5CCS2SEG - Major Project Report

Dental Radiation Visualisation

Client: Dr Isabel Sassoon and King's College Dental School

Vu, Kim-Anh
1707295

Safiyuddin, Abdulqadir
1741772

O'Neill, Michael
1724137

Qadri, Haroon
1704847

Balasundram, Aakash
1718841

Gunaratnam, Logithan
1750132

Anton, Luca-Dorin
1710700

Alim, Mohamed (Awad)
1731405

Introduction

The aim of the project is to assist the researchers and clinicians in the oral cancer field assess the effects of radiotherapy and aiding in the devisement of treatment plans for new patients before they undergo radiation therapy. This has been achieved by creating an interactive 3D visualisation of the teeth that displays data from previous patients with similar characteristics loaded in. In addition, a graph feature is available to provide an alternative way to visualise the data sets.

Outcome

This desktop application was developed using the real-time engine Unity 2018.3.5f1. The teeth model was created using the 3D computer graphics software called Blender and was imported into Unity. Some of the most important high-level features of our system are:

- 3D visualisation of the teeth
- Application of a custom heat map onto the teeth model
- A feature which allows the user to filter patients where filtering is dependent on eight variables
- Ability to aggregate multiple data sets into one value set and display it on the teeth model
- Ability to compare two different sets of data and visualise the comparison simultaneously
- Navigation cube, which allows quick modification of the viewing angle
- Feature to compare different data sets using a bar chart or box plot
- Help option to assist in navigating the application by providing a tutorial
- File Manager - allows the ability to browse user's files and import new CSV files

Due to the Unity platform's ability to deploy applications to over 25 platforms, release versions of Windows, Linux and macOS can be generated, which fulfills the client's request that the desktop application should be compatible with Windows. However, using our initiative, we are providing all three version to the client, because it would save the client's time if they decided to use a different operating system, as they would not have to use other tools, such as a virtual machine to run this application.

All important features were implemented.

Design

The diagram below shows the several possible actions a user can take.

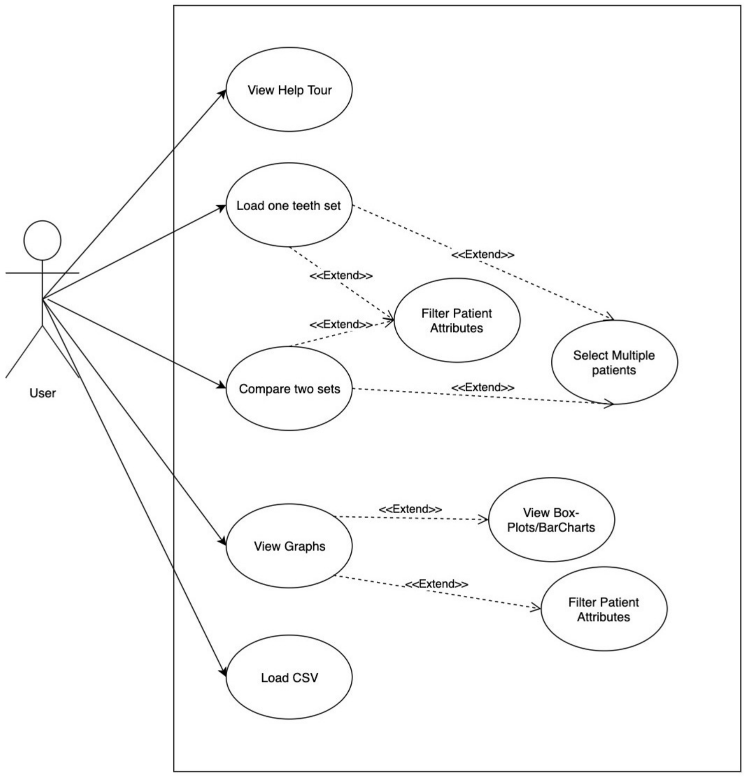


Figure 1: Use Case Diagram

We used Unity for our project, thus, we were inclined to adhere to Unity design standards. Unity design standards guide on how to structure the project, in aspects such as file structure and architecture. This is very important, as, for example, improper use of Resources folders will increase application startup time and the length of builds. Specifics for Unity design standards can be found on the Unity website: www.unity.com. One example that was particularly prominent for is, was the class all Game Objects must follow: all files that are game objects must inherit from the Unity root class, MonoBehaviour; these are slightly atypical such that they cannot be normally instantiated and thus not directly testable.

Due to the use of Unity, we felt it was necessary to implement the "Humble Object Design Pattern" in our code, as it enabled us to take a test-driven development approach, which made development more agile and allowed us to make changes to code with ease when adapting to modifications in the client's requirements. Furthermore, due to Unity being commonly used for game development, achieving loose coupling is not a priority, as importance is placed on performance rather than maintainability, thus the decision to implement "Humble Object Design Pattern" was to also achieve loose coupling, high cohesion and SOLID code.

We centred the design of our application around performance. As this is a live 3D rendering program, we took every step to ensure our program uses the least amount of system resources while also running smooth and looking sharp. Naturally, compromises had to be made. From the start, we knew this would be one of the key areas of concern. When designing the teeth model using Blender, we tried to minimise the number of polygons used while also trying to maintain a crisp image. As the teeth model can be rotated 360 degrees with zoom functionality and the ability to open and close the jaws, we took careful consideration when adding light sources to the scene. As the light sources are static and the teeth model is dynamic, shadows will be formed. By strategically placing the light sources and setting their intensity, we were able to get good lighting, reduce shadows while having only a few light sources, thus improving performance again. Adhering to these good Unity design standards, our application averages over 60fps, which is an industry standard.

Below is an architecture diagram which shows how the Unity Engine interacts with the application that we have made. The "Core Engine" and "Scene Manager" interact with one another. The "Scene Manager" controls what scene is displayed to the user and as you can see, there are two types of scenes: Data Controller and Graph Controller. Data Controller is where the tutorial and all the actions that are related to the teeth model are situated, whereas Graph Controller is where all actions related to any type of graphs are located. This shows our use of the Model-View-Controller pattern, which provides many benefits to our codebase, one being the ability to cleanly test our logic classes, and the separation also allowing us to effectively pinpoint errors, and swiftly correct them as we progressed.

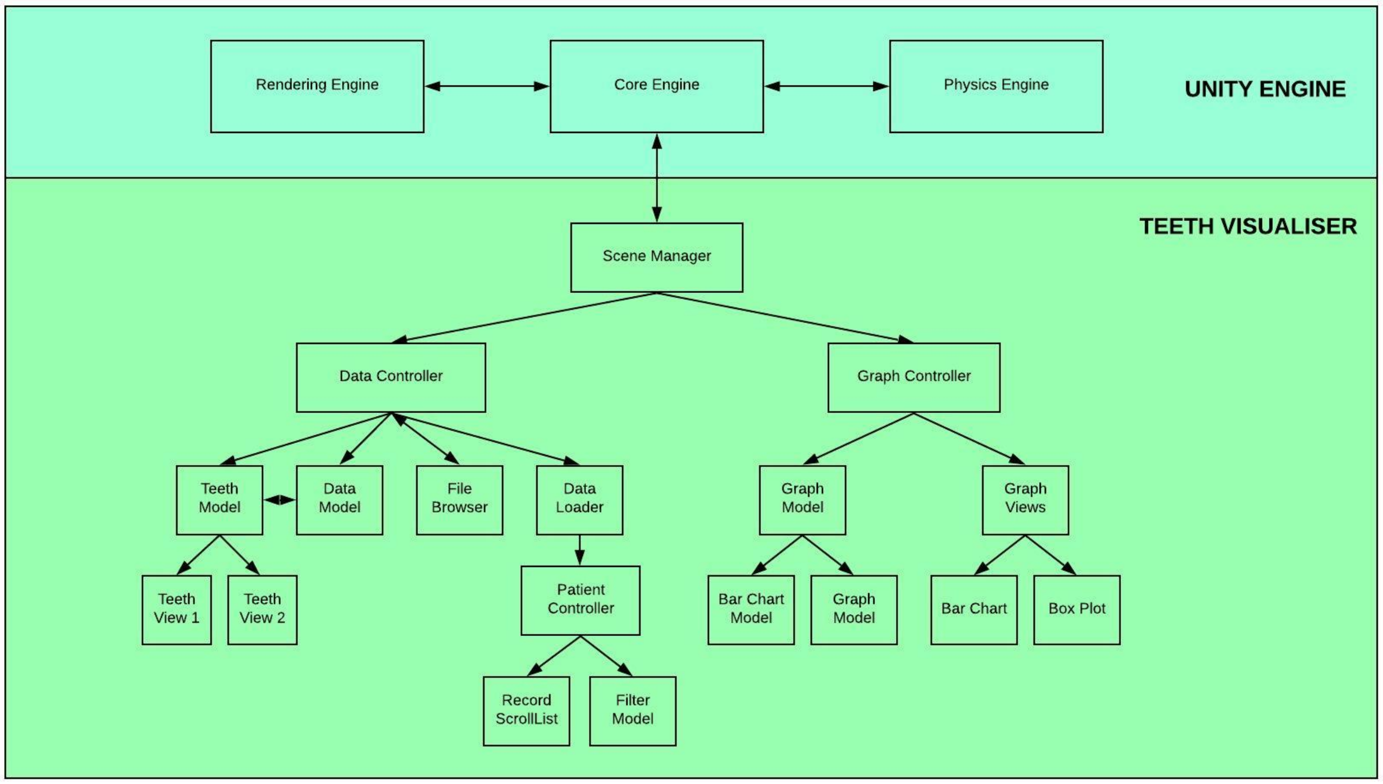


Figure 2: Architecture Diagram

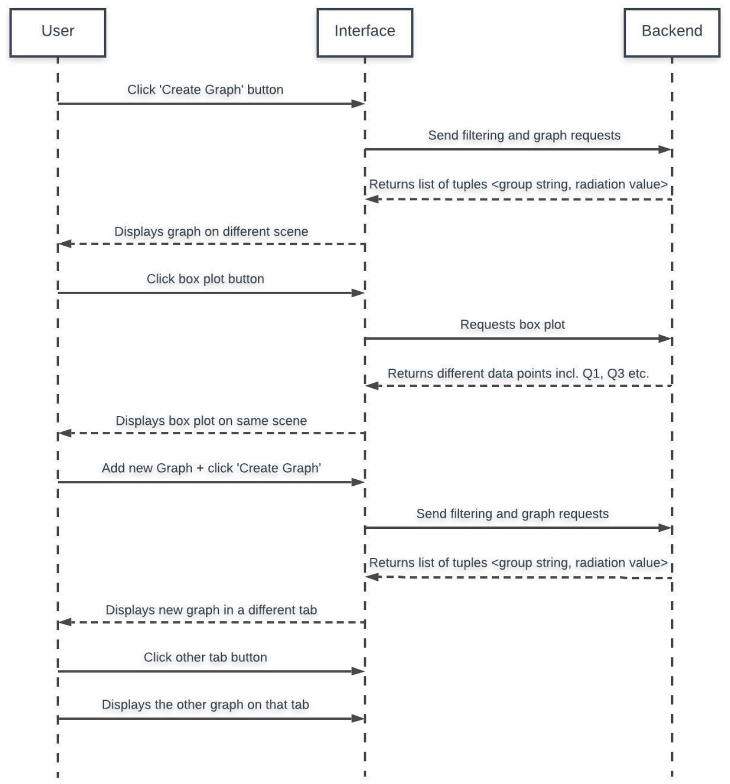

User
Interface
Backend
Click 'Create Graph' button
Send filtering and graph requests
Returns list of tuples <group string, radiation value>
Displays graph on different scene
Click box plot button
Requests box plot
Returns different data points incl. Q1, Q3 etc.
Displays box plot on same scene
Add new Graph + click 'Create Graph'
Send filtering and graph requests
Returns list of tuples <group string, radiation value>
Displays new graph in a different tab
Click other tab button
Displays the other graph on that tab

Figure 3: Graph Sequence Diagram

Figure 3 displays a sequence diagram describing the functionality of the "Graphs" feature of the application.

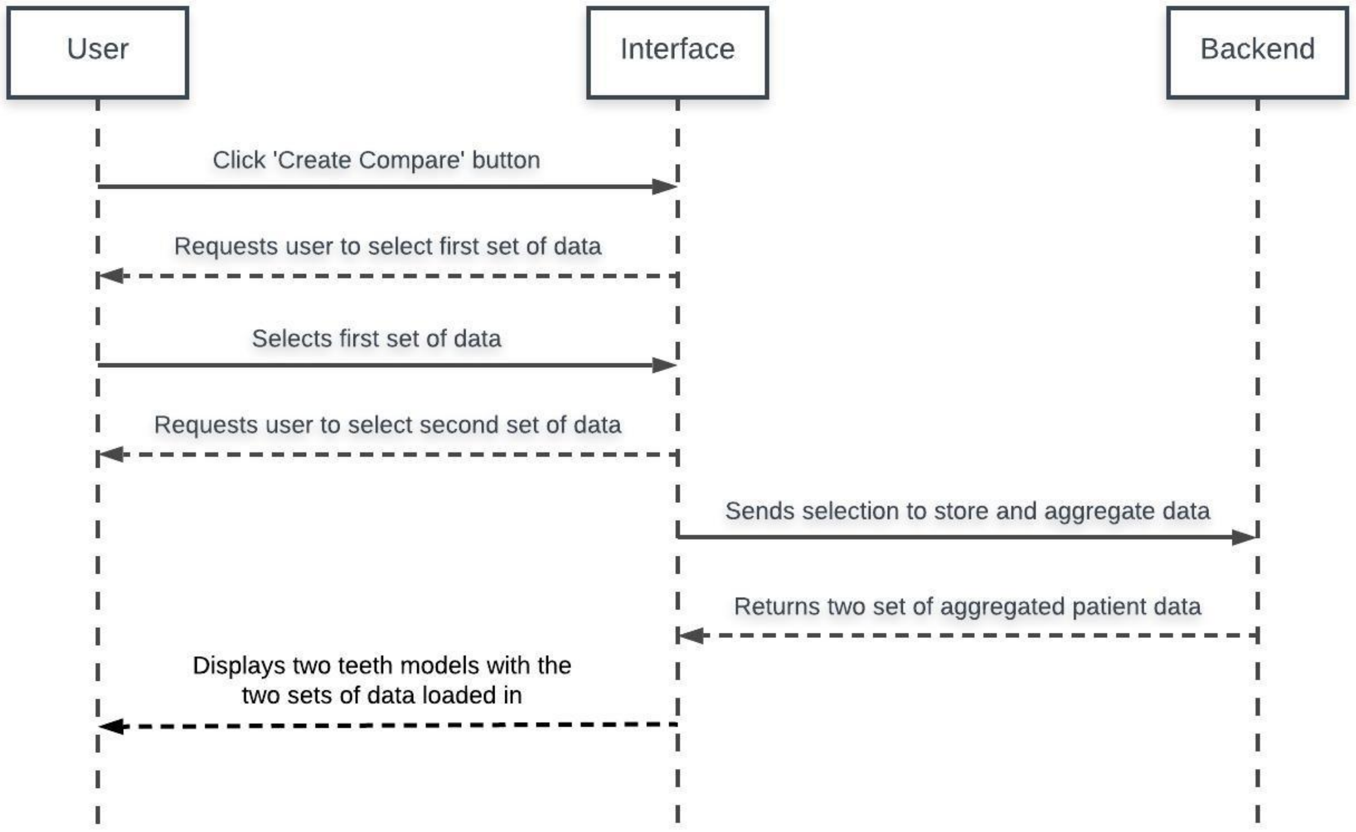

User
Interface
Backend
Click 'Create Compare' button
Requests user to select first set of data
Selects first set of data
Requests user to select second set of data
Sends selection to store and aggregate data
Returns two set of aggregated patient data
Displays two teeth models with the two sets of data loaded in

Figure 4: Compare Feature Sequence Diagram

Testing

The possibility of changes in requirements and giving the client early opportunities to see the software were two main reasons why we adopted the agile approach, resulting in using agile testing practices.

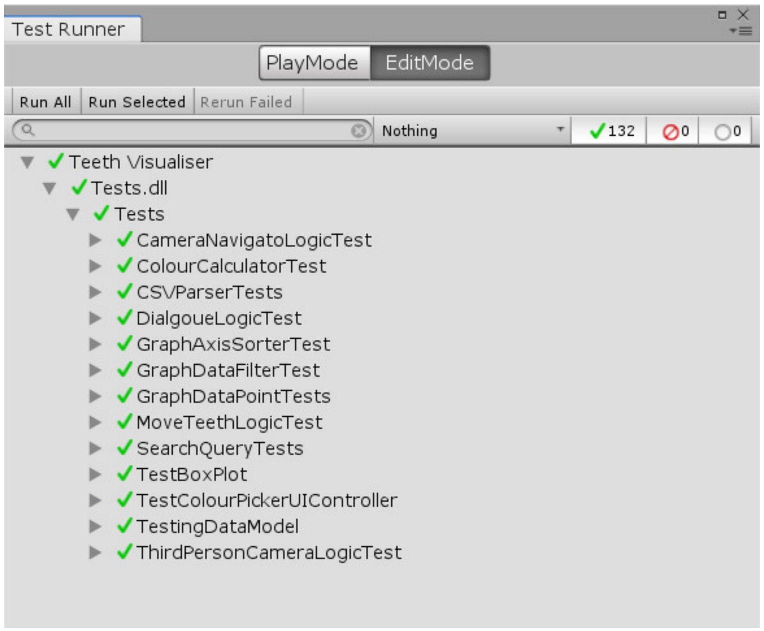
From the beginning, we embraced the extreme programming practice, ′Test first development′. We decided that it was more effective for each team member to be responsible for the testing of the user story that was assigned to themselves for each sprint, rather than creating a development team and a testing team.

Thus, our main testing approach was “Test Driven Development” and the “Humble Object Pattern” using Unity’s own test framework known as Unity Test Tools which includes: Unity Test Runner, a tool which integrates the NUnit library into Unity, allowing developers to perform unit testing; Integration Tests Runner, a tool which aids developers with integration testing. Also, acceptance testing was used alongside a screen recording (Table can be found in “Appendix”) for UI testing as automated UI testing tools were very limited as Unity is a game development platform and is frequently used to develop games, thus importance is placed on the presentation of the game, which is difficult to create automated tests for. Moreover, we created a new class which provided mock data for testing, as that would remove the dependency on the CSV parser to be functional and also isolates the behaviour of the patients data and complexity of the class to simulate the behaviour.

The “Humble Object Pattern” is when you extract all the logic from a component into a separate object and this approach was used because GUI objects, as mentioned before, are difficult to test efficiently for many reasons, including the need to use asynchronous tests due to Unity UI. This was an approach that made Unit testing possible and helped ensure us and the client that our code has been sufficiently tested .

NUnit test scripts can be found in the “Assets $\rightarrow$ Tests” folder - please see the figure 5 as evidence of the 132 Unit Tests passing.

Furthermore, to make sure the client was happy with the progress and the product, we regularly checked the application against the requirements and also set up meetings with the client to make sure that features were implemented correctly to the clients’ satisfaction.



```
▼ ✓ Teeth Visualiser
  ▼ ✓ Tests.dll
    ▼ ✓ Tests
      ▶ ✓ CameraNavigatoLogicTest
      ▶ ✓ ColourCalculatorTest
      ▶ ✓ CSVParserTests
      ▶ ✓ DialgoueLogicTest
      ▶ ✓ GraphAxisSorterTest
      ▶ ✓ GraphDataFilterTest
      ▶ ✓ GraphDataPointTests
      ▶ ✓ MoveTeethLogicTest
      ▶ ✓ SearchQueryTests
      ▶ ✓ TestBoxPlot
      ▶ ✓ TestColourPickerUIController
      ▶ ✓ TestingDataModel
      ▶ ✓ ThirdPersonCameraLogicTest
```

Figure 5: Evidence of 132 Unit Tests Passing

Team Organisation

The team was initially divided into 2 sub-groups; front-end and back-end teams. The front-end team consisted of Logi, Awad, and Aakash. The back-end team consisted of Michael, Haroon, Luca and Abs. Due to the nature of the project and the agile approach that we have taken, all team members started working on the front and back end tasks simultaneously. Some members were assigned additional roles, Haroon being the team leader, Logi the GitHub Master, Luca the team representative and Slack Master, Awad and Abs were responsible for recording the meetings on Team Feedback . We used the agile methodology with a sprint every week and a Kanban Board to track our progress. For testing, we took a test-driven approach with every member testing as the project progresses. From weeks 1 to 3, Logi and Kim-Anh worked on the teeth model. Aakash and Awad worked on the UI and the rest of the team worked on the backend. From week 4 onwards, Logi focused on the front and back end of the model colour with radiation. Aakash, Abs and Kim-Anh worked on the Graph implementation. Luca worked on the camera. Haroon worked on the data parser. Awad, Aakash and Michael worked on the UI with Michael also working on the backend architecture.

Other Pertinent Information

Our team maintained important documentation such as a Risk register, a Gantt Chart, Acceptance testing form and other documents on our google drive. Link to the google drive is below.

Link: `https://drive.google.com/drive/folders/1rU73zZ3FfNZOgyyOEulj4GX_ffmgQVlr`

The Risk Register was used to document risks, and actions to manage each risk. It was accompanied by a Risk Assessment Matrix to help us classify each risk. We also had a Risk Management document. Our Acceptance testing form had the procedure number, the name of the input/procedure, the expected result/output, a tick box for whether the test passed and a timestamp for when the procedure is done in our screencast. This acceptance testing form was crucial in our testing as in Unity you can't test the User Interface components.

Due to the nature of this project involving computer graphics and none of our team members having done any graphics related work beforehand, there was a steep learning curve for all members of the team. We spent the first weeks researching on which graphics engine to use. After our first client meeting with Dr Sassoon the requirements became much clearer, so we decided to use Unity. The team spent another 2 weeks learning Unity. We had a few hiccups that we were able to overcome like some of the team members found their scripts and animations to be missing every time they pulled from remote, thanks to our Git Master Logi, we were able to trace the problem back to the gitignore file in the repo having a .meta tag in it. With the resilience and determination of the team, we were able to overcome many such hiccups that arose due to Unity being a foreign object to us at first but now is like second nature.

Major changes in the requirements occurred over the course of the project, such as the "Compare" feature which was not specified in the requirements beforehand and also the ability to visualise the data sets as box plots, which was required by the client later on in the project. Furthermore, it was discovered that the client also wanted to pick their own colours for the heatmap which would be applied to each tooth depending on their values within the range specified. All modifications in the requirements were taken aboard and implemented.

Extendability

During the development of the project, we took into consideration the possibility of adding new features as the requirements of the client changed over time. These features include:

- The ability to have a four-way split screen
- Split screen graphs

- Adding more types of graphs and more graph features, such as overlaying different graphs
- More advanced tutorial tour with interactive features
- Improved 3D teeth model
- More sleek UI
- More advanced file management system
- Using machine learning to aid in pattern recognition in data sets provided

Appendix

Procedure No.	Input/Procedure	Expected Result/Output	Passed	Video Time
1	Opening desktop application	All teeth models and buttons are displayed on the screen	✓	2:16
2	Displaying filtered list of patients	—	✓	-
2.1	Clicking left sidebar button (*SidebarButton.cs*)	Panel slides from the left and is visible to user. Contains dropdown lists and sliders.	✓	3:06
2.2	Clicking dropdown buttons (*Dropdown.cs*)	Dropdown expands to show list of toggles	✓	6:02
2.2.1	Clicking a toggle that is not “All” (*Dropdown.cs*)	“All” toggle automatically is turned off	✓	6:25
2.2.2	Unselecting all toggles (*Dropdown.cs*)	Application will not allow the unselection of all toggles	✓	N/A
2.3	Modify the age min-max slider (*MinMaxSlider.cs*)	Numbers on top of the slider changes dependent on the positions of the 2 buttons.	✓	6:36
2.4	Click “Filter” button (*UserInput.cs*)	List of patients (results) are displayed in the panel underneath the filter components and are within the age range specified.	✓	6:44
3	Click “Reset” button on the filter panel (*Dropdown.cs*)	All filters and lists of patients are resetted to default positions	✓	7:48
4	Click “Select All” (*RecordScrollList.cs*)	All instances of the displayed patients are selected	✓	8:29
4.1	Click “Select All” again (*RecordScrollList.cs*)	All instances of the displayed patients are unselected	✓	8:33
5	Loading one patient onto teeth model (*Search Query.cs, RecordScrollList.cs UserInput.cs*)	—	✓	-
5.1	Click “Load” button without selecting a patient/s	Displays error message	✓	N/A
5.2	Select one patient from the list (*DisplayButton.cs*)	Whole row is highlighted	✓	3:52
5.3	Click “Load” button	Panel collapses and teeth model is visible	✓	3:56

Procedure No.	**Input/Procedure**	**Expected Result/Output**	**Passed**	**Video Time**
5.4	Hover over different teeth on teeth model (*TeethScriptAssigner.cs*) (*TeethDisplayPanel.cs*)	Radiation values are displayed on the top left and changes depending on where the mouse pointer is at	✓	6:02
6	Clicking "Reset" button (*ResetTeeth.cs*)	When hovering over the teeth, no radiation values should appear on screen, indicating data has been removed from the model	✓	-
7	Loading many patients onto one teeth model + aggregate function (*DataModel.cs*)	—	—	—
7.1	Select the same patient from Procedure 5 and another random patient	Two rows are highlighted	✓	7:10
7.2	Click "Load" button	Panel collapses and teeth model is visible	✓	7:12
7.3	Hover over different teeth on teeth model	Should observe an increase in radiation values from Procedure 5 when hovering over the teeth.	✓	4:10
8	Comparing 2 sets of data using teeth model (*ComparePatients.cs*)	—	—	—
8.1	Click "Compare" button	Button displays "Select First Set"	✓	8:09
8.2	Select 1 or more patients from the list and click Select First Set button	Button displays "Select Second Set"	✓	8:19
8.3	Select 1 or more patients from the list and click Select Second Set button TeethScriptAssigner	Behind the filter panel, two teeth models are displayed	✓	8:40
9	Camera Lock + Cube Navigation (*ThirdPersonCamera.cs CameraNavigator.cs*)	—	—	—
9.1	Default setting is camera is locked + drag the navigation cube around	Both teeth are moving simultaneously in the same direction	✓	8:49
9.2	Unlock camera lock	Two navigation cubes are displayed. One for each teeth model so they are able to move independently of one another.	✓	9:04
10	Move zoom slider back and forth (*ThirdPersonCamera*)	Camera can zoom in and out of teeth model/s	✓	2:45

Procedure No.	Input/Procedure	Expected Result/Output	Passed	Video Time
11	Display Patient Details (*LoadPatient.cs*)	—	—	—
11.1	Click on “Patient Details” at the bottom left	Shows aggregated details of all patients in first set	✓	10:20
11.2	Click on “Patient Details” at the bottom right	Shows aggregated details of all patients in second set	✓	10:18
12	Visualise radiation using colours (*ColourCalculator, ColourPickerUI ColourPickerUIController, ColourSelect*)	—	—	—
12.1	Open “Colour Picker” panel by clicking the button with an arrow on the right hand side	Two input fields, seven colours, 4 buttons and information on the min and max radiation of each set of teeth are displayed	✓	5:02
12.2	Insert non-numeric characters inside the input field and click “Apply”	Panel collapses and “Invalid input!” is displayed on the screen.	✓	N/A
12.3	No patients are loaded onto the teeth model and you try to apply colours to the model	Panel collapses and “No patients selected!” is displayed on the screen	✓	N/A
12.4	One patient is loaded in and try to apply colours to second model	Error message displayed	—	—
12.5	Insert a number in both input fields and click “Apply”	Panel collapses and “Invalid range!” is displayed on the screen	—	N/A
12.6	Input in “From” field is larger then input in “To” field	Button displays “Select First Set”	✓	N/A
12.7	Input in “From” field is less then input in “To” field, 2 colours are selected and click “Apply” button	Colour within the gradient between the two colours chosen are applied to the teeth. On “Colour Picker” panel, gradient bar and the radiation range is displayed	✓	5:10
12.8	Input in “From” field is less then input in “To” field, 2 colours are selected and click “Apply Second Set” button	Colour within the gradient between the two colours chosen are applied to the second set of teeth. On “Colour Picker” panel, gradient bar and the radiation range is displayed	✓	9:56
12.9.1	Click “Reset colour map A”	Gradient bar on panel and colour on first teeth model disappears	✓	5:34
12.9.2	Click “Reset colour map B”	Gradient bar on panel and colour on second teeth model disappears	✓	N/A
13	Add CSV file (*CSVParser.cs*)	—	—	—

Procedure No.	Input/Procedure	Expected Result/Output	Passed	Video Time
13.1	Click “Add CSV File” on navbar	New window appears which is the file manager.	✓	3:18
13.2	Click on a non-CSV file	Unable to select file	✓	3:24
13.3	Select a CSV file and click upload button	—	✓	3:37
13.4	Click X on file manager window	Window disappears	✓	N/A
14	Creating graphs (*Graph, GraphData, GraphDataPoint, BarChartVisual, BoxPlotVisual, SceneTransition*)	—	—	—
14.1	Click “Graphs” on the navbar	Filter panel should appear beneath	✓	11:03
14.2	Under “X Axis”, select “Age” from the dropdown	Input field should appear on the panel	✓	11:14
14.3.1	Enter non-numeric characters in input field	Unable to enter non-numeric characters	✓	N/A
14.3.2	Enter a number ≤ 0	Displays error message		N/A
14.4	Click “Create Graph” when “Age” is selected for the x axis and the input field is empty	Displays error message	✓	N/A
14.5	Under “X Axis”, select any option bar “Age” from the dropdown	Input field disappears	✓	N/A
14.6	Click “Graphs” or “Cancel”	Filter panel closes	✓	N/A
14.7	Select “Nodal” as the X axis and filter the patients to show only N0, N1 and N2. Then click “Create Graph”	New scene appears, initially showing a bar chart that only consists of average radiation values for patients that have N0, N1 or N2. Y axis and bars have scaled correctly	✓	N/A
14.8	Click on “BoxPlotButton”	Bar chart is converted to a box plot which can also include outliers	✓	11:25
15	If graph has many data points, graph is scrollable. (X Axis variable is Patient)	Can scroll with your trackpad or using the the scrollbar	✓	N/A
15.1.1	Press + button near the tabs Click “Create Graph”	Filter panel is displayed New tab and graph are displayed on a different tab	✓	N/A

Procedure No.	**Input/Procedure**	**Expected Result/Output**	**Passed**	**Video Time**
15.1.2	Click “New Graph” button Click “Create Graph”	Filter panel is displayed New tab and graph are displayed on a different tab	✓	N/A
15.2	Click “New Graph” button. Click + to add a tab	Duplicate graph is created	✓	11:50
15.3	Click on previous opened tab	Transitions into the old graph	✓	12:09
15.4	Open 4 new tabs and click “New Graph” then “Create Graph”	Last tab is replaced with new graph	✓	N/A
15.5	Hovering over bars in bar graph	Tooltip displayed over the bar showing the exact radiation value.	✓	11:58
15.6	Click X on the top-right corner	Home scene is displayed, where you can see the teeth model	✓	12:35
16	Tutorial Tour *Dialogue, DialogueManager, DialogueTrigger*	—	—	12:44
16.1	Click “Help” on the navbar	A panel should appear with letters appearing one by one	-✓	13:09
16.2	Click “Continue”	New text is displayed onto the panel and should be invoking different actions across the application	✓	14:30
16.3	End of tutorial	Should consist of 23 steps	✓	14:43

References

- **Max-Min Slider**
  `https://github.com/brogan89/MinMaxSlider`
- **Graph Icons**
  `https://icons8.com/icon/15/bar-chart` Bar Chart icon by Icons8
  Box Plot Graph - Designed by Freepik from Flaticon
  `https://icons8.com/icon/1510/multiply` Multiply icon by Icons8
- **Basic Graph Structure**
  `https://unitycodemonkey.com/video.php?v=CmU5-v-v1Qo`
- **File Manager**
  `https://github.com/GracesGames/SimpleFileBrowser`
- **Teeth Model**
  `https://www.blender.org/`
- **The Whole Application**
  `hhttps://unity.com/`