CS 368 Final Project Report

A Basic Spaceman Game

Jeffery Tucker (jtucker@cs.wisc.edu, 907-124-8893)

Alec Yu (ayu@cs.wisc.edu, 907-075-3653)

Computer Science 368: Learning C++

Department of Computer Sciences

University of Wisconsin-Madison

December 20, 2016

1 Introduction

.

1.1 Motivation

When we were debating on what we should do for this project, we wanted use our project as a learning experience instead of doing something we are familiar with. We considered many ideas but ultimately decided on constructing a game. As this is a class about learning C++, we decided that we should use this project to learn C++ in a fun way; and what better way to have fun than through a game. Games cover a wide spectrum of programming aspects. It includes graphics, user interface, and the use of hierarchies. Overall, we find constructing a game is a interesting and practical way of learning C++. We considered that this sci-fi Mario style game was the best way to use our abilities as other classic games (such as snake and space invaders) was too easy and building a 3-D first-person view type game was too difficult. The benefit of this type of game is that there is great versatility to what we can do. There are many different features that can be included in future development.

1.2 Overview

We have created a basic spaceman game where you can play as an wandering spaceman and explore your environment as well as interact with the alien life present on the planet. The game is in the form of a 2-D platformer (the style used in 64-bit Mario.) Currently, our game is still in development; however, the main portion of our system works. Our system, in its current form, allows a user to control our spaceman character through the keyboard. Pressing down on the RIGHT arrow (or the D key) will move the character right; pressing the LEFT arrow (or the A key) will move the character left. The UP arrow (or the W key) will allow our character to jump in the air. In our game window, we have included terrain in the form of multiple platforms. These platforms are made from multiple terrain blocks. Throughout the platforms, there are computer enemies that hurt our character on contact. These enemies move back and fourth across the platforms in an attempt to attack your character. However, by pressing SPACEBAR, the character can shoot out a small bolt of lightning to retaliate against the enemies. You can keep track of the character's health with through a health bar displayed in the window. When the character's health hits zero, the game is over. Our game currently consists of only a demo for the mechanics of the game. Later development will add more features and include a storyline.

2 Design

2.1 Classes

As of now, we currently have 2 different classes. The first one is the class for the player, the second is the class for the terrain. For now, all our enemies are set a terrain objects since this is still in development. We plan on later giving it its own class. All objects in the project are given a sprite, which is the graphic for the object. For our player, the sprite is our little spaceman. For the terrain objects, they are blocks used to build platforms. All these objects are also given a position to be displayed. The `main.cpp` file creates a window and the main loop calls on the display methods of all the objects which then displays them at their given positions in the window. You can see the methods and descriptions for the project in the table below.

Files	Methods	Description
`Player.h`, `Player.cpp`	`Player(...);` `void move(...);` `void drawPlayer(...);`	A player object. The constructor `Player(...)` creates a new player. `void move(...)` reads for any keyboard input and if they are found, perform the specified movement action for the player. `void drawPlayer(...)` draws the image of the player into the window.
`Terrain.h`, `Terrain.cpp`	`Terrain(...);` `sf::FloatRect getBoundBox(...);` `int getX(...);` `int getY(...);` `void drawTerrain(...);`	A class for a terrain object. There are two types of terrain: platform blocks, and enemies. The `Terrain(...)` constructor initializes a new terrain object. `sf::FloatRect getBoundBox(...)` returns the coordinate are occupied by the object as a `sf::FloatRect` object. `int getX(...)` and `int getY(...)` return the coordinate position of the object and `void drawTerrain(...)` draws the terrain object in the window.
`main.cpp`	`main`	Creates and initializes all the characters and makes the map using terrain object. Creates the window for the game to be displayed in. Contains main loop for executing character actions and drawing all the sprites.

2.2 Relations

The main.cpp class runs the program. In the main file, the main method creates a new Player object and creates many terrain block objects to make a platform. These are then displayed in the window. The different objects in the window interact with each other through collision handling. Meaning whenever the space occupied by two different objects touch, things happen. When the player touches a terrain block object, they can no longer move in the direction of the collision. This is what makes our objects seem solid. Through collision handling, our character can walk on top of platforms, run into blocks, and get attacked by enemies. You can see our class diagram in the figure below. Since we currently only have 2 different classes, we do not have much of a class hierarchy. In the future, we hope to create two main classes: Character and Terrain from which we create hierarchies with sub-classes like PlayerCharacter, ComputerCharacter, PlatformTerrain, InteractTerrain, and so on.

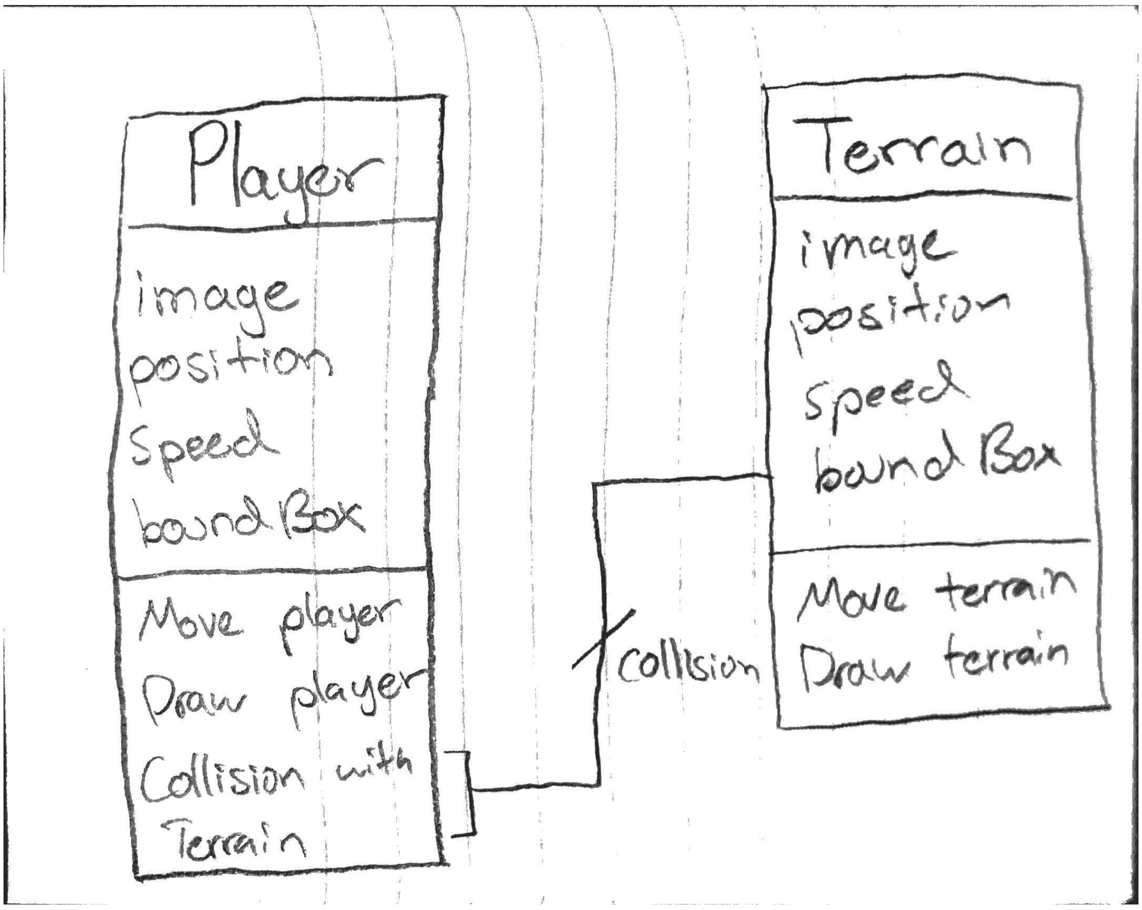

Figure 2.1: Our class diagram showing how the Player interacts with the Terrain

.

3 Final Thoughts

3.1 Implementation

To make this project, we developed with Microsoft Visual Studio Community IDE. To allow us to use graphics and allow user interface, we used the package SFML 2.4.1. SFML stands for Simple and Fast Multimedia Library. It is a cross-platform media package. Meaning it includes functions to enable us to create our game.

Our game is run through the main method in the `main.cpp` file. It starts off creating and initializing the objects we need for the game (the play, the platforms, and the enemy) and then enters the game loop. The game loop continually checks for user input and constantly refreshes the window, so all changes made will be displayed. Only user interface is run through the gaming loop, which allows us to implement all our methods in our object classes. All collision handling is done by the objects themselves. When the game loop detects user interaction, it passes the actions to the `Player` class to be handled; these actions include moving, attacking, and collision handling. The main method also keeps track of all the objects in the game. Lastly, it also keeps track of the Player's health, once the Player's health hits zero, a Game Over window pops up to show the end of the game.

3.2 Things Learned

When working on this project, we encountered many challenges and though those challenges, we have learned a lot of great information. To start off with was learning C++ in general as both of us only have a couple of months experience in programming with it. Through this project, we have encountered many bugs and new types of method implementations which have allowed us to overcome this challenge and grow in our understanding of C++. Another problem we had was learning how to use Graphics and User Interface since neither of us has ever done anything with graphics. By using online tutorials and small demos, we have learned how to use basic SFML graphics ad user interface packages to develop a good game. The other problem we encountered was optimizing our gaming mechanics to make our game run smoother. Through much trial and error we even managed to develop a decent gravity system to make the physics of our game seem more real. Overall though, the main thing we learned while working on this project was game development; the best thing we experienced was learning how to let a user manipulate and interact with our software.