COMP 2673-1
Dr. Jeffrey Edgington
Final Project Final Milestone

Medieval Arrow

Team Name: MLG Coders

Team Members: Tresdon Jones, Jason Komoda, Chris Tracy

Overview

Our project is a dungeon-crawling type video game. There are multiple rooms for the player to explore that contain skeletons for the player to fight. The theme of the game is a medieval dungeon. The problem that our team tried to address is to make an enjoyable game using the programming techniques we have learned thus far, and learning more techniques along the way that will prove to be assets for this project and the rest of our computer science/programming careers. Making video games is an extremely good way to learn more about computer programming, especially in the areas of object-oriented programming (perhaps the most obvious place to apply OOP), GUI/graphics programming, and keyboard/mouse interaction with Java graphics frameworks/libraries (JApplet, JFrame, Slick2d, lwjgl etc). We think that we made a decently fun game and we're proud to have worked on it. For our project we used Slick2d which is a wrapper for lwjgl and helped speed up the development process as well as help with smooth rendering. Using slick exposed us to many tools and the idea that it's not always a great idea to reinvent the wheel.

Development and Division of Labor

Our team often worked together kind of coding together while we were hanging out and so the division of labor is kind of fuzzy but it will be outlined as accurately as possible, but note that these aren't completely accurate.

Chris - Did pretty much everything related to the art in the game. Designed the dungeon, created the sprites using a generator, cut them up into rows to use for animations.

Jason - Created the Projectile class. worked on the player class. Did some bug fixing in the main class, and balanced the game to be playable.

Tresdon - Worked on the main class/ main GameState. Worked on the Camera class that follows the player around while they move. Set up the animations for the player and enemy.

We found that we worked as a team very well. Working in a team presents challenges because of the fact that there needs to be so much collaboration. Making sure everything is the right size so that things look professional was something that we needed to take into account, as well as everybody's level of skill: what they could make independently, what they needed help with, etc. It was a very valuable skill to practice.

Classes

public class Player {

This is the class that represents the player. The player will be the only one in the world since there is never another player. The player is a very involved class having a health a way to change the sprites to make sense for the game state and a way to move around the map in a player defined way. The player is possibly the most complex class of the program since we wanted it to be natural, considering it's the thing that players interact with the most.

}

public class Archer extends Player{

The Archer is a child class of the Player class. The only thing that is done in the Archer class is set sprites and animations that the class will use. This is the only playable role implemented in the game so far. Similar classes like a Mage would do very similar things.

}

public class Enemy {

The enemy class is a class that represents all things that can harm our player. This includes the creatures, the bosses(not implemented), and anything of that type.

The enemy has a certain amount of health, a certain amount of damage, and a way to move around the map. The enemies currently move randomly and shoot randomly, there is no AI implemented for them, but that's something we would have liked to accomplish. This class handles all of the logic for anything that an enemy can do.

}

public class Skeleton extends Enemy{

The Skeleton is a child class of the Enemy class. The only thing that is done in the Skeleton class is set sprites and animations that the class will use. This is the only enemy implemented in the game so far.

}

public class Projectile{

This class represents all of the projectiles that are used in the game. These projectiles are implemented only in the form of arrows currently but the logic is available to change the sprite to be whatever is desired. The Projectile has collision detection methods and it gets created at a point and then just travels in a direction for a specified duration.

}

public class GameState extends BasicGameState{

BasicGameStates are used by our program to be able to move from state to state (or that was the goal). We wanted there to be a main menu, a pause menu, and things like that, and GameStates would help to do that, we just didn't get around to implementing them, so this is our main GameState and it's the only one currently. It handles the drawing in this state and the logic for all of the entities that exist in the state.

}

public class DungeonCrawlingGame extends StateBasedGame{

This is the main launcher provided by Slick2d that draws a window and initializes the game. In it an AppGameContainer is specified, so anything that we want to run can be that class and it becomes runnable. This class is very short as it only really initializes things, I believe there were probably some things that could have been utilized in it that were not, and so that would be interesting to look into

}

Closing Thoughts

Overall we were quite proud of the project that we made. We thought it presented an applicable, interesting challenge to use a new library, but we're glad we did because of what it facilitated us making. Things that we really would have liked to add would be an AI for the enemies, since they always shot randomly it was kind of too easy of a game (except on occasion where the enemies would fire tons of arrows really quickly.) We

also would have liked to add more classes/roles. We would have liked to include NPCs to make the game more story based and immersive. Pretty much all of the things that are standard in modern video games of the genre we would have liked to add, but unfortunately time didn't really permit that. It's reassuring though because it definitely feels like we're capable of implementing those features if we choose to, both with the way the codebase is set up and our skill level. This project was very useful for gaining experience, learning tons of new things, it was fun to work on, and we're proud of what we made so it was a pretty good way to end our first year of computer science education.