Technical Documentation

Apple Adventures

2013

Inflatable Srawberries

12/28/2013

Contents

[Game Overview 3](#_Toc388864007)

[Game Architecture 3](#_Toc388864008)

[Development Schedule 4](#_Toc388864009)

[Initial Plan 4](#_Toc388864010)

[Actual Plan 5](#_Toc388864011)

[Game Media 5](#_Toc388864012)

# Game Overview

Our Final game is a simple (G rated) collect items game. It involves the player collecting apples and small tasks for the villages around the map. The world itself is fairly large and contains a few main areas. First and most plentiful are the villages there are three found in all corners of the map. Each town has a small task for you to complete before you face the final final boss.

# Game Architecture

Because we used a pre-built engine most of the underlying code and design were abstracted from us (the users). We can however talk about how we used the engine in brief and how the scripts and objects interacted.

As a team we tried to have one major script (maximum) with many smaller scripts attached to each object. That large script would call and set most of the other supporting scripts allowing them to be handled mostly in one place. However as the game progressed the line between the scripts had started to blur. This was because changes had the be made to the scripts as we found more problem.

One of the larger complaints with unity and with very dirty solutions is having multiple triggers on one object. For our wolf model we needed multiple triggers one to check if the player was close enough to be detected. The other was if the player was close enough to be attacked.   
As mentioned above the problem was the if we had two collision objects on a single model, there is currently no way to differentiate between the different triggers. The solution was to create another game object and make it a child class of the wolf model,

Then refer to the parent class from the script. This caused a few problems, firstly the only way to refer to the parent was by having a tag. This means that every instance of the wolf object would alter all parents with that tag. To solve this took some time and research. Thanks to the extensive documentation kept by unity we managed to find a function that allowed us to get children objects. So with a bit of re-factoring of different parts of the game and scripts the program was fixed.

# Development Schedule

### Initial Plan

|  |  |  |
| --- | --- | --- |
| Week | Planned Tasks | Description |
| 1 | Talk about the game we want to create | Discuss with group members what kind of game we want to make this semester. |
| 2 | Choose tools | Look into the types of tools we could use to accomplish what we had chosen. |
| 3 | Selection, compromise and starting | Select a tool and decide the compromises that might need to take place as a result of our selection. Then start on the project |
| 4 | Every member should learn a certain part of the tool. | As a result of ther units ramping up around this time as well, let everyone have a bit more time but they should still put some effort into their respective roles |
| 5 | More playing with tools | Everyone should try and learn how to use the selected tool so they can be become proficient and don't need others in order to accomplish their task. |
| 6 | Planning | Plan what sortstheirthings our game needs, in terms of assets and programming/scripting on our end |
| 7 | Search for assets | Look for assets and any other tool we may use in (eg the auto-desk FBX exporter for images and models built into one) |
| 8 | Slow down a bit | Other assignments and things are coming up all together work on them |
| 9 | Assign an overall role for each member of the group (eg one for NPC's one for Camera movement) | Give a topic to each person so they can focus on a particular area as well as lessen the work load for the entire group. |
| 10 | Start modifying and building the base of the game |  |
| 11 | Work | Work on the project have gatherings from time to time in order to check on peoples progress |
| 12 | Finish the work and last minute polish | Finish the work we have been assigned and do any touch ups and last minute items that are needed we shall add and help anyone who is in need to help |
| 13 | Testing and documentation |  |

### Actual Plan

|  |  |  |
| --- | --- | --- |
| Week | Planned Tasks | Accomplished? |
| 1 | - | - |
| 2 | Meet up | Talked about what we were going to accomplish |
| 3 | Meet up | Cut down on what we wanted to do |
| 4 | - | - |
| 5 | Research XNA and unity | Still trying to decide which of the two were a better choice |
| 6 | Researched unity (out chosen development tool | Found some cool things we could add to the game |
| 7 | - | - |
| 8 | - | - |
| 9 | Did the unity tutorial | Accomplished with great success |
| 10 | - | - |
| 11 | Looked for assets | Assets found, have not settled on which we are using |
| 12 | Repairing the scene and basic functions | Success |
| 13 | finish scripts and merging our projects | Success but with issues |

# Game Media

Our game was made in unity which provides a simple but powerful user interface for developing games. The unity package also allows developers to obtain models from a large selection of assets provided by the community. There are both free and paid assets online which can be directly downloaded into your assets folder.

Our world consisted of a large terrain generated by us through blender. Trees, bushes, rocks and buildings were provided through varies free online asset packs. They were all then opened together in unity.