Still Don't Know What to Call This Game

Submitted for CIS 4721 Senior Project

Vermont Technical College

Nicole Hurley
Seth Lunn

Revision History

Revision Number	Date	Comments
1.0	10/4/16	First Draft

Introduction

Roleplaying Coder (RPC) will be a roleplaying game designed to teach children at the middle school level programming concepts while simultaneously giving a storyline and unique and entertaining gaming experience.

Audience

Many children in grades five through eight are not exposed to computer science topics, despite the prevalence of computer science jobs in today's society. The popularity of gaming in all age groups is continuously rising. This makes using games to teach middle schoolers coding concepts a very good platform. There are already some smaller scale games available to teach different age groups, but there are none designed to be targeted to middle schoolers who enjoy playing role playing games and also want to learn about programming.

Goal

The goal of RPC is to introduce children of ages ten to fourteen to programming concepts. The game will focus on a male or female character (player's choice) who goes on a quest. During the combat stages and the gear creation phases of the game the player will have to use creative, logical, and problem solving skills in order to advance in the game.

This particular requirements document will be for the first combat stage that a player would experience in the game. The basis of this requirements document is for a demonstration of what the finished game would be. The game mechanics and combat phase are the basis of this demo. There are descriptions for various non-functional requirements. There is also a competitive analysis of similar products.

Key Concepts - Combat

The style of RPC's combat is turn-based. When a combat phase starts, the camera will shift and change perspectives. The player will always have the first turn. The player plans their turn out and then when they have finished planning the combat sequence begins for that turn. The enemy/ enemies will attack the player, if one side is still standing, the next round of attacks starts until one side loses.

- Around any enemies there will be a clearly marked combat zone that once entered the player will begin a combat phase in a turn-based style.
- Time will appear to be slowed or even halted when combat begins.
- There will be a panel that appears on the screen near the character.
- The panel will have 4 tabs each containing logic fragments.
- The logic fragments will all be put together inside a single space somewhere connected to the tabs. This space should be able to be seen in full whether the player is inside a tab or not. The windows should be separated somehow, but visibly related.
- There will be a tab for movements, gear, combat, and tactics.
    - The **Movement** tab will contain the logic that involves the characters movement (jump, dodge left, backflip, etc.).
    - The **Gear** tab will contain the armor/weapons that the character can switch to using based on what is appropriate for the current battle.
    - The **Combat** tab will contain logic that has to do with using their weapon or shield that they have equipped (slash, block, spin attack, etc.). Or any other special skills the player may have unlocked.
    - The **Tactics** tab will contain logic that is dependent on other factors, such as enemy behavior, health or stamina status etc. For example, the character could say something like "if health is below [Threshold]…. drink [amount] small health potion".

- Once the player has finished putting together their combat logic, there will be a button and/or hotkey to commence the fight. If the player loses the fight they can immediately decide to retry the fight or give up.

Diagrams

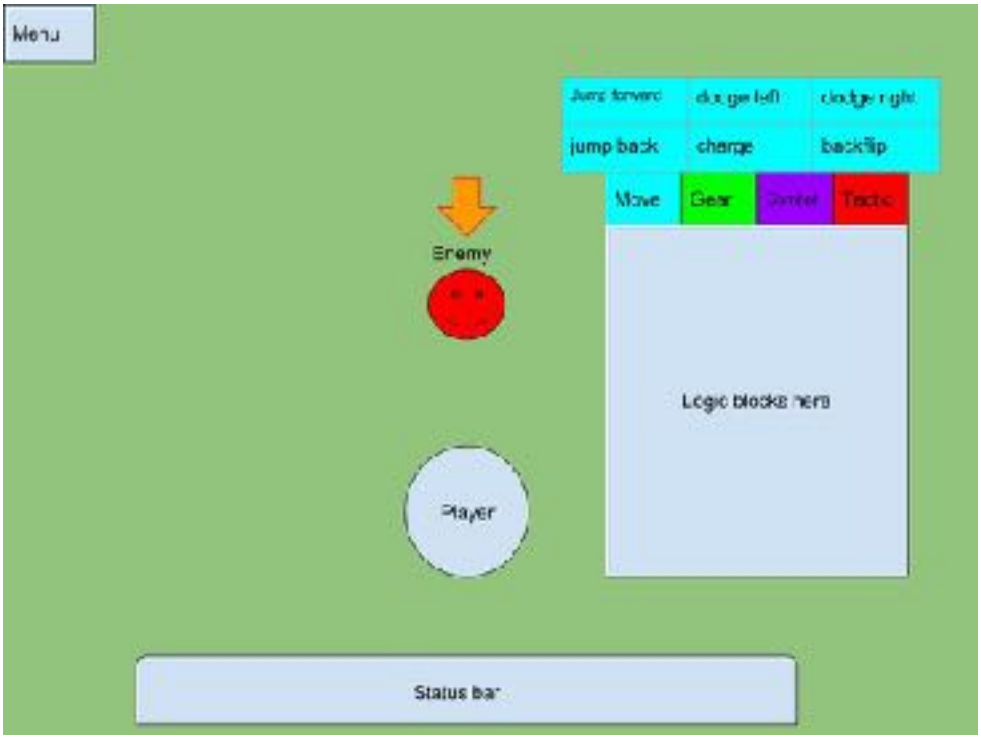

Figure 1: No logic blocks selected, the movement tab has been selected.

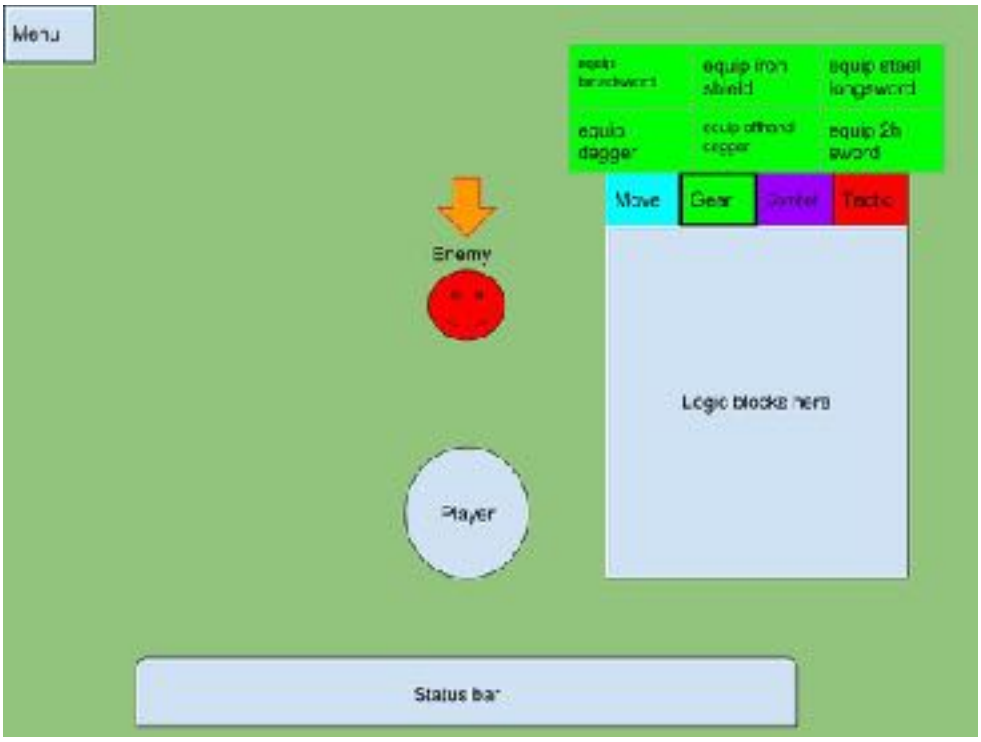

Figure 2: No logic blocks selected, the gear tab has been selected.

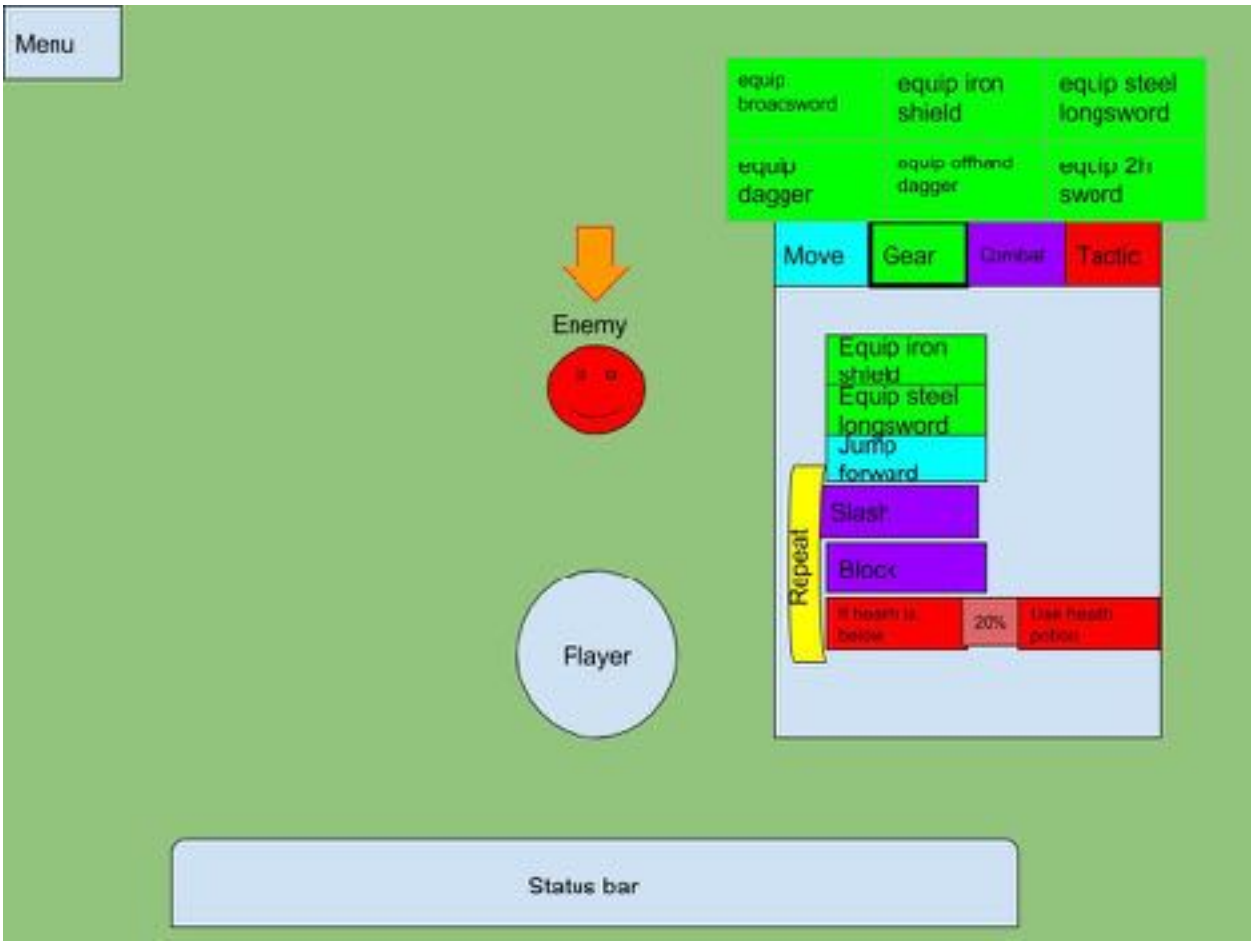

Figure 3: This shows what an example logic space would look like after the player chooses some of the blocks. It also shows how the repeat might look.

Other Features

Some of the details about combat are not clear in the key concepts or in the diagrams included.

Logic Fragments/Blocks

The various logic fragments will be in the shape of blocks that can connect together using a dragging motion with the mouse. Inside some of the blocks, if it makes sense to do so, there will be an amount that can be changed either via drop down table or inserting an integer.

Using the example above in the section about Tactics, the player would be able to choose

> "if health is below [20%] … drink [2] small health potions"

These conditions can also be based on enemy health.

"if [Selected enemy] health is [below] [10%] … do finishing move".

The blocks will also be color coded according to which tab they belong to.

Looping Behaviors

There should also be a way of looping certain behaviors. For example, if a player wants to dodge, attack, dodge, attack, they should not have to continuously place those blocks. Instead, there should be a way of having a repeat block that surrounds other blocks. This repeat feature would only be allowed to surround logic that does not include gear blocks.

The repeat block should have a way of using conditions to break out of the loop. For example, with an integer value. If the player does not indicate how many times something should be repeated, there should be an indicator to warn the player if some moves will never happen.

Multi-Combat

In multi-combat situations, the player will click to highlight a particular enemy. The highlighted enemy will then be the enemy that any attacks/blocks are directed at. If the player is out of range of a certain selected enemy, there would be an indicator before combat commences to warn the player that the selected logic will not work as they probably intend it to. This could be in the form of a color change or a dialogue that pops up.

Combat Area

If the player does anything that would mean that their character leaves the indicated combat area, the player will be asked if they would like to leave combat. If they click yes there will be no penalty. And they will be transported to just outside of the combat space. If they click no then they will remain at the edge of the combat area.

Non-Functional Requirements

Some of the elements of the game that relate to design such as music, art style, etc. will not be described in this document.

Perspective

The game will be in third person perspective, but during the combat phase the view will switch to a higher, almost bird's eye view in order to see entire combat field.

Platform

The game will be a desktop local game for Windows, Linux and Mac.

Saved Games

The player should be allowed to have no less than three different saved games within one installation of the game.

Stability

There should be an auto-save feature that is connected to a specific auto-save file. It should not overwrite the players current save game that they have loaded or any other saved game files. This should be triggered before every combat phase as well as major storyline plot points.

Speed

The initial load times and load times between location changes should not be more than 5 seconds (excluding hardware limitations).

Risk Factors

Given that the developers working on the project have never been involved with a project of this nature, there is a risk of running into problems regarding lack of experience and steep learning curve. There is also a chance of losing the focus of the scope. However, having a clear goal in mind from the beginning of the project should mitigate most of the said risks.

Competitive Analysis

There are several products available to teach children about programming concepts. There is also a product that does not teach programming concepts, but it is an RPG game that teaches children about American history.

Scratch

Scratch is a programming language made by MIT. It was a lot of the inspiration in how RPC's combat is to be structured. It uses the blocks that RPC will use as well as a similar structure to how the blocks are organized.

It provides children ages 8-16 with the ability to create animations, stories, small games, and share them with friends and people all around the world. Scratch is free and available in 40 different languages. Currently RPC will only support the English language. There is also an online community associated with Scratch.

While Scratch is a great tool to teach children logical thinking and the basics of programming, Scratch itself is not a game the same that RPC is. There is not one character that the player follows through a quest, and there is not a clear structure, it's very much an open sandbox.

Stencyl

Stencyl is a toolset to create games on a variety of different platforms including iOS, Windows, Android, and more. It also uses the block structure that Scratch and RPC uses. Stencyl has a free version and two paid options. The paid options offer the opportunity to make money off of the game that the user has made by publishing it on various platforms. Again, Stencyl is not a game itself, but merely a toolset to make games, which is very different than being immersed into a storyline.

Flight to Freedom

Flight to Freedom is an free RPG game that focuses on a history lesson. The history lesson is based around the underground railroad system, slavery conditions, and groups associated with anti-slavery movements. While there is no computer science related questions or programming in

anyway, it is similar to RPC in that it uses a roleplaying game to teach children a subject. They follow a character named Lucy through a storyline. The player is faced with many different options throughout the game that give them cause-effect experience. There is also the feature of different branches of gameplay, something RPC currently does not support.