University of Dublin

TRINITY COLLEGE

Interactive Android application
for children on healthy eating

Jennifer McGrane
B.A.I. Engineering
Final Year Project April 2014
Supervisor: Gavin Doherty

3
School of Computer Science and Statistics

O’Reilly Institute, Trinity College, Dublin 2, Ireland
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>3</td>
</tr>
<tr>
<td>Chapter 1</td>
<td></td>
</tr>
<tr>
<td>Game specifications</td>
<td>4</td>
</tr>
<tr>
<td>OpenGL Vs Game engine</td>
<td>5</td>
</tr>
<tr>
<td>Chapter 2</td>
<td></td>
</tr>
<tr>
<td>Which game engine</td>
<td>7</td>
</tr>
<tr>
<td>Chapter 3</td>
<td></td>
</tr>
<tr>
<td>Child nutrition</td>
<td>13</td>
</tr>
<tr>
<td>Communicating with children</td>
<td>14</td>
</tr>
<tr>
<td>Chapter 4</td>
<td></td>
</tr>
<tr>
<td>3D scene</td>
<td>15</td>
</tr>
<tr>
<td>Character</td>
<td>17</td>
</tr>
<tr>
<td>Models</td>
<td>20</td>
</tr>
<tr>
<td>Controls</td>
<td>22</td>
</tr>
<tr>
<td>Sounds &amp; voice overs</td>
<td>23</td>
</tr>
<tr>
<td>Dialogue</td>
<td>24</td>
</tr>
<tr>
<td>Chapter 5</td>
<td></td>
</tr>
<tr>
<td>Game logic</td>
<td>26</td>
</tr>
<tr>
<td>Chapter 6</td>
<td></td>
</tr>
<tr>
<td>Game outcome</td>
<td>31</td>
</tr>
<tr>
<td>Changes</td>
<td>32</td>
</tr>
<tr>
<td>Future Expandability</td>
<td>33</td>
</tr>
<tr>
<td>Resources</td>
<td>34</td>
</tr>
<tr>
<td>References</td>
<td>36</td>
</tr>
</tbody>
</table>
Introduction

From the outset of this project I wanted to create some form of an educational 3D game for children. The reason I wanted to do was is because having had a lot of experience with children over the last few years I had seen the 3D animation available to them through TV shows such as Dora the explorer or Mickey Mouse club house. However one thing I felt was lacking was a game similar to these TV shows. Shows such as this provide children with an interactive learning experience while at the same time telling them a story and keeping them entertained. This is what I wanted to create in my game. I wanted to create a game for those times when parents allow their child to play game on their phone or tablet, I thought “wouldn’t it be great if the game they were playing kept them entertained but at the same time taught them something of real value?” and that’s how the idea for my game, called Food Quest, was formed.

The topic I thought would be of the most benefit to young children in this game was healthy eating. The reason for this was because in a society where childhood obesity is an ever growing issue, educating children about the benefits of eating correctly from a young age is of vital importance.

To really understand what I needed to create in my game I researched the type of games already available. While there are some games on the market right now on the topic of healthy eating, such as Energy Balance by the British Nutrition Foundation and Solusville by Nourish Interactive, these games are only available to play online and are primarily an educational tool as opposed to a game. Seeing the types of games already out there helped me define exactly what I wanted my game to be and how I wanted it to be played.

The final decision I made was that I wanted the user to learn about healthy eating while walking around a 3D scene interacting with characters and carrying out tasks. In creating it this way the user is primarily playing a game but while also learning, in the same way that the TV shows I discussed earlier are primarily an entertaining show but all the while using educational subject matter to teach the viewer’s something.

In this report I will explain how I created this game and why I chose to create it this way. I will take you through the research I did on healthy eating for children and communicating with children and how I incorporated this into my game. I will also explain the game logic and every aspect of the game itself including the models, controls, characters, sounds and dialogue.
Chapter 1

Game Specifications

When I began planning how I would create my game there were three main specifications that I had:

1. I wanted the game to be 3D.
2. I wanted the game to be built for Android application.
3. I wanted to write the game in Java.

The reason I wanted the game to be 3D was because I wanted the user to feel immersed in the game, as if they were a character within it. The reasoning behind my desire to create the game as an Android application has been previously mentioned in my introduction, I felt that children would be more exposed to Android devices rather than computers and web applications. Finally having using both Java and C++ in the past to create 3D games I have found that Java gives a better result and is easier to understand and debug, and this is why I wanted it to be written in Java. Having laid out the foundations for the game I then had to decide where I was going to build it, there were two main options that were applicable to my project and these were, using the OpenGL library with Eclipse or Visual Studios and using a game engine. In the next section I will discuss the benefits and drawbacks of each and how I decided upon my choice.
OpenGL versus a game engine

The two options that I felt were most suitable for building my game were; using the OpenGL library in conjunction with Eclipse Android SDK or using a game engine. Both of these options were very applicable choices and each had their benefits and their drawbacks.

OpenGL

OpenGL is a cross-language, multi-platform software interface that is used to render 2D and 3D graphics. The API is typically used to interact with a Graphics Processing Unit to achieve hardware-accelerated rendering. OpenGL was created by Silicon Graphics Inc. and has been widely used over the years from its release in 1992 in; CAD, scientific visualisation, information visualisation, flight simulation and video games. The interface consists of about 150 distinct commands that are used to specify the objects and operations needed to produce the interactive three-dimensional applications. OpenGL’s multi-platform capabilities means it is hardware-independent and for this reason it contains no commands for performing windowing tasks or high-level commands for describing 3D models to create complicated shapes such as cars, parts of the body or airplanes. OpenGL requires you to build models from a small set of geometric primitives, i.e. points, lines and polygons. OpenGL is used as a library import in conjunction with programming software’s such as Visual Studios and Eclipse.

OpenGL has many benefits when using it to create games of either 2D or 3D nature, however it has just as many drawbacks, especially as applied to this game. The benefits of using it for this game are; firstly that it is 3D capable which is the most important specification I have, secondly that it can be written in Java, thirdly it can be used to develop Android applications and lastly, OpenGL allows you to have a high degree of graphics control because all models must be specified from points, lines and polygons. Although these benefits are conducive to creating a game such as Food Quest, there are major drawbacks of using it also.

Using OpenGL in games development requires major assembly from the ground up. Unlike other methods there are no inbuilt commands that can make creating an application a more streamlined project. All models, scenes and interactions must be completely coded and created from scratch, and although this allows the creator to have a high degree of control it also means that for a project such as this, with a time frame of 7 months it is not an ideal way of building a game. Using OpenGL for this project would either result in an unfinished game or a finished game of very poor quality, because this game is not just a showcase of graphics capabilities, it is also an important median to convey nutritional information I felt it would be important to be able to have some form of finished product to demonstrate the entire message and capabilities of my project. [1]
Game engine

The second option that I investigated for creating my game was using a game engine. A game engine is a system used for creating video games, they provide a software framework that can then be used to create games for multiple platforms. Typically game engines provide 2D or 3D graphics rendering, a physics engine, sound, animation, networking, memory management, threading and a scene graph. In some game engines scripting and collision detection are also provided. The key point to using a game engine is the ability to reuse or adapt the same game engine to create different games or the same games on multiple platforms. Most game engines are built on API’s like OpenGL, however some game engines provide an entire suite of visual development tools, and these allow the developer to rapidly develop a game on a range of platforms from computer applications to phone applications.

Within the game engine category there are game engines called middleware. These provide reusable software with all the core functions that are needed to develop a game application. Middleware game engine suites include facilities such as graphics, sound, physics and AI functions. As well as these middleware suites also provide platform extraction which allows the same game to be run on different platforms with small if any changes to the source code.

Game engines, like OpenGL have their benefits and their drawbacks. Their benefits are very similar to that of OpenGL; again they are 3D capable, they can be used using Java, they can be used to create Android applications and they provide a reasonable amount of graphics control. However game engines provide another and greater benefit than OpenGL. Game engines can be used to develop games of a higher standard in a lot less time. This is especially true for middleware game engines. Game engines allow fully animated and rigged models, included complex models, to be imported and controlled without creating them from points. This is a huge benefit and saves a lot of time, a developer can create or download models already of finished game quality and integrate them into the game in one call of a function. Like OpenGL, game engines have their drawbacks and a game engine’s biggest drawback is its limits. A developer is limited by the capabilities of their chosen game engine and must alter or tailor their game to suit it. However in the grand scheme of things this drawback is minor compared to the drawbacks of using OpenGL as it is.

Having researched and compared both of these choices it was clear that using a game engine was a much more suitable choice for what I wanted and needed. In the next chapter we will discuss the game engines I researched and why I chose the game engine I chose. [2]
Having decided that I wanted to use a game engine it was time to choose which one. There is a huge array of game engines on the market today, some open source, others you buy and each of them provides something different for the developer. The four key things that I was looking for when researching which game engine to use were:

1. That it was capable of creating a 3D game.
2. That it included some form of an Android development kit.
3. That it could be written in Java.
4. That the performance of the game engine was of high quality.

One final additional specification that I had, which I felt wasn’t completely necessary if all the other specifications were met was:

5. That I wanted the game engine to come with some form of documentation so that I could easily learn how to use the game engine and how to best make use of its capabilities.

Using these specifications I was able to filter through the plethora of game engines online and discovered the following four which I then researched in detail to ascertain if they were suitable for my game development, the four I found were; CatCake, AndEngine, Angle and JMonkey. All four of these game engines are open source, and in today’s world the game engines available for free download are just as good as some of the paid versions.

**CatCake**

CatCake is a 3D graphics engine, which is easy to use and has a high performance rate. CatCake is also cross-platform running on Windows, Linux and Android. This game engine also comes with a multitude of features for game development such as animation, mouse and keyboard handling and sound playing. As well as this CatCake provides collision detection, 3D model auto-generation from 2D images, font generation and motion playing. Fig 1.1 is a screenshot from a game created using CatCake. Using my specifications as a check list CatCake fulfils three out of four of them; it is 3D capable, it has an Android SDK and it has high performance, however it is not written in Java it is written in C++.[^3]
AndEngine

AndEngine is an OpenGL ES based, open source, Android game engine, it is similar to OpenGL in that it is a library project which you import into your programming workspace, i.e. Visual Studios or Eclipse. AndEngine is a 2D capable engine built specifically for Android, while the initial download only contains the basic libraries for creating a game there are free extensions that can be downloaded to give extra functionality to your game, such as the physics extension or the multi-touch extension, all of which are very useful to quickly build a 2D game for an Android device. AndEngine is written in Java, can be used to build an Android game and has a very high performance, but it is not 3D capable. AndEngine is also greatly lacking in useful documentation.\[4\]

Angle

Angle is a very similar game engine to AndEngine, like the former Angle is specifically designed for Android development and is also only 2D capable. It is also based upon OpenGL ES technology. This engine is entirely written in Java which means every object can be overloaded for the developer’s convenience. Unlike AndEngine, Angle does not have downloadable extensions to make it a better-rounded game engine. What Angle does have is a series of tutorials included with the engine to teach the developer how to use it.\[5\]
JMonkey

The final game engine of the four that I researched was JMonkey. JMonkey is a free open source engine that is based upon OpenGL ES technology and the framework is entirely written in Java. This engine is used to create 3D games following modern technology standards and can be used to develop games across a multitude of platforms such as desktop, web and mobile, because of this any games created within the engine require minimal adaptions to run on different platforms. JMonkey is a complete development environment, unlike some of the previous engines, it includes visual editors, integrated tools and unique plugins.

This engine is built around a shader based architecture and allows the developer to import game ready models and interact with them in the scene straight away. The key essence of this engine is that the developer is given all the essential tools straight out of the box while at the same time still remaining in control of the game.

Fig 2.2 shows the layout of the IDE and a 3D model being rendered to a scene. JMonkey, as well as having very clear and informative documentation included, also includes several tutorials and various models and scenes that can be used in the developer’s games. [6]
Which to choose

Having researched all four of the game engines above and compared their features with the specifications I had for this game, I then created a table, Fig. 1.3, to ascertain which game engine fulfilled my brief to the fullest.

<table>
<thead>
<tr>
<th></th>
<th>3D</th>
<th>Android</th>
<th>Java</th>
<th>High performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>CatCake</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>AndEngine</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Angle</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Jmonkey</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>

Fig. 2.3

As can be seen from Fig. 2.3, JMonkey was the only engine to fulfil every specification that I had and combined with its informative documentation, book and accompanying website it was a clear winner and the best fit for what I needed to create this game.
Chapter 3

Having decided that I was to use the game engine, JMonkey, it was then time to do some research into child nutrition and communication with children. The reason that I wanted to do this research as the next step as opposed to beginning to code the game, was because before I began creating it I wanted to ensure that I had all the knowledge on the topic so as create a game that fully showcased and conveyed this information.

Child nutrition

Nutrition for children like that of adults can be represented by a food pyramid. Each level represents a different food group, the size of these levels compared to each other is proportional to their amounts to be consumed each day. The food pyramid can be seen below in Fig 3.1.

![THE FOOD PYRAMID](image)

Fig 3.1 [7]
Carbohydrates

Starting at the bottom, at the largest level, we have carbohydrates. Carbohydrates consist of things such as cereals (rice, oats etc.), bread and potatoes. A child over the age of five should consume six or more servings of this level, where one serving equates to one slice of bread, one medium potato, one small bowl of breakfast cereal or three dessertspoons of cooked rice or pasta. At least one serving of this level should accompany every meal time. Foods like this that are high in carbohydrates which provide children with the energy they need to be active and play. [8]

Fruit and Vegetables

The next level up is the fruit and vegetables level. A child should consume five servings of fruit, vegetables or fruit juices every day, where one serving equates to one piece of fresh fruit (e.g. pear, banana), half a glass of fruit juice diluted in water, a small bowl of homemade soup, two tablespoons of cooked vegetables or three dessertspoonfuls of salad. This level is very important because fruit and vegetables are excellent sources of vitamins and minerals. For example oranges, strawberries and peppers are an excellent source of vitamin C which helps boost a child’s immune system to ward off colds and flus. Apples also help keep a child’s teeth and mouth clean and healthy. [8]

Dairy

The third level is the dairy level, this contains things such as milk, cheese and yoghurt and a child should consume 3 servings of this level, where one serving equates to one glass of milk, one carton of yoghurt or 25g of hard cheese. These foods are an excellent source of calcium, which is essential for bone and teeth development, especially in young children. [8]

Protein

Next we have the protein level, this level contains meat, fish, poultry, eggs and pulses. A child should consume two servings of these every day, where one serving equates to two eggs, two slices of meat or chicken, a medium fillet of fish or six tablespoons of beans, peas or lentils. These foods are a great source of protein which is essential for growth and development. As well as protein these foods also contain iron. Children have high iron requirements and eating red meat or chicken is an excellent way fulfilling it.
Eating meat or poultry can also help the body to absorb iron more readily from vegetables if eaten at the same meal. [8]

Sugar, fats and oils

Children of this age do not have very large appetites so it is very important not spoil their appetites for healthy food by giving them large quantities of sweets, biscuits, salty snacks or fizzy drinks. These foods should be limited to meal times. As well as these; fast foods, chips, nuggets, burgers and sausages should also only be given on rare occasions as they contain little nutritional benefits. [8]

Water

A child’s healthy diet is not only limited to the foods they consume. Another aspect that is vital to their health is drinking of plenty of water. The body needs water to carry out lots of jobs, for example, a person’s blood, which is made up of lots of water, carries oxygen to all our cells, without the oxygen they would die. Water also makes up part of our immune system which helps fight off illnesses and water is also needed to digest our food. For these reasons and many more water is vital to a healthy diet and should be the drink of choice with all meals. [9]
Communication with children

The second aspect of this game that I researched was the communication with the child. I felt that it was very important for me to research and understand the correct and appropriate manner to address a child in this game so that the game was as appropriate and well-constructed as possible. Under the topic of communicating with children there were two sub topics, firstly what to say to a child and secondly how to say it. [10]

What to say

When communicating with a child the most important thing to remember is to stay brief. The rule of thumb when speaking to children is to use the one-sentence rule. The longer you ramble when talking to a child the more likely it is that they will stop listening to you. Therefore if you must give additional information to a child the main information should be in the first sentence. Another thing to remember when communicating with a child is to stay simple and be positive. One should use short sentences consisting of one-syllable words and be positive in the way we construct these sentences. For example instead of saying “No running” we say “Inside we walk, outside you may run”. The final thing to remember is to speak correctly, this basically means to expect a child to be polite and to encourage this politeness by using words such as please and thank you when addressing them. [10]

How to say it

The final aspect of communicating with children, and from my research I have discovered that it is also the most important aspect, is how you talk to children. On average it is less likely that a child will hear what you say but will actually pay more attention to how you say it. Therefore in this game, where I plan to use voice overs to read aloud the dialogue between characters, it is vitally important that the tone of voice is correct. When speaking with children it is important that the tone conveys the correct message. If what you telling a child is exciting then the tone should coincide with that, the same is true if what you are asking is important or urgent. [10]
Chapter 4

Following my research into child nutrition and communication with children I then began creating my game in the JMonkey programming suite. In this chapter we will discuss the different elements that make up the game; how I created them, how I implemented them and how they are interacted with in the game.

The 3D scene

The 3D scene in this game is made up of two things, firstly the sky and secondly the model of a small town, a picture of the scene can be seen in fig 4.1. The sky is created by the function call:

```
SkyFactory.createSky(assetManager, "Scenes/Beach/FullSkiesSunset0068.dds", false)
```

This function calls the SkyFactory library, creates a dome around the entire scene and applies the desired sky image upon it. In this case the image is that of a sunset. By creating a sky it gives the illusion of an infinite scene, which adds to its realistic 3D nature. The entire town structure is made up of various building models which have had building exteriors placed on them as a texture. The flat ground that the user walks around on, like the buildings, has had a texture applied to it. It comprises of a grid like pattern of various textures such as grass, tarmac and pavement. The model town in this scene came included within the game assets and was created by loading the model and saving it as a scene node using the following code snippet:

```
assetManager.registerLocator("town.zip", ZipLocator.class);

Spatial scene = assetManager.loadModel("main.scene");
```

Before the scene can be added to the game we must add some light so that we can see it. To create a realistic scene we add some ambient lighting to replicate that of sunlight using:

```
AmbientLight al = new AmbientLight();
```

We then attach this lighting to the scene to illuminate it:

```
scene.addLight(al);
```
The final aspect of the scene is the trees within it, these trees are added to create a more realistic and complete town. These trees are created by loading the tree model included in the game engine assets:

\[
\text{Tree} = \text{assetManager.loadModel("Models/Tree/Tree.mesh.xml");}
\]

We then clone it the desired amount of times:

\[
\text{Tree2} = \text{Tree.clone();}
\]

Then we give each a location and scale and attach it to the scene:

\[
\text{Tree.setLocalTranslation(150f, -5f, 75f);} \\
\text{Tree.scale(5);} \\
\text{rootNode.attachChild(Tree);} \\
\]
### The Characters

Within this scene there are also three characters. These characters provide the basis for the game as the user interacts with them to carry out tasks. The characters used in this game are models that are already animated and rigged before they are imported into the game, because JMonkey allows models of this nature to be imported it decreases the time and amount of code needed to create realistic and interact-able characters.

The first model is that of a monkey, this character is the central character of the game and the user does the most interacting with it as he directs them to other characters to carry out the tasks necessary to complete the game. The other two models are that of a ninja and a green monster, these characters supply the user with tasks throughout the game and through these tasks the user learns about healthy eating. All of these models came as part of the game assets, like the town model. By using models that already exist within the game instead of creating them all myself, you can see how easy it is to incorporate models into the game and that given more time one could create character models oneself, using programs such as Blender, that are more aesthetically cohesive and replace the current models very easily. Each character in the game is imported using the same function calls.

Now we will look at how we create the animated character Milo the monkey in this game as an example. First we declare a spatial node for the character and then import and save the model as this node.

```java
private Spatial Jaime;
Jaime = assetManagerloadModel("Models/Jaime/Jaime.j3o");
```

Next we add a material to the model, currently the model is just a skeleton of the character, by adding a material it is like adding a skin to this skeleton.

```java
Material material = assetManagerloadModel("Models/Jaime/Jaime.j3m");
Jaime.setMaterial(material);
```

Finally the model is given a scale and location in the scene and then added to it.

```java
Jaime.setLocalScale(4f);
Jaime.setLocalTranslation(-25.0f, -5.0f, -50.0f);
rootNode.attachChild(Jaime);
```

To make the model visible in the scene we must again add lighting, this time of the directional variety.

```java
DirectionalLight sunny = new DirectionalLight();
sunny.setDirection(new Vector3f(-0.1f, -0.7f, -1.0f).normalizeLocal());
rootNode.addLight(sunny);
```
To add movement to the character we must create a control for the character, this is how the program will interact with the models available animations, and then we create a channel where instructions of which animations we wish to use are sent to the control.

```java
control2 = Jaime.getControl(AnimControl.class);
channel2 = control2.createChannel();
channel2.setAnim("Wave");
```

Other instructions that we can send through the channel to the control are things such as the speed of the animations and whether or not we wish the animation to run in a loop and only change when the next animation instruction is sent.

```java
channel2.setSpeed(1f);
channel2.setLoopMode(LoopMode.Loop);
```

The result of all of the above code can be seen in Fig 4.1.
The final aspect of the characters that we must create is the user interaction with them. What we wish to happen in this game is that the user may walk over the character and talk to them, where they are given their task which they then carry out and again return to talk to the character. The way we do this is by creating a volume, similar to an imaginary box, around the character. Another volume is then placed around the player. When these volumes collide we carry out the desired interaction.

```java
BoundingVolume MiloVol = Jaime.getWorldBound();
BoundingVolume pVol = playerModel.getWorldBound();
if (pVol.intersects(milo)){
    channel2.setAnim("Punches");
    if (miloquests == 0){
        guiNode.attachChild(p);
        milo1.play();
        guiNode.attachChild(Miloinitial1);
        guiNode.attachChild(Miloinitial2);
        guiNode.attachChild(Miloinitial3);
        rootNode.detachChild(cube1);
        rootNode.detachChild(fire);
        guiNode.attachChild(q);
        guiNode.attachChild(task1);
        rootNode.attachChild(cube2);
        rootNode.attachChild(fire14);
    }
}
```

The desired interaction in this case is a dialogue. The dialogue that occurs changes depending on the value assigned to the variable `miloquests`. In the example above if `miloquests` is equal to 0 this means that this is the first interaction with Milo and as such the dialogue that appears on screen is the initial dialogue providing the user with their first task. Other things that occur in this step are: that a voice over is played and the spinning cube and particle emitter that previously appeared above Milo’s head, indicating that he wanted to talk to the user, are removed. When the user completes their task these items will appear above Milo’s head again. As the user completes their tasks `miloquests` is assigned increasing values, in this way when the user returns to talk to Milo the appropriate dialogue occurs.

The above character implementation and interaction is almost completely the same for each of the models. The only difference between Milo and the other two models is the dialogue interaction, the volume collision detection is used however instead of the dialogue appearing straight away, an instruction to press A or D appears instead and upon pressing the instructed button, depending on the users progress in their tasks, the appropriate dialogue appears.
Models

Within this game there are thirteen 3D models. These models are of various fruits, vegetables, carbohydrates, dairy products, proteins and drinks that the user collects during their tasks. These models like those of the characters are loaded in the same manner. We will look at the apple model as an example. First we declare a spatial node, load the apple 3D model and save it to this variable.

```plaintext
Private Spatial apple;
apple = assetManager.loadModel("Models/Apple/Apple.j3o");
```

Again we add a material to the skeleton of the model.

```plaintext
Material appleskin = assetManager.loadMaterial("Materials/newMaterial.j3m");
apple.setMaterial(appleskin);
```

Most of these models were downloaded from various open source websites, some of them were then edited in Blender to make them more appropriate for the game and one of the models was also created by myself using Blender, this was the broccoli model, this showcases that models from all different sources can be imported and implemented into the game using the same functions.

The models in this game, unlike the characters, are allocated a zone within the scene and their location within this zone is then randomly generated.

```plaintext
apple.setLocalTranslation(x1, 0.5f, z1), Where x1 and z1 are randomly generated variables.
```

As well as a location each model is rotated around its Y axis. Every model also has a particle emitter connected to it, this looks similar to a magic spell or sparkler, it makes the items stand out in the scene and adds something extra to collecting them. The models are collected by using the same collision detection technique as the character models. A volume is placed around each model and when the player volume collides with the item’s volume a chime noise plays, the item is removed from the scene, a dialogue appears on screen, a voice over reading the dialogue is played and the users collected items counter increases by one.

```plaintext
if (pVol.intersects(aVol)){
    rootNode.detachChild(apple);
    rootNode.detachChild(fire1);
    guiNode.attachChild(apple_n);
    guiNode.attachChild(p);
totes1 =1;
ap.play();
boom.play();
}
```
Below in Fig 4.2 you can see the apple model loaded into the scene, as well as two other models in the distance.
Controls

The controls of this game are relatively simple to use, although their implementation
is slightly more complex. The user moves forward through the scene using the space bar.
This is done by taking the player position and current camera direction and when the user
presses the space bar the player node and the camera position move forward in the
direction the camera is pointing. The spacebar also acts to remove dialogues from the
screen when the user has finished reading them and wishes to begin their tasks.

```
else if (START=true & binding.equals("Forward")){
    Vector3f cc = cam.getDirection();
    cam.setLocation(player.getLocalTranslation().add(0, 0, 0));
    player.move(((cc.x)/7), 0,((cc.z)/7));
}
```

To change direction the user uses the mouse and the left button to click and drag the
camera in the direction they wish to go. This updates the camera direction and position and
when the user next presses the space bar they move in this new direction.

The final control that the user has is the right mouse button. This button is used
when the user has collected all their items and returned them to the. The character then
asks the user a question regarding the items and nutritional information they have gathered
during their task. The character asks them to choose which of two items provides a certain
nutritional benefit. The user then right clicks on the item they think is correct. If the item
they click is correct then the character tells them so and the opposite if the item the chose is
incorrect.
Another aspect of the game that creates a well-rounded experience is the sounds and voice overs that occur throughout the game. The first sound that the user hears is the soundtrack that plays in the background during the entire game. This sound is an upbeat melody that creates a fun atmosphere to add to the game. This sound is a royalty free melody that was downloaded from an open source website. Sounds are added to the game by first creating a variable of the type Audionode. The melody is then imported and saved to this variable. Once the sound has been imported it can then be altered in many ways, such as volume, position, tempo and whether the sound repeats constantly on a loop.

```javascript
nature = new AudioNode(assetManager, "Sounds/Call to Adventure.ogg", true);
nature.setPositional(false);
nature.setVolume(0.1f);
nature.setLooping(false);
rootNode.attachChild(nature);
nature.play();
```

Another sound that occurs during the game is; footsteps, which plays every time the user presses the space bar to move around the game. This gives the illusion that the user is a character in the game and the footsteps are their own. The final sound that plays is the chime noise that occurs when a user collects an item during their tasks. These sounds are paused and played at various times of the game.

Something else that the user hears during the game is the voice overs. These voice overs occur when the user interacts with a character or collects items. Having voice overs in the game means that users of all reading levels can play the game, as the dialogue both appears on screen and is read aloud. The voice overs were recorded using a software called Audacity, here one can record and alter sounds to suit their purpose. The audio is then exported to the project folder in the .ogg format. These sounds are then imported and used in the same manner as the previous sounds. The voice overs are played at the same time that the dialogue between the user and characters are triggered or when the user collects an item and nutritional information appears on screen. The voices used in the voice overs, which can be heard throughout the game, are that of myself and my family.
Dialogue

The dialogue that appears on screen is of two varieties. Firstly there is the character interaction dialogue and secondly there is the item interaction dialogue. Both of these are created in the same manner. First like everything else we have discussed so far we declare a variable for each dialogue, this variable is a BitmapText, and next we can then assign a font and a colour. Next we load the text we wish to appear.

```java
private BitmapText dialogueText;
dialogueText = new BitmapText(comicsans, false);
dialogueText.setColor(ColorRGBA.Black);
loadText(dialogueText, "I've to dropped all my food for the party, can you help me find them? ", comicsans, 5,5,0);
```

When we wish to have this dialogue appear on screen we call the following function:

```java
guiNode.attachChild(dialogueText);
```

To make the writing readable we place a picture behind it, in this case a picture of a textbox, this makes the dialogue visually appealing and creates a more polished finish to the game. This picture is called every time we attach a dialogue to the screen and is removed when the dialogue is removed. An example of how the dialogue appears can be seen in Fig 4.3.

```java
p = new Picture("Picture");
p.move(0, 0, -2); // make it appear behind dialogue
p.setPosition(0, 0);
p.setWidth(settings.getWidth());
p.setHeight((settings.getHeight())/4);
p.setImage(assetManager, "Interface/textbox.jpg", false);
```
Milo the monkey looks like he has a question for us, let's go talk to him!
Game logic

In this chapter we will discuss the game layout from start to finish. Food Quest begins with the start screen, below in Fig 5.1 we can see the layout of the start screen, it is made up of a background image, a short summary of the back story to the game and two buttons, one to start the game and one to quit it. This start screen is created from three files, two .java and one .xml. The XML file and the Java file of the same name specify the layout of the start screen. These files split the screen up into layers and panels where you can then place text, images or buttons. These files are then called by the main program;

```java
NiftyJmeDisplay niftyDisplay = new NiftyJmeDisplay(
    assetManager, inputManager, audioRenderer, guiViewport);
Nifty nifty = niftyDisplay.getNifty();
guiViewport.addProcessor(niftyDisplay);
nifty.fromXml("Interface/screen3.xml", "start", startScreen);
```

MyStartScreen, the second java file, tells the program what to do when the user presses one of the buttons, this file is called by the game by the below function call.

```java
startScreen = new MyStartScreen();
stateManager.attach(startScreen);
```
After the user has selected to start the game they appear in the town. The user plays this game from a first person perspective, this immerses the user in the game and allows them to view the town as a character within it. Next the user is directed to Milo the monkey who has a task for them. Upon speaking to him they are asked to go and talk to Mike the monster. In the top left corner a text box appears, as can be seen in Fig 5.2, this is our current task.
Now they talk to Mike monster who asks them to collect all the food he has dropped on his way to the party. Once the user interacts with Mike six items appear in the scene which the user must collect. The current task box has changed in the top left corner to a tally of how many of the six items we have collected, as can be seen in Fig 5.3.

![Fig 5.3](image)

As the user moves about the town collecting the items the current collected items increase, and nutritional information appears on screen and is read aloud. As can be seen in Fig 5.4.

![Fig 5.4](image)

Once the user has collected all the items they must return to Mike monster. Here they are asked a question about the information they have learnt while collecting all the items, as can be seen in Fig 5.5.

The character question interaction, visible in Fig 5.5, is created in the following way. First we create a new node called clickables. This is what we will attach the items to for the question. Next we create a new action listener. This runs in the background waiting for the user to right click on the screen. When the user does right click on the screen the listener takes the on screen location of the click and estimates where in the 3D environment the user has clicked and stores it. Next the program takes this result and checks if this location coincides with a clickable object. If it does result in a clickable object we set a certain boolean variable to true, depending on which item the user has clicked on. The program then uses this boolean variable to know which dialogue to place on screen, the correct or incorrect one.

```java
if (name.equals("Click") && !keyPressed) {
    results = new CollisionResults();
}```
Ray ray = new Ray(cam.getLocation(), cam.getDirection());
clickables.collideWith(ray, results);
for(int i = 0; i < results.size(); i++) {
    hit = results.getCollision(i).getGeometry().getName();
    if(results.size() > 0) {
        if(hit.equals("Apple1")) {
            ninja = true;
        }
    }
}

To display the items we wish to use for the questions, we attach the desired items to the node clickables when we wish to display them.

    clickables.attachChild(apple);
    clickables.attachChild(Banana_3d);

We translate the objects so that they appear in front of the character and are both visible to the user. After the user has answered the question the items are then removed from the node.

    if (ninja == true) {
        guiNode.attachChild(p);
        guiNode.detachChild(levelComplete);
        guiNode.attachChild(correct);
        clickables.detachChild(apple);
        clickables.detachChild(Banana_3d);
        bobDia3.play();
        ninja = false;
    }
    ninja = false;

Fig 5.5
Having answered the question the user is directed back to Milo who then sends them to another character and the process repeats itself as before but with different items to collect.
Chapter 6

Game outcome

At the beginning of this project I set a goal for myself of creating a game along the same lines as shows such as Dora the explorer and Mickey Mouse club house, having found that none really existed on the market. I personally believe that I have succeeded in creating a game that forms a great basis for a game such as that. The 3D nature of Food Quest means the user feels that they are part of the game, as if they are a character within it. This means that their interactions with characters and items around the scene are more real to them and in turn more enjoyable.

By creating a story around the game, and including the user in the progress of this story, the educational aspect is well built into the tasks and the user subconsciously picks up the nutritional information. As well as this, because the user must recall the information they have learned around the game, the user is made much more aware of the nutritional benefits of the items in question.

The inclusion of both voice overs and dialogue on screen means that the game can be played by users of all ages and all capabilities. Users with reading difficulties or users with hearing difficulties both have an alternative way of learning about the tasks or nutritional benefits throughout the game. This makes the game very appealing to children of all ages and to parents of children with these difficulties.

The nutritional information in this game is given in a clear and simple manner that is very easy for children to understand. By learning about the nutritional benefits of the foods in this game and how to communicate properly with children I was able to word the dialogues, questions and nutritional information in an appropriate manner that children can apply to their own lives. Such as, the nutritional benefits of eating oranges, which boosts ones immune system due to its high vitamin C content, I wrote this as “full of vitamin C to help keep the sniffles away”. By using language like this a child can equate words that their own parents may use, with the ones in this game and be able to associate the foods and their benefits with them.
Changes

If I had more time or could do the project over again there are a few changes I would make. The first major change I would make would be to get the game to work on an Android device. To do this there are only a few minor changes to be made in the project and these are mainly regarding the controls. At the moment the user uses their keyboard and mouse and with an Android device it is necessary to use the touch facilities.

The next thing that I would change relates to the characters in the game. At the moment because the characters are imported from the game engine assets they are not cohesive or consistent with the story. In order to do this I would like to, to either create my own models to the exact specifications I desire or to find better or more cohesive models from open source websites, which I found to be very difficult.

As well as changing the characters, I would change the items that the user collects as they move about the game. The backstory to this game is that the characters are going to a party but have lost their food and need the user to help them find them. To create a more uniform story, perhaps the models that the user collects should be items that one might eat at a party, such as sandwiches. Then when the user collects a sandwich they can be informed of the nutritional benefits of the bread and the filling within it.

The final thing that I would change would be some of the language that I use in the game. There are times where the user is collecting items around the scene when the word yummy is repeatedly heard. This can be quite annoying and this is definitely something I would change, by perhaps using words of the same meaning such as delicious or wonderful or sometimes leaving it out.
Future expansion

The final area of this project that I wish to discuss is the future expandability of it. As I have previously said this game is a very good basis for future development. If I was to expand this game or recommend it as a future project for someone, there a few ideas that I have about how this game could be expanded.

Firstly and most obviously there is the inclusions of more levels, characters, items and questions of a similar nature to ones already in the game. This would add more depth and education to the game. As well as that each level could occur in a different scene to add more life and variety to the game, such as one level being at a circus or an amusement park and another at the beach. Here the user can collect foods that they might find at places like this and learn about their nutritional benefits. Another way to add more education to the game could be to could create the entire game based on the food pyramid where every level is a different level of it. This means that the foods to be collected are ones that we eat from that particular level of the food pyramid.

Another aspect to the game that could be introduced is the idea of additional collectible items that can be used in mini games to win extras for the game. Such extras could be things like a car that the user can drive around the scene in collecting items instead of walking. Education could be added to the mini games by using them as a tool to show where food items come from, for example one mini game could be to milk a cow or pick all the carrots before the time runs out.

The final idea that I had for expanding this game would be by introducing other topics as well as nutrition, such as oral hygiene or exercise. By including other topics into the game you could create a great all round educational game for children.

In conclusion, Food Quest serves as a basic version of a game that provides educational information to children in a fun manner and I feel that the resulting game is very effective.
Resources

Online resources

www.foodafactoflife.org
www.nourishinteractive.com
www.glprogramming.com/red
www.gamecareerguide.com
www.code.google.com/p/catcake
www.code.google.com/p/angle
www.mobilegameengines.com
www.Androidtechworld.com
www.hub.jmonkeyengine.com
www.fooddudes.ie
www.bordbia.ie/aboutfood/nutrition/pages/nutritionfortwotofives.aspx
www.healthpromotion.ie
http://kidshealth.org/kid/stay_healthy/food/pyramid.html
www.askdrsears.com/topics/parenting
www.hub.jmonkeyengine.org/wiki/doku.php/jme3

Book resources

Between parent and child by Dr. Haim G. Ginott
Nutrition through the life cycle by Judith E. Brown
JMonkey Engine 3.0 Beginner’s Guide by Ruth Kusterer
Models Resources

www.turbosquid.com
www.tf3dm.com
www.blender-models.com
www.cgtrader.com
www.artistx.org
www.3dmodelfree.com
www.animium.com
www.jmonkey.org

Sounds resources

www.incompetech.com
www.jmonkey.org
References

DECLARATION

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university

________________________________________  ________________________________
Name                                      Date