, an app being able to do this is highly demanded for iPad users.

Goal of the project

The app has to give the user a means to view the 3D PDF.
Assignment formulation

The main idea of this project is to fulfill the requirements stated above. The encryption of the data that is downloaded is not necessary in this version. The framework where the requirements are programmed on is the main request. The project owner wants to have a solid backbone that his team can build on in the future.

Deliverables and services

The main deliverable is the iPad app. Further more, because the time span of the project isn't that long, the project owner wants a framework and as much of the above requirements fulfilled as possible.

Demands and Restrictions

This is up to the project team. The restrictions and demands will be updated throughout the project. Requirements:

- Has to work on a iPad 2, iPad 3, iPad 4, iPad air and iPad mini
- The user has to login with a id and password
- To unlock the app, a pin/password has to be entered
- When the app is booted for the first time, the user has to login and set a pin/password
- The user can browser through stored files and can delete these files
- The user can download new files from the server
- The server request has to send the id and password this the request
- The user can view the 3D PDF, the text and the model
- The model can perform animations on the 3D models
- The user has can change his pin/password in settings
- The pin/password and the id with that password have to be encrypted
- Has to work in both portrait and landscape view
- Has to work on standard iPad resolution (1024x768). For Retina displays, this will be upscaled automatically.
- The user should be able to log out.
- Connection with the server of Clinical Graphics is established via the https protocol.
- The waiting time between two failed pin/password attempts grows exponentially (until the maximum of one hour?)

Conditions

The code is reviewed by SIG twice. After the first submission (before the 13th of June) to SIG, their feedback has to be processed in the second and final submission. The requirements that are been set have to be met, as much as possible. Since the scrum method is applied to this project, there has to be a working prototype available every week - each week the prototype is improved with new features from the previous sprint.
Approach and planning

The first two weeks are dedicated to research, which is finished by delivering a research report. See appendix A for a global weekly planning. See Appendix B for a function planning. The project is divided into two phases. The first phase covers the framework of the app. This includes the graphical user interface (GUI), database connection and encryption. The second phase is reserved to integrate the 3D PDF viewer into the app, which mainly consists of parsing and loading the 3D models into the viewer and storing the files.

Project organisation

The purpose of this section is to show how the project will be approached. Agreements that have been made will be listed.

Organisation

The project team is responsible for the development of the app. The coach of the TU Delft has to monitor the progress of the project team. The project owner has to set the requirements, monitor the team on product level and provide the project team with resources.

Staff

Each member of the project team has to invest up to 420 hours into the project.

Administrative procedures

A meeting with the TU Delft coach has to be once every 2 weeks. Every week the project team has to meet with the representative of Clinical Graphics.

Finance

Clinical Graphics has to provide the project team with the equipment to build the iPad app. At the end of the project, a small fee will be given to the project team.

Reports

Clinical Graphics and the TU Delft Coach are given updates through mails and reports. The deadline of the first report to be handed in, a research report, is set in week 2 of the project. At the end of the project a document containing all the information about the project process and the product is to be provided to Clinical Graphics, the TU Delft coach and the Bachelor Project Coordinators 7 days before the presentation.

Resources

An iPad is provided by Clinical Graphics along with a Apple Developer Key. Remote access to a Mac Mini and sometimes a possibility to work locally at the office on the Mac Mini are provided. Not yet fully specified.
Quality Assurance

Quality is assured by means of the scrum process and by having the Software Improvement Group (SIG) check the code.

Since scrum is being used, at the end of each sprint, which in this case means every week, a working prototype should be available on the master branch of our GitHub.

The quality of the software is rated by SIG. This company will rate different aspects of the software by giving a number of stars to each aspect. The code has to be submitted twice to SIG: the first submission, which has a deadline of June 13th, is reviewed by SIG. The feedback given on this submission has to be processed for the second and final submission.

Appendix A: Weekly Planning

<table>
<thead>
<tr>
<th>Week</th>
<th>Tasks</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>21-24-07-04</td>
<td>- Determine and prioritize requirements</td>
<td>Project Plan</td>
</tr>
<tr>
<td></td>
<td>- Draw basic layout and models of the app</td>
<td>25-04-2014: meeting with Anna &amp; Wynyard</td>
</tr>
<tr>
<td></td>
<td>- Getting familiar with Xcode and Objective-C (Tutorial)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Research based on requirements</td>
<td></td>
</tr>
<tr>
<td>25-04-05</td>
<td>- Writing tests for the functions to be implemented</td>
<td>26-04-2014: meeting with Wynyard</td>
</tr>
<tr>
<td></td>
<td>- Create basic preview app</td>
<td>16-04-2014: mail Research Report to Anna</td>
</tr>
<tr>
<td></td>
<td>- More research, mainly encryption and database connection</td>
<td>02-05-2014: deadline Research Report</td>
</tr>
<tr>
<td></td>
<td>- Research report</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Log in &amp; connect to database on server</td>
<td></td>
</tr>
<tr>
<td>02-05-11-05</td>
<td>- Secure (encrypt &amp; decrypt) data, pin and login</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Browser, search jobs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Store jobs (PDF &amp; 3D model) on iPad</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Testing &amp; debugging</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Optimization</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Clean up code, extra time</td>
<td>27-05-2014: meeting with Anna</td>
</tr>
<tr>
<td></td>
<td>- Start on OpenGL viewer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Viewer, show model</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Viewer, app transformations, animations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Documentation for SIG</td>
<td></td>
</tr>
<tr>
<td>02-06-08-08</td>
<td>- Documentation for SIG</td>
<td>Data report</td>
</tr>
<tr>
<td>02-06-12-06</td>
<td>- Polish viewer</td>
<td>13-06-2014: deadline SIG [code &amp; documentation]</td>
</tr>
<tr>
<td></td>
<td>- Clean up code for SIG</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Submit code to SIG</td>
<td></td>
</tr>
<tr>
<td>06-06-22-06</td>
<td>- Final Report</td>
<td>17-06-2014: Meeting Anna</td>
</tr>
<tr>
<td></td>
<td>- Testing &amp; improvements</td>
<td>20-06-2014: receive SIG feedback</td>
</tr>
<tr>
<td></td>
<td>- Presentation</td>
<td>27-06-2014: Deadline Final Report [7 days before presentation]</td>
</tr>
</tbody>
</table>

Appendix B: Scrum Planning

<table>
<thead>
<tr>
<th>Event</th>
<th>Function 1</th>
<th>Function 2</th>
<th>Function 3</th>
<th>Function 4</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Project Plan</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Research Report</td>
</tr>
<tr>
<td>3</td>
<td>App password protection</td>
<td>Login for viewer</td>
<td>PDF viewer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>New connection with database</td>
<td>Browser for jobs on server</td>
<td>Browser for local files</td>
<td>extended jobs server browser</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>log and store data</td>
<td>PDF viewer running on the browser</td>
<td>Log not in settings</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>OpenGL window</td>
<td>Use OpenGL window</td>
<td>3D model viewer</td>
<td>Controller for assigning simulations on model</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Annotations/simulations model</td>
<td>Lighting</td>
<td>Prepare code for SIG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Debug</td>
<td>Optimize</td>
<td>Submit code to SIG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Final Report</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Presentation</td>
</tr>
</tbody>
</table>
Introduction

This report presents and explains the result of the research phase of the project *An iOS App for Viewing 3D Motion Simulations*. The main problem to be focused on is as follows:

What is the best solution for viewing the 3D models and motion simulations provided by Clinical Graphics on an iPad?

This report reviews each aspect that is needed for the app to be developed.

Analysis

First of all, the structure of the app is described to show what components are needed followed by the graphical user interface (GUI) design of the app. Now the overall structure and design is clear, the straightforward choices for development environment and programming language are explained. Several file formats and libraries to be used are addressed. Finally, on specific areas like encryption, file storage and database connection has been done some research, of which the results are explained in the final sections of this report.

Architecture

The app is divided in components. By using a Model-View-Controller model combined with a Client-Server model, our app is represented the way its intended.

![Model View Controller](image)

Figure 15: Model View Controller

**Model View Controller**  Apple suggests using the Model-View-Controller (MVC) model to build apps for the iPad. Controller objects are a conduit through which views learn about changes in the data model, and vice versa. Views are notified of changes in model data through the apps controllers, and controllers communicate user-initiated changes to model objects. Whether they're responding to user actions or defining navigation, controllers implement your apps behavior. This model is recommended by Apple, and the program to be built can be achieved by using this model.
Server-Client model  For the app to fully function it has to have data. This data is acquired by communicating with a server. To model this function, a server-client model is preferred.

![Basic client server model](image)

Figure 16: Basic client server model

Combined  The app itself is a MVC architecture but as a whole its a server-client architecture. The app calls the server for its information and the app self uses this information in his architecture.

Design

![Apple Development Process](image)

Figure 17: Apple Development Process

The design process is divided in stages. First, one begins with the concept. Then, a list of questions about the app are set. This is to provide the next stage with a body. By making the user interface next, the concept is specified further. A user interface does not do much without any logic backing it. After an interface has been created, one defines how users can interact with what they see by writing code to respond to user actions in the interface. The flow of the app is determined by events: system events or user actions. So the code behind the user interface has to react on the users input. These events result in the execution of the apps logic and manipulation of its data. The apps response to user action is then reflected back in the interface.
So to summarize this, an app is an event-driven program. This means that the app’s logic is called in response to events that the user generates. The development of this app is going to be structured as Apple suggests it: the concept first, then the interface and after that the backlogic.

Pin lock on the app is optional if the iPad itself is already locked with a pin code. Otherwise, the pin lock on the app is obligatory. If the user chooses not to set a pin lock on the app, the user has to agree with the terms and conditions. In that case, Clinical Graphics is not responsible for the privacy of data.
Figure 20: Pin unlock screen prototype

Figure 21: Main browser screen prototype
Development environment and programming languages

The default environment and programming language are Xcode and Objective-C respectively. Xcode is necessary to compile and test the program. An iPhone/iPad simulator is included in Xcode for testing. Objective-C is the language that works on iPad and is specifically developed for this. Other languages can be used by using wrappers, but this decreases flexibility and performance. This approach is usually used for porting programs. Cocoa Touch is a framework built on Objective-C and runs on iPads.

Compatibility iOS versions

The app must be able to run correctly on iOS 7. iOS 6 is a ‘should’ requirement in the MoSCoW model.

File formats

To ‘convert’ a 3D PDF to a readable file format for iOS, the PDF is split in two parts: one part consisting of the 3D model with the possibility of applying motion simulations and the other part consisting of the original PDF without the 3D model.

Text

If the text and images included in the original PDF are again represented in a PDF, but excluding the 3D model, some consistency between viewing the PDFs on the PC and iPad is preserved.

3D Model and Animations

OpenGL ES is a framework included in Cocoa Touch. OpenGL can be used for displaying .obj files with a loader that is written by us. For specification of .obj, see [8].

Bundled files

To retrieve these files from the server, it is convenient to have a bundle format so that associated files can be downloaded as one file. The .zip format is highly preferred by the client. There exists no standard unzip framework in Xcode’s standard libraries, but open source libraries for this purpose do exist. See section 3.5.

Libraries

As reinventing the wheel would be inefficient, different libraries are used for specific parts of the iPad app. Useful libraries are sorted by category below.

Interface

UIKit is used for the interface of the app. This library is included in Xcode. UIKit provides all the interface elements that are necessary to build an app. The UISearchBar class is used for creating a search bar.

Building blocks

Foundation library is used for all the primitive data models. The data models are wrappers of the original C primitives.

3D Model viewer

Core Animation Library and OpenGL ES are used for the 3D model viewer. Core Animation is a technology that allows one to make advanced animations and visual effects. OpenGL ES is a low-level framework that provides tools to support hardware-accelerated 2D and 3D drawing. The GLKit framework provides several other features to ease OpenGL ES development on iOS.
**Pdf Viewer**  “To gain even more control over file previews, you can use the Quick Look framework directly. The frameworks primary class is QLPreviewController. It relies on a delegate for responding to preview actions, and on a data source for providing the preview items.”  

A Quick Look preview controller can display previews for the following items:

- iWork documents
- Microsoft Office documents (Office 97 and newer)
- Rich Text Format (RTF) documents
- PDF files
- Images
- Text files whose uniform type identifier (UTI) conforms to the public.text type (see Uniform Type Identifiers Reference)
- Comma-separated value (csv) files

The Quick Look framework is integrated in Xcode’s standard libraries.

**Unzip**  For convenience, Clinical Graphics wants their users to be able to download a bundled file, preferably in .zip format. Since Xcode has no unzip framework, SSZipArchive is used. SSZipArchive is an open source library for unzipping .zip files.

**Encryption**

Some aspects of the program have to be encrypted. For instance, the user has to log in to the server with an email address and password, which has to be encrypted and sent via an https connection. Besides this, a lock screen with a pin password is required as an extra security measure, so that no unauthorized person can view the patient’s information. This pin code must be encrypted as well. There are different kinds of encryption. Because the goal is to develop an iPad app, the security measures provided by Apple can be used. Every iOS device has a dedicated AES 256 crypto engine built into the DMA path between the flash storage and main system memory, making file encryption highly efficient. Along with the AES engine, SHA-1 is implemented in hardware, further reducing cryptographic operation overhead.

**File storage**

PDFs and 3D models (and the associated animations) must be downloaded and stored on the iPad in order to view them locally. Each app has a so called ‘Sandbox’. This is a place where all the folders and data files of the app are stored. Each app has his own Sandbox. This is created when the app is installed. See figure 3. iOS uses the class NSFileManager to perform many generic file system operations and insulates an app from the underlying file system.
Database connection

Communication with the Clinical Graphics server will occur via https (NSURLSession class.) To get the data from the server an id and password has to be send with the request. To download a file the NSURLSession class can be used. This class supports custom authorization and piecewise data receiving. This class also supports HTTP and HTTPS connections. [6]
H Storyboard - GUI design

Figure 23: Storyboard - GUI design