Projectreport

Computergame

Johan Bjäreholt

Polhemskolan TE3E

2014-01-28

1 Abstract

Contents

2 Introduction

What does the design of a game look like abstractly in code and how does everything work together? What are some nice architectual code designs and how do i implement them properly? These are questions i think are very interesting and only have scraped the surface of before, and therefore have decided to learn how this works in depth with this project.

The aim of the making of this game is not to make a good looking and fun computer game, but to learn how to make a extensive and well made code base with a test game made on top of it. This is because i am not interested in learning the aesthetics of game making, but how to write the base code for a game. This means that i will avoid doing as much graphical and musicial work as possible and focus on the technology underneath this such as physics, world creation, character saving etc. This will make the project more scientific and more interesting to analyze around when the project is done.

A schedule has been made to endorse a stable development pace. It is also a good way to get a good overview on how close the project is to it's goal.

2.1 Recommended reading

There is terminology and a few concepts that will repeatedly be mentioned in this report. Most of the report will still be understood even if they are not experienced in programming and game making, but to fully understand some parts of the report (especially the analysis), you will have to understand the basics of how computers represents 2d graphics.

3 Theoretical background

When considering on how to actually make the game, i decided to use technologies i already knew how to use and was familiar with. This resulted in a combination of the programming language Python and the game library PyGame, which are are both designed very concise so i've been able to write more effective code on less amount of lines and therefore saving time which is crucial on a one man project.

The focus of this project is to learn how to create a advanced, flexible and stable game framework. The aim is not to add lots of game elements and make it fun, but is simply a proof of concept on how to make a good game engine. Consicely, it's about features and not content.

There are a few issues that are known that might be encountered. First off, writing a game is difficult and takes more time than people usually thinks. To explain everything the game does in code is alot of work, even though some of it might be very simple. Secondly, i chose a high level language which is efficient to write, but as all high level languages it is slow to execute.

3.1 Planning, Ambition and Time Estimation

By estimating how efficient my coding was the first weeks of my project, i could estimate and plan what i would be able to implement until the completion date. Since the focus was on the engine, there were a few specific things included in my engine that are usualy not present in simple games as this.

The first is my menu system. I want to be able to easily and efficiently create a menu, and make the code easy to understand to others if they would like to base their own game on my code.

The others are more game centric to make it feel premium. Physics are important to make the game dynamic, and some extra things that would improve but not as important are multiplayer, character customization and easy world creation.

These are my ambitions,

3.2 Software

3.2.1 Python

I chose the programming language *Python*[1] because of it's concise and therefore fast written code. It is not as fast in execution

[1]http://python.org/

as some other languages which are often used in game development, but since this is a one man project the effectiveness of the language is crucial. Since the game will not be too advanced and use to many calculations, Python should be able to run even on most low tier computers.

3.2.2 PyGame

The graphics and input library of choice has been *PyGame*[2]. This library is a python hook of the lower level SDL library, with simplified functionality but it is still enough for my use. This is used in the game for handling graphics, input and audio by communicating to the operating system and its drivers.

The graphics include tools such as creating a window, drawing and blitting the graphics to the window, simple geometric figures, transparency, sprites and image conversion. The input fetches keyboard and mouse input so events easily can be connected. Audio is sound file playback and conversion. I will not work much with sound though because the projects focus is technology and not content.

3.2.3 Open Source

All software used in this project is open source, because it gives flexibility and the ability to actually read the libraries code on parts where the documentation is not sufficient. By having this, compiling is straight forward and distribution of my program is legal even without having to get licenses for software. When proprietary libraries lack documentation you are left alone to fix it without any clues, while with open source you can simply read the library code if it is open source.

3.3 Source Code Management

The source code management system Git has been used to keep history and other stats on how the project grows over time. This has been nifty when having to rollback as well as keeping a stable branch of the code and just keep history of my changes. Every commit also has a short note on what the changes are, so it is easy to go back and see at which time and date what changes were made to the code.

[2]`http://pygame.org/`

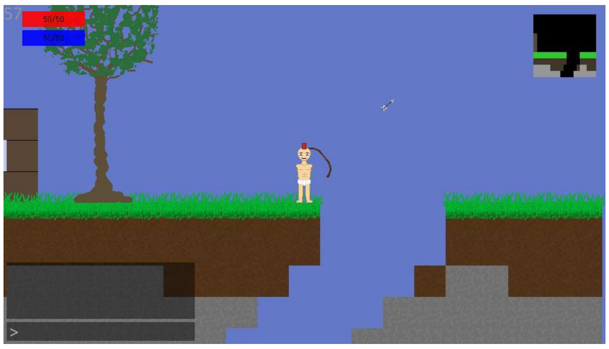
In addition to Git the website GitHub has been used to host the files, as well as giving others the opportunity to brows my code and try my game if they would like to. This website also gives a great overview of my project, as well as easier way to collaborate with others (which has not been done in this project though).

4 Method and Approach

Before beginning this project, i already know some very basic game design. I began the project with building a very simble code base with input support and a menu before actually making the game. It was a long time since i wrote my last game, so i wanted to start simple. After i had a solid input base and a nice custom made menu, i started working on the basics of the game.

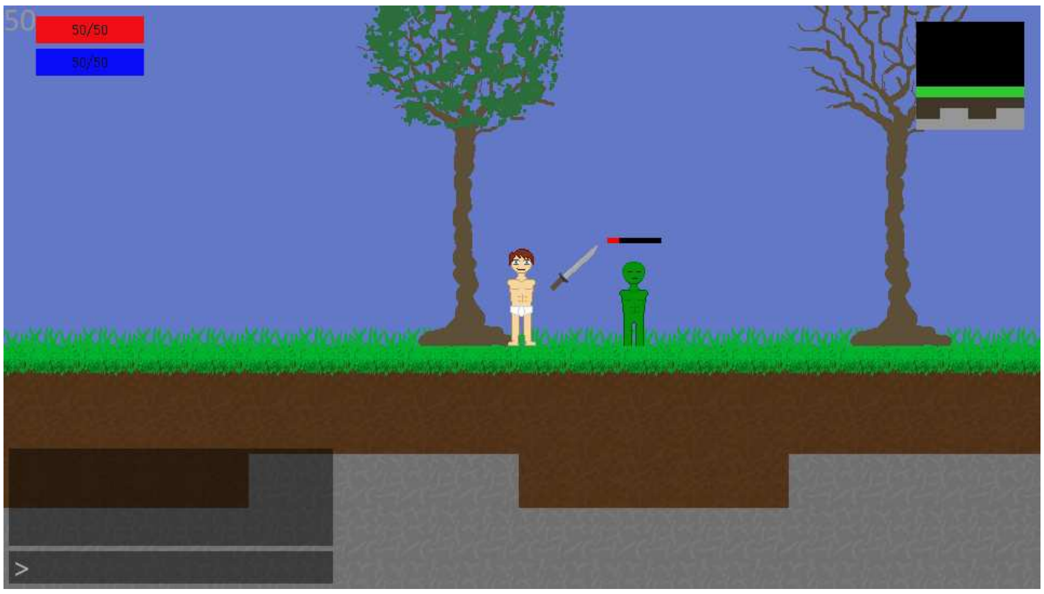
First off was the camera and world system was created. I decided to make the game up with worldblocks, because this was the easiest and best way if i wanted to make a world creator later on.

After this i started working on how to implement objects (both collidable and uncollidable) such as trees, stairs, bouncy balls, signs etc. The difference between a worldblock and a object is that worldblocks always are 50x50 pixels in size and cannot be moved in game. Objects are dynamic and can be of different sizes and can be either collidable or uncollidable. I then added so collidable items could push eachother if they were both movable, which was a real challenge. After 4 rewrites i finally got it working somewhat, it was still buggy and could crash the program when using bouncing objects but i decided that it took to much time so i decided to continue with the rest of the code.



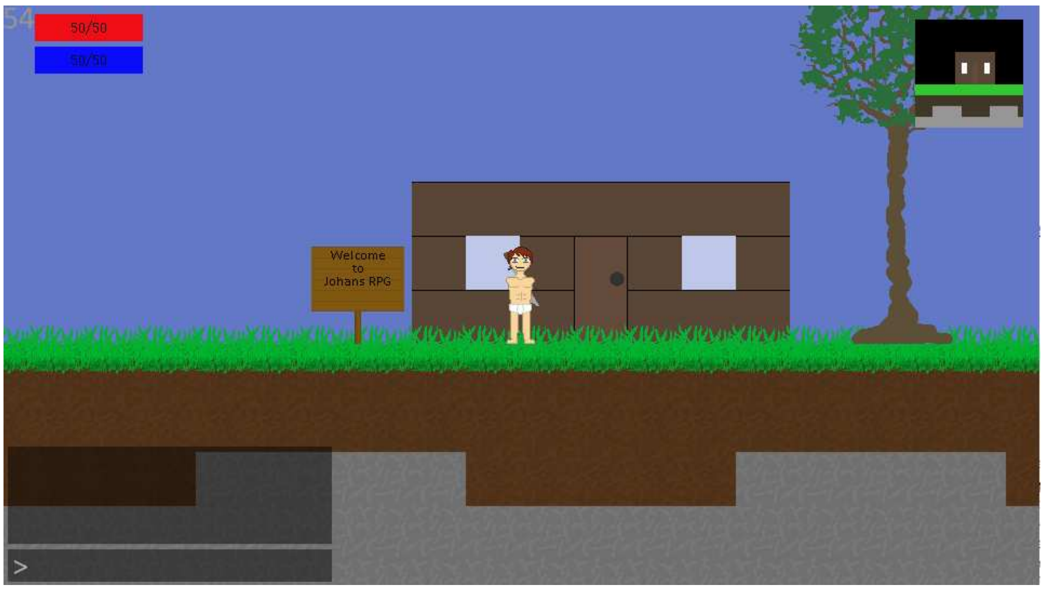
Then i started working on the character, characters appearance and weapons. I made two types of weapons, bows and swords. On both of these i wanted a animation that the sword got swung and that the bow was aiming where the mouse pointed, but since i am not good at animation i instead only rotated the images which made it alot easier. I also made an arrow object which was tricky to make, because it was supposed to rotate according to how fast was shot and how fast it was falling. To do this i had to do use cotangens on the x and y speeds to find the angle it was supposed to be displayed.

By now the world started to feel more like a game, and i started working on monsters and user interface. The only monster implemented is a zombie which runs towards a player if one is within a specific radius from it. Every monster has a healthbar above it and can be killed, although it cannot kill you! The user interface is simple and consists of only a chatbox, minimap and health- and manabar.



5 Result

During the development of this project, i have tried to always be ahead of my schedule in case a week with multiple exams would turn up. I was in pace until the last 3 weeks before the report deadline where alot of homeworked suddenly turned up. I also thought that the deadline was in April when it actually was in February, but luckily i had planned only to add extra features such as multiplayer the last week and not work on the core of the game.



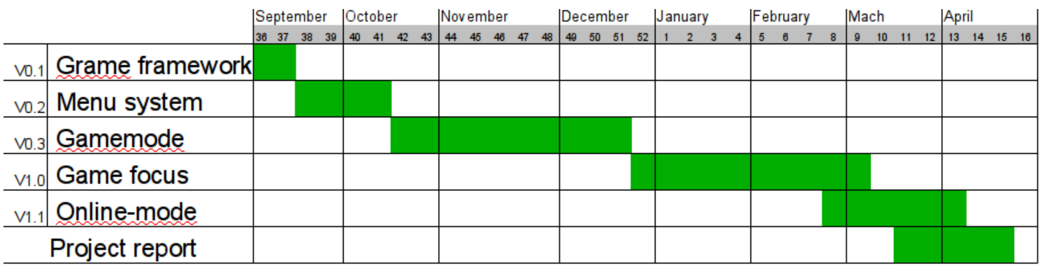
6 Analysis and Discussion

6.1 Time planning

Even though i have tried to always keep up with my schedule, there have still been a couple of occasions where i have not been able to implement things by the time planned.

Both were on the end of the project, and one was because of bad planning. I did not know when the project report was supposed to be turned in and i thought i was pretty sure it was sometime in april, but i was wrong. The whole project was supposed to be done by April(not only the report), and the project report was supposed to be turned in before the end of Febuary. I realised this in the beginning of February, but was not able to keep up because of lots of homework.

As a result, the project is still complete but i was not able to accomplish as much as i wanted to in terms of features. During the development period i also implemented features that i had not planned from the beginning such as character customization and text input, but now i would rather had focused on the game than on its interface and customization.



This was the original time plan, which was followed except for the last three months

6.2 No hardware acceleration

When i started writing my game i knew that PyGame had support for hardware acceleration. Hardware acceleration means that the program can be accelerated by using the computers graphics card instead of solely on the processor. What i didn't know was that PyGames hardware acceleration is limited to only Windows and even then it is not fully hardware accelerated.

The biggest performance hog in my game is the drawing of every frame to the window. Since most computers will not support the hardware acceleration though, the game will run easily on almost any computer on lower resolutions, and will run on the monitors native resolution on most desktops.

This is pretty bad performance since the graphics are not advanced at all and not many sprites are in the game.

6.3 Physics and collision

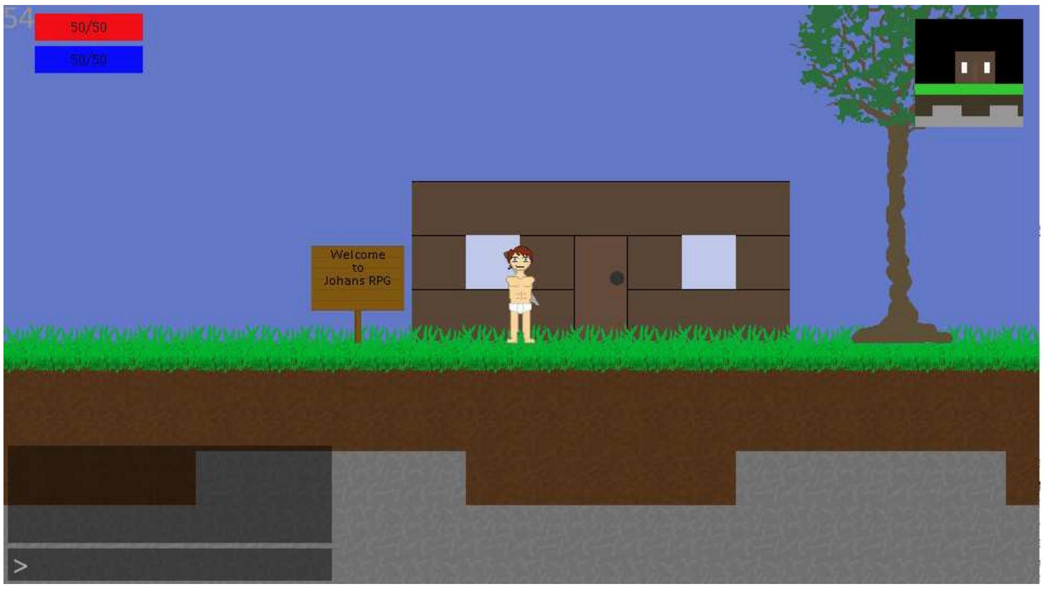
I have previously worked with collision detection and i thought that it was easy to implement. What i wanted in this game was actual physics, so i simply tried to trabslate the physics i have learnt in school into programming. The idea did seemed very easy and implementing the formulas was easy, although there was one thing i was not aware of. In reality everything moves with a constant speed, but a computer works step by step and if objects do not move with constant speed at the same it becomes somewhat hard. The way to make this as realistic as possible it to have very small iterations of movement on each item, but that would be alot of work on the processor so that was not how i implemented it.

After re-writing my code over 4 times, i finally came up with a acceptable result. I first moved each object in each direction, and if the speed was faster this frame than it's height or width, it should extra check so it did not jump through an object. This worked well if one of the objects are static, but if either both objects were moving (and especially if they had the bouncy property) it was not as straight forward.

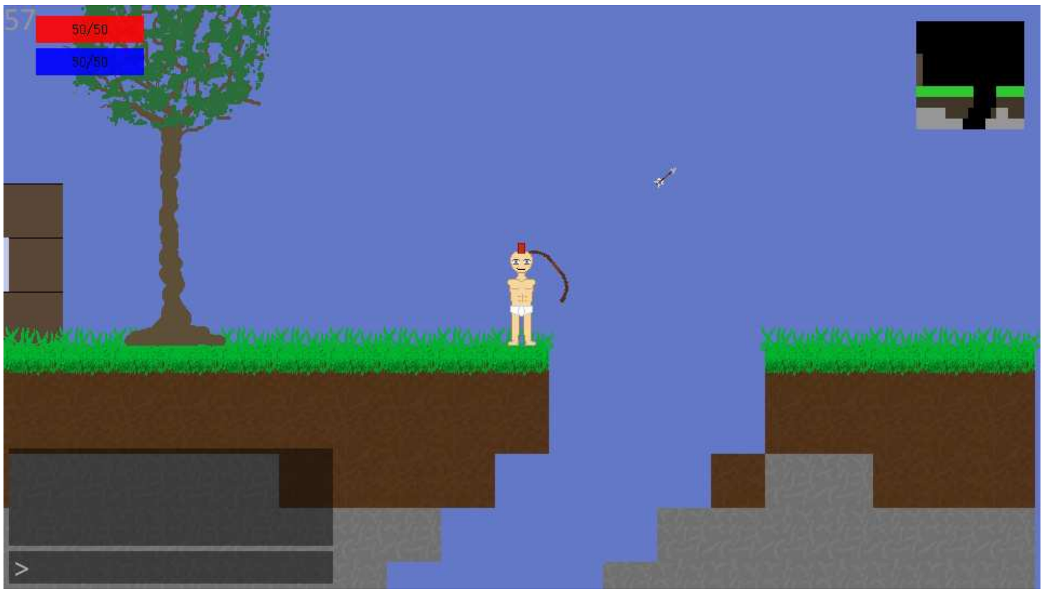
Moving objects can collide with each other pretty well, but i have not been able to make the bounce work as intended. A object with the bounce property can bounce onto static objects and can be pushed as long as it is not pushed into a wall (If that is the case, the game will crash).

7 Screenshots

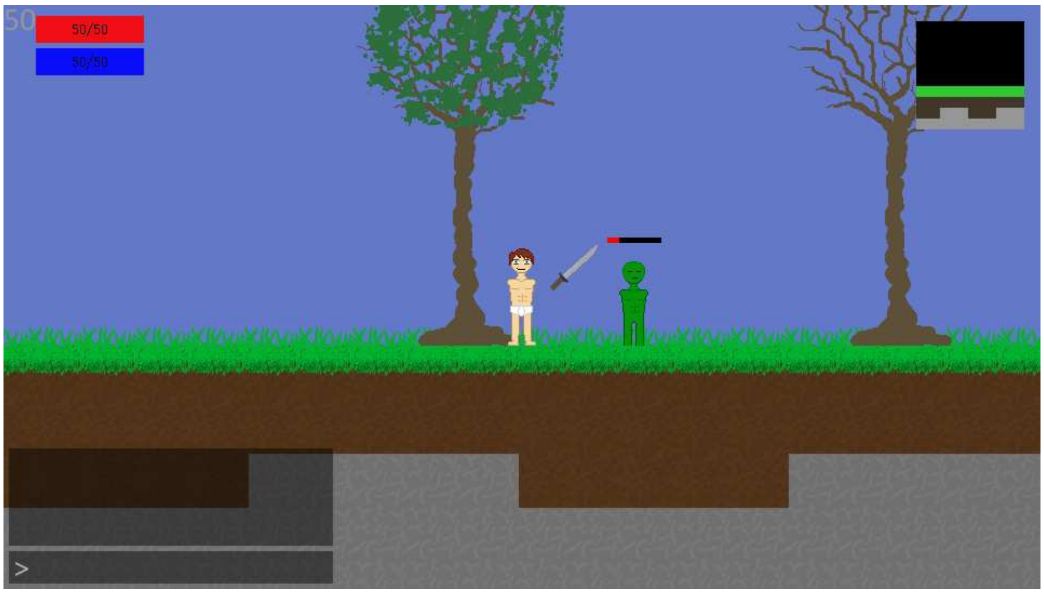
7.1 In-game



Player standing at spawn with his sword unwielded

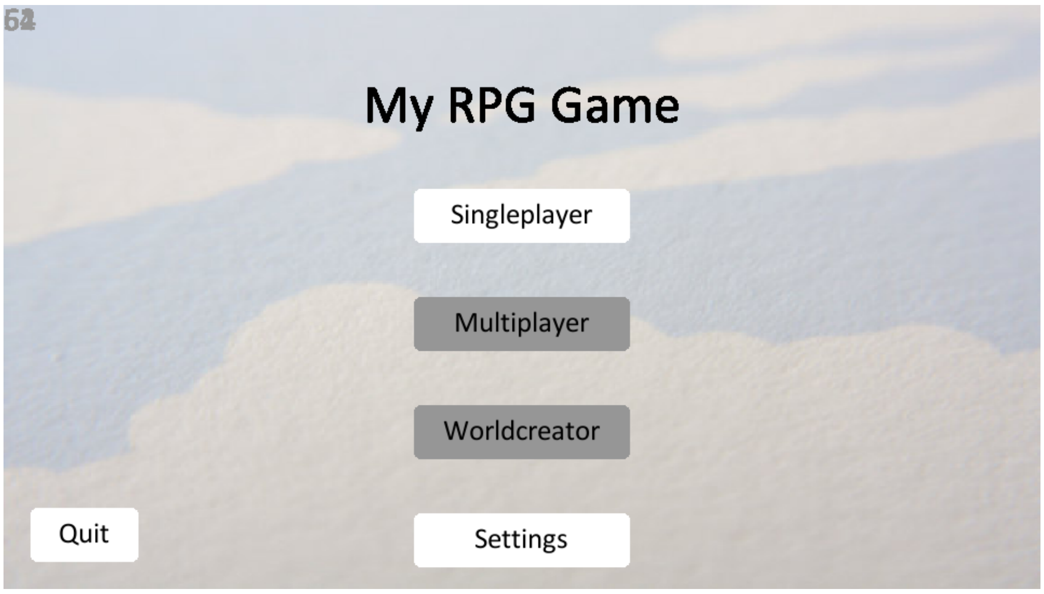


Player shooting an arrow with his bow

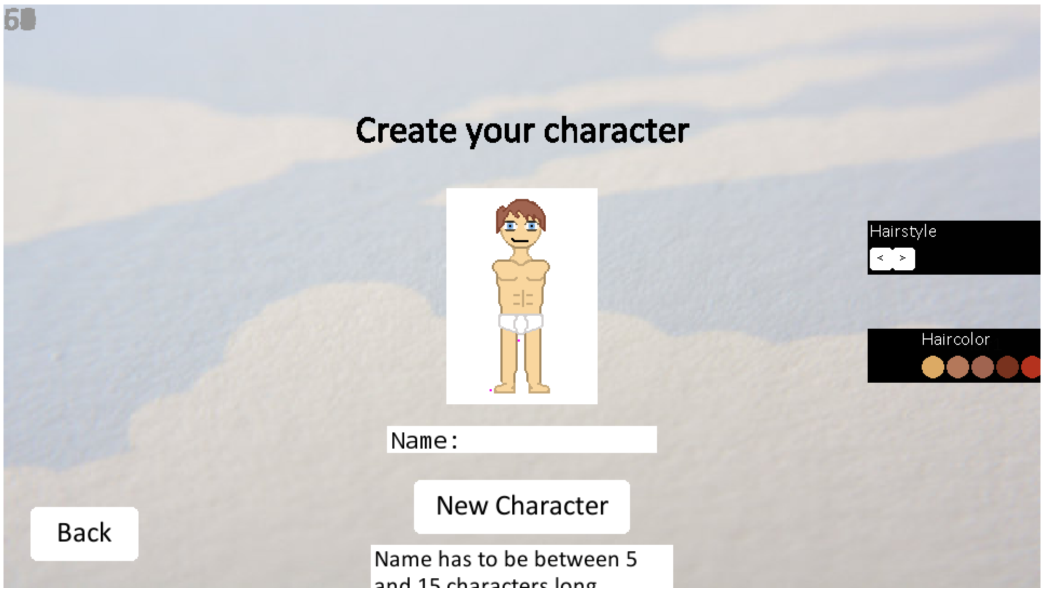


Player hunting a zombie with his sword

7.2 Menu

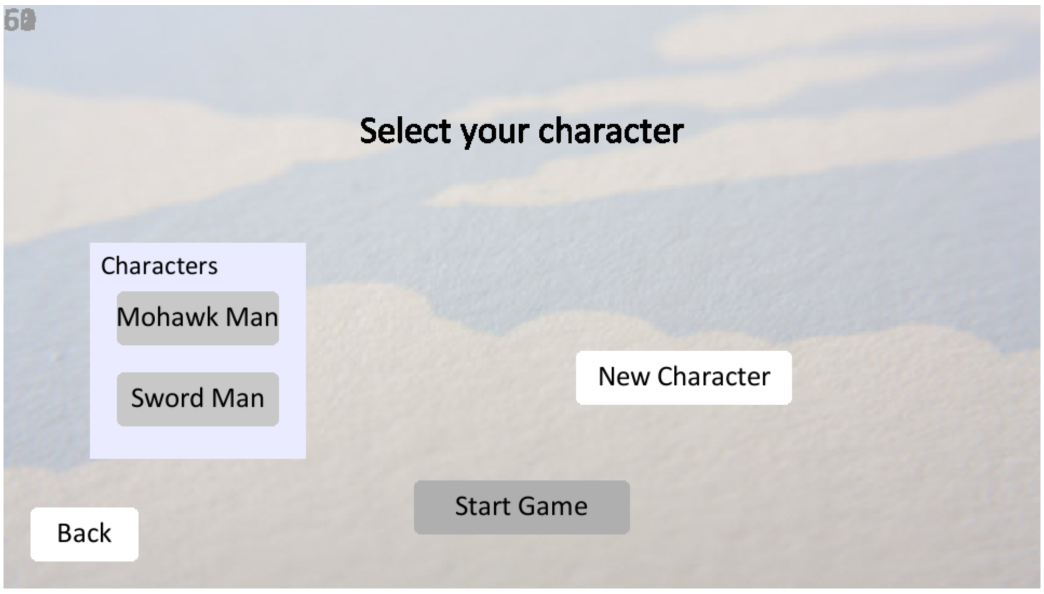


Main menu

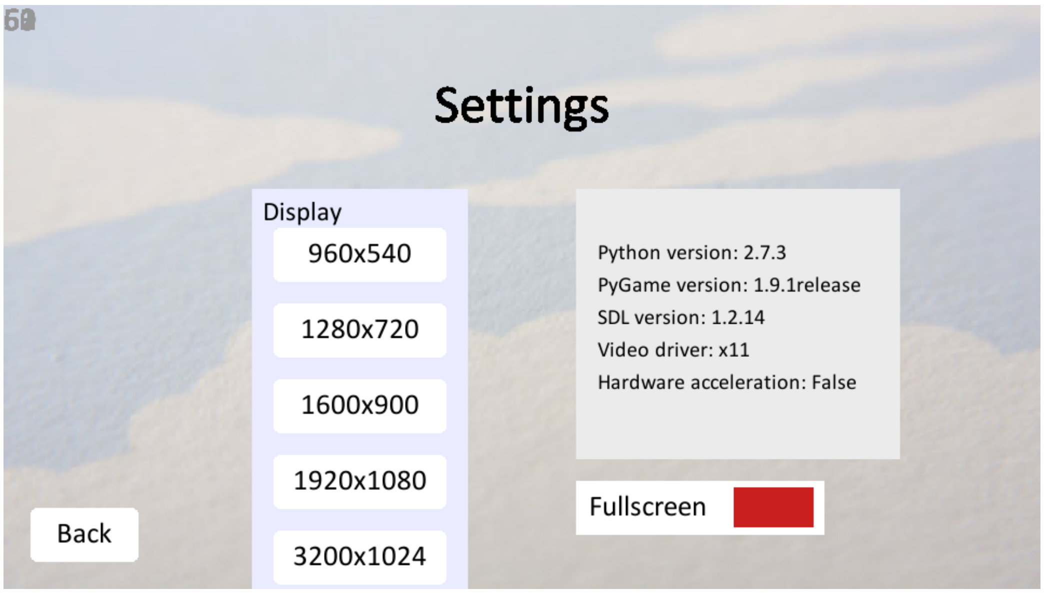


Name has to be between 5
and 15 characters long

Character creation menu



Character selection menu



Python version: 2.7.3
PyGame version: 1.9.1release
SDL version: 1.2.14
Video driver: x11
Hardware acceleration: False

Settings menu