The Rogue Knight

The Rogue Knight

This game is a side scrolling shooter where a Knight must clear the monsters from the Springstar Fields. The character must kill all the enemies using his melee and long range weapons, and can use the surrounding platforms to avoid enemy attacks and get closer to the skybound monsters. I was planning on having a Boss Level and 2 overall levels however I did not have time to implement these, however the overall mechanics of the game can be played with in this level.

Genre

Original Code vs My Game

Side Scroller:

My game is a side-scroller, meaning that as the player moves, the world is moving around them. This changes the genre from top-down space shooter to a 2-D interactive world and allows more elements of skill to be used (i.e. platforming, jumping, etc.) than the original game. Players are encouraged to utilise the game's environments to kill the enemies. In the code, this is achieved by defining a Camera class which works based off the concept of "the world doesn't move around the player, the player moves around the world and the camera follows". This means that as the player's X-coordiate moves, we are keeping them in the centre of our screen and moving each of the sprites and backdrops a certain offset each frame.

Platformer:

The original game supported a top-down space shooter where the players was free to move up,down, left or right using WASD. In this game, the player must utilise more than moving left and right to play as some obstacles block a player from just walking forward. The player must jump, walk, and interact with the 2D space, which adds more moving elements to the game and further changes the game from its original genre of a space-shooter where the controls interact completely differently with the world.

Shooter:

Rather than turning the game into a pure platformer, I kept some of the elements from the space shooter (i.e. the shooting) in order to give the game more excitement. By doing this, it is not making the game any less of a platformer, but instead adds more possible enemies such as flying enemies that must be defeated from a higher area only accessed through a series of platform traversals.

Theme

All the artwork obtained in this game is referenced in the code. When developing this game, I wanted to use music and consistent art in order to maintain the illusion that this game was taking place in it's own world. I wanted the backdrop and characters to all resemble the world a magical knight may travel in, with an enchanted forest, mythical enemies and a mysterious player character sprite to achieve this aesthetic.

Features

Original vs My Game

Player Status – Health and Magick

The original game did not have any way of denoting a players status, so I added in player health and magick. The player can lose health by being attacked by the monsters in the game as they try to clear the level, and for every magical bullet fired the player depletes their magic store. The magic replenishes over time, but if used continuously it will not recharge faster than the player uses it. The player cannot replenish their health however, as I wanted the effects of being attacked to be riskier and long-lasting, adding an element of tension and importance to the moves and tactics made. In the code, the player magick recharges by incrementing the players "magic level" every time our CurrentAnimationTime MOD 10 is equal to 0. This means we are never recharging it faster than the player can use it.

Sprite Animations

Each character, monster and projectile has it's own set of animations which are all triggered by individual events. When our character uses magic or swings their sword, we see them open their cloak or swing the blade. If left idle, the player changes to a standing animation and when running or jumping, they have different animations as well. The Mushroom and Flying Eye enemies all have their own animations too, for flying above the ground or biting the player if they get too close. Sprite animations are loaded into a HashMap<String, AnimationObject> in our Model where the String is the key for finding a specific animation (i.e., running, jumping, hit, etc.) and the AnimationObject is a class which holds the number of frames, y1 and y2 coordinates of the specific frames, and other details necessary for loading the animation.

Bullets and Projectiles

There are two kinds of projectiles in the game – the player's magic and the Flying Eye's bullets which rain from above while the player deals with the ground mobs. The player shoots towards wherever the mouse points, allowing them to focus simultaneously on platforming with WASD and keeping off enemies with their magical projectiles. The projectiles had animations as well for when they hit an enemy, however the frames weren't loading up quick enough, so they are not seen in-game unfortunately. In the code, the bullets are calculated by a Mathematical formula referenced in the code, and then in order to translate the correct dimensions I needed to transform the coordinates with the Camera class position every frame.

Platforms and Collision Detection

Since this game's genre is a platformer and side-scroller, it needed to be able to detect platforms and walls to move around the world correctly. I created a Collision Manager class that contained all the Platforms as java.awt.Rectangle objects. This has a built in method for checking intersections with Rectangles, and I used this to detect and react to collisions between the player and it's environment by checking the player against all the platforms every frame.

Jumping and World Physics

By implementing a gravity check, the player is always being effected by gravity. In order to get the player's jumping mechanics correct for this platformer, I needed to make the gravity a constant value, and the players jumping power was stronger than gravity (i.e. the jump speed is larger than gravities pull down).

Smarter Enemy Logic

Enemies will spawn randomly on the map and have random speeds as well, so the player must adapt to this during gameplay to avoid getting caught in a corner. This also keeps the overall game more exciting as you have to perfect the use of the bullets in order to not waste any.

Music and Sound Effects

This game has a background music and multiple sound effects in order to make the game more immersive. The player has sounds for being hit, jumping, hitting an enemy, and upon the completion of the level.

Functional Game

The game has a start menu to begin the game. My intention was to use the beginning level as a tutorial level with dialogue boxes to instruct the player on how to play, however due to time constraints I was unable to do so. However if you look in the "res" file, you will see the tutorial dialogue I was intending on implementing. To finish the game, the player must score a minimum of 200 points before dying, where every successful hit on an enemy counts as a point, and sword hits count as 5.

Multiple Controllers

Uses both keyboard and mouse. The left mouse button shoots the bullets for the player and the right button swings the players sword.