**ASSIGNMENT**

**COVER SHEET**

This sheet must be completed as required and attached to the front of each assignment which you are required to submit.

|  |  |
| --- | --- |
| **Student ID** | FENJPD1201 |
| **Student Name** | Justin Fenech |
| **Unit of Study** | SIT153 – Introduction to Game Programming |
| **Assignment Title** | Game Project |
| **Lecturer’s Name** | Daniel Lowe |
| **Class Time / Day** | 11:00 PM / Friday |
| **Assignment Due Date** | 01/06/2012 |

***N.B.*** *Your submitted work should either be your own or, if not, your source(s) should be acknowledged. Submitting written work which is copied or paraphrased from other authors (including unpublished work from other students), without correct acknowledgment is considered plagiarism and is likely to result in penalties being imposed.*

**Student Declaration**

The material contained in this assignment is my own work unless otherwise acknowledged.

Student Name: Justin Fenech Date: 01/06/2012

*By typing your name here, you indicate agreement with the Student Declaration. Your hard copy requires a signature.*

Contents

[Part B: Project Report 3](#_Toc326351833)

[Software Architecture 3](#_Toc326351834)

[Development Log 3](#_Toc326351835)

[Concept 3](#_Toc326351836)

[First Compile 4](#_Toc326351837)

[First Classes 4](#_Toc326351838)

[More Maps 5](#_Toc326351839)

[Interface 5](#_Toc326351840)

[Battle the Monsters 5](#_Toc326351841)

[Context Sensitivity 6](#_Toc326351842)

[More characters and More Errors 6](#_Toc326351843)

[Random isn’t that random 6](#_Toc326351844)

[Fusion 7](#_Toc326351845)

[Tooltips for dummies 7](#_Toc326351846)

[Balancing 7](#_Toc326351847)

[Future Plans 7](#_Toc326351848)

[Game Walkthrough 8](#_Toc326351849)

# Part B: Project Report

## Software Architecture

The core of the program is in the main.cpp. This file is used to initialise our global variables, data and hold our main game loop. The game loop handles simple statements and relies heavily on the classes for most of the functionality. Adventure Command consists of multiple base and derived classes that interact with each other to create and perform functions for Items, Characters, Interface, Console window, Inventory, etc.

The **Character** class contains all of the definitions and functions for players and monsters, including a derived NPC class.

**Item** handles all of the loot items in the game.

**Interface** handles all of the visual elements besides the map.

**Window** handles all of the functions relating to the windows console.

**Objects** is the base class for waypoints and triggers (events)

For each of our key objects such as Characters, items and monsters, a global multidimensional array, with a vector that holds a pointer to class objects… is defined. The array container allows us to refer directly to a map or zone easily. For example **array[map][X][Y]**. Storing a vector inside this array allows us to then push and pop the vector with the pointers to objects when they are created or deleted. Most of the functionality of the game is designed around the data in these containers.

## Development Log

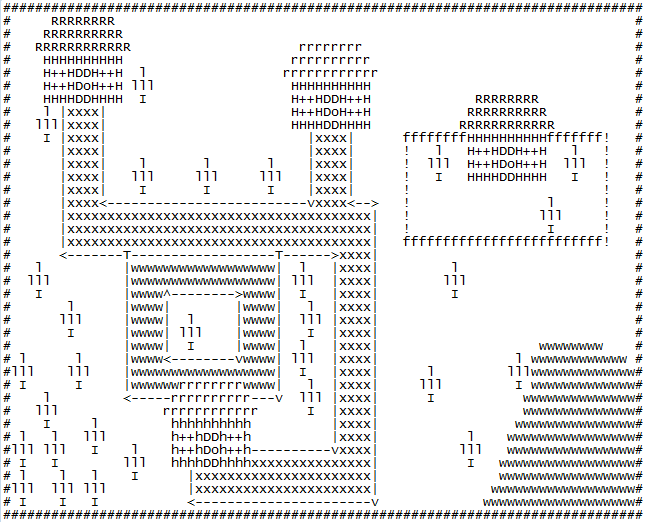
##### Concept

The approach initially is to start drawing a map to get a feel for the game world. This will help paint a picture of where the items, characters, and monsters will likely be placed. As well as provide a wireframe for when the Map is drawn in ASCII.



*Concept of town map*

Once the concept was figured out it initially seemed convenient to store each map in an array of const chars and then output each line in white text to the console, giving the player a visual representation of a town. This was not practical as each character in the array would need to be converted to an ASCII character. This was not only time consuming but virtually impossible to keep track of.



*ASCII Town*

##### First Compile

After a few for loops were created the console was outputting something similar to the above picture and was not pleasant on the eye, leading to the creation of a map editor which could create and edit maps with colour. The maps are now stored in a file with individual Unicode characters for the Attributes and Characters

##### First Classes

A Window class was created to handle the size and modification of the windows console.

The Map class was created to handle the saving and loading of the map files. Once there was a working prototype of the map we needed a character. A simple character class was created that stored health, defence, attack, dexterity, and player location. Being a novice coder everything’s made public for testing purposes… It’s still public.

The move method has been created for the player to handle movement. Using CHAR\_INFO and WriteConsoleOutput the character can be outputted to the correct location when the movement keys are pressed. The initial approach was to peek at the input. This caused an issue as there was a repeat delay set by windows. In order to avoid changing the key repeat delay on a user’s system the game now uses GetAsyncKeyState to detect if a key is pressed.

A strange error would appear when the character would walk from right to left, which is still present in the program. It turns out the console doesn’t like going backwards and there is a 1 pixel overlap when a character is placed. Because the right character usually gets placed on top of the left character doing the opposite it was causing a problem.

Storing the old tile background colour was also a challenge, and since we needed to restore the tile when the player has walked across it, it was mandatory. This was resolved by using bit shift and checking each colour against the CHAR\_INFO attributes.

##### More Maps

More maps have been created using the map editor and the character can now run between the maps. The function checks if there are a new map automatically and loads one if it exists. Otherwise it appears as if you hit a dead end.

Check out the full map so far here: http://www.deakin.edu.au/~jfenech/FullMap.png

Collision detection is now completely functional and compares the number value of the ASCII character to a pre-defined array. If the character is “Walkable it lets the player move”.  
New walkable tiles are being added constantly but the array makes it easier.

One of the problems with initially travelling between maps was that the “tile” on the new map must be walkable or else the game would not display the character. This was rectified by lining up all the zones of each map correctly.

##### Interface

At this stage we have a simple game that allows the player to move so it was about time to create a simple title menu. It displays 3 options

1. Play Game
2. Show Help
3. Quit Game

Next the inventory was created and so far only displays a simple vector of pointers to item objects. The inventory displays in the centre of the screen.

An Item object was created. One of the bugs that was quite annoying, is when we we’re trying to remove an item from the center of the vector, the for loop was iterating over each item so when it deleted item 5… 6 became 5 so we skipped it when displaying the items.

##### Battle the Monsters

So we have a movable character and loot, but nothing to do.

This is when the monstrous multidimensional array of vectors with pointers to character objects come in… It’s a mouthful, but it seemed like the easiest solution. Accessing all the maps and zons to create and delete monsters was a lot easier for each zone.

The array is created so now we need to have a battle mechanic. The interface was designed using the map editor and displaying the stats became part of our Interface class, the character class is used to handle the battle and update the stats. The most difficult part of this was figuring out how each stat was going to affect the battle. This calculation was handled in an excel spreadsheet.

##### Context Sensitivity

A function to handle context sensitivity was create to detect if the player was near anything when the action key is pressed. If near a monster it would enter the battle screen. Another multidimensional array was created to store the loot and was included in this function.

A feature was implemented to enable the player to drink water. This works by getting attributes of the surrounding tiles. However if the player had the inventory open, and the player was behind it, all the surrounding tiles would be colours in the inventory... not what’s “behind” it.  
The error was resolved by having the inventory change sides depending on the player position.

##### More characters and More Errors

More character and quests have been introduced using the existing functionality of the Character class. The NPC pointers are stored in another multidimensional array.

Some of these quests require and reward players with items or even inventory resizing.

A bug in the code was unintentionally created when the player would receive or loot items. A quick explanation of the problem was that a for loop was using “i” to iterate through an array. Then inside that another for loop defined “i”. However inside the nested loop there was still a function using the variable which was replaced. Of course this still compiled and is a great example of a logical error. This bug took more than an hour to find.

Our new class Objects was created purely to handle the creation of Objects, this base class has 2 derived classes, Waypoints and Triggers. Waypoints are invisible on the map and transport the player to another location when stepped on. This was created so that when we enter a cave we can load a new map. Triggers can be visible or invisible and are triggered when an item is used near them. If we use the correct item near a trigger it will trigger an event.

Plenty of bugs have been encountered here and one of the most destructive was caused by the fact that I had created new maps. Inserting new pointers into the array caused vector out of range errors. This error was puzzling and stepping through the application didn’t even help. I then discovered that the initial array I had declared was too small in size to handle the new map locations, causing a vector out of range error.

##### Random isn’t that random

With the core game content well on its way it was time to start implementing random loot and some random elements into the battles. Although the traditional rand() function was not that random. So in attempt to get a more random number randlux (from <random>) was used... this made it worse and for some reason would return the exact same value every time the player looted an item. Other more experienced programmers were also baffled. The solution was to re-seed the generator every time it was called. This caused an additional problem when a function was created to generate more than on random number consecutively. I decided to return to the original rand() function but instead of a random up to 100, I have divided it by RAND\_MAX and calculated the defined range from that.

##### Fusion

When testing the game for the millionth time it occurred to me that you would often pick up weapons with the same name. One of them is useless and at this stage cannot be sold. Thus Fusion was created. A simple modification of the “stack item” code (when looting), was changed so that weapons could now fuse together and create a stronger weapon with a new name.

##### Tooltips for dummies

As the creator it’s easy to forget that other players don’t automatically know what to do, what keys to press, what items do what and so on. So it was at this stage that there needed to be something to at least tell the player what each item does. Inside the game loop a new feature was added so that the player could press the “alt” key to display tooltips when in the inventory. This proved a slight hindrance as the dynamically moving inventory meant that we also needed to dynamically move the tooltips. In the end it wasn’t much of a difficult task be during crunch time your brain can get a bit fuzzy.

##### Balancing

The game is not entirely balance and there is still a lot more to do. The game at this stage still needs some tweaking. As far as I know all the bugs have been squished and the result is hopefully a stable game.

##### Future Plans

This game is far from over. Here are some of the features planned for future beta tests;

* Monster Re-spawn
* Tooltips for the world (ground loot and NPC names)
* Save/Load game feature
* More of a story rather than just a bunch of quests
* Creating more challenges and puzzles
* Expanding the map
* Balance the weapons and include armour

## Game Walkthrough

Feel free to talk to all of the NPC’s as you travel along the way..

Travel north and defeat all the frogs first and then kill the toads.

Travel West and defeat the bear there is a hidden to the left which leads behind the cave..

Collect the items and enter the cave. Follow the cave around until you meet craver.

You can find the chisel by heading to the far left and then go down following the path.

Once you return the chisel to craver he will provide you with a pick axe.

Exit by going east and head back to town.

From here travel east and you will find some boulders, use the pick axe by opening your inventory and pressing the appropriate key.

Continue east and you will find 2 NPC’s and some dynamite (red i) pick up the dynamite and use it in front of the dragon. The cave will explode.

Pickup the loot and return the supplies to the 2 NPC’s.

With the dynamite head to any water and use them to obtain a fish..

Keep heading west until you find an NPC called Wakefield, trade him the fish for his yeah.

Now continue to where you got the dynamite and use it to open the gates to the maze.

Once inside the maze follow it until you reach the exit.

Talk to Patchie who will tell you about his brother and that he needs a skybeak.

There is a hidden path in between the maze and the water on this same screen where you can find the Skyhawk. Kill it to retrieve the item for Patchie..

Take the compass that you receive into the maze again to find Quinto. Quinto will reward you with potions.

\Congratulations you have reached the end of the demo