OvE (Orcs vs. Elves) - 2D Strategy Game

Slightly Less Nice Green

Mark Dockendorf, Isaac Thomas, Luci Tran, Thomas Truong, & Ryan Vo

Sponsor: Dr. Ian Parberry & Curtis Chambers

Project Overview

The idea of the game would revolve around a 2D turned based system that allows users to use strategy to progress. The progression of the game revolves around multiple characters that the user is able to control. The result of this project will be a game that users can interact with. The game would contain an in depth weapons system that alters a user's choice and effectiveness in the game. The decisions chosen by the user will be saved and used to determine the results of combat in the game.

The motivation for this project is to create a free to play game that is similar to strategy games available people have to pay for. At its core, the game is inspired by the Fire Emblem series but will be a minimized version that can be done within the time constraints given for this project. All group members have a strong interest in creating this turn based game in addition to having prior knowledge on how the game should be played.

Features

F1) Character sprites
F2) Weapon/armor system
F3) Turn based combat system
F4) Character levels
F5) Class progression
F6) Weapon skill levels
F7) Main menu
F8) Currency
F9) Character animation transitions
F10) Two dimensional top-down grid based map
F11) Keyboard or mouse input
F12) Character selection

Technologies

We are using Unreal Engine (v. 4.24.2) alongside Visual Studio (2017) to create our game. The game will be coded in C++. Assets will be created by ourselves (sprites) and royalty free sources (background, etc.).

The system will need a keyboard, mouse, and monitor. Additionally, the computer in use will need to run on Windows and Linux operating systems.

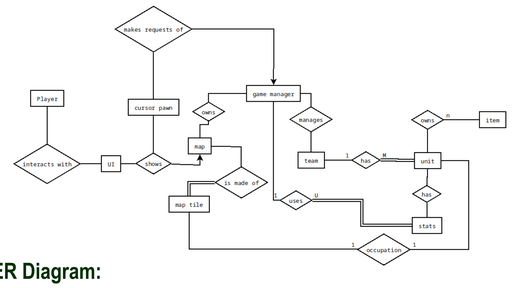
Design

The main gameplay feature of our project is turn-based combat between player controlled characters and artificial intelligence (AI) enemies. The starting point for this feature is designing how all future calculations will be based on. The unit class will be utilized by both player characters and AI enemies so that we can have reusability of all numerical calculations.

The combat system revolves around character stats and the weapon each character has equipped. Before entering combat with an enemy, the game will calculate the damage, hit rate, and other bonus effects that will be displayed to the user. If the user decides to initiate combat, then the final calculations will be decided. This combat design allows the user to plan out attacks on the enemy before initializing attacks. Additionally, the weapon and magic triangles will add variety in the different types of ways the user can play out each level.

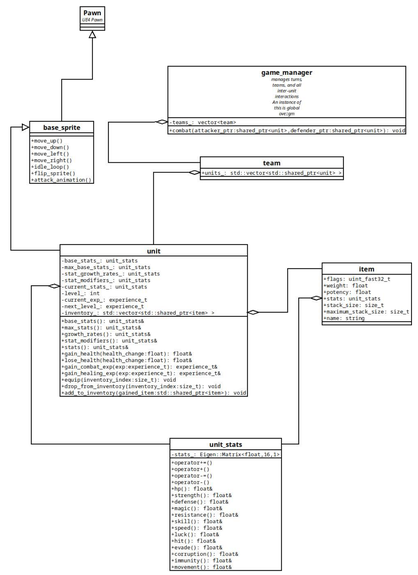
As characters move to perform their actions, their art needs to correlate to the situation. The sprite state diagram is a high-level overview of sprite transitions based on an action. Its purpose is to guide the creation of a character's art as well as listing the conditional trigger to switch one sprite for another. While it is possible to only create sprites as needed, all player characters and AI enemies share the same type and number of sprites (i.e. moving in a direction, attacking, or idling). The reason for this is that during gameplay, any action a player character could do, an AI enemy needs to be able to do on its own.

2D shots of all characters.



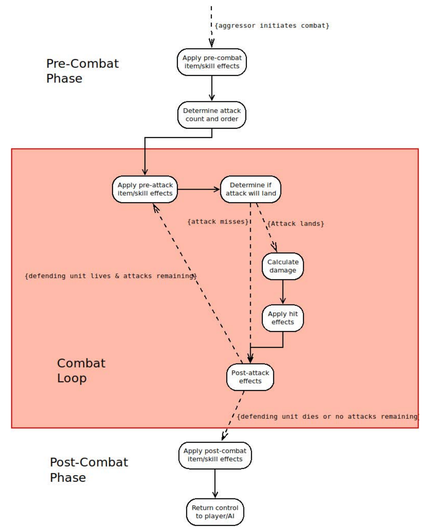
ER Diagram:

Shows the relationships of entity sets stored in a database and how they interact with each other.



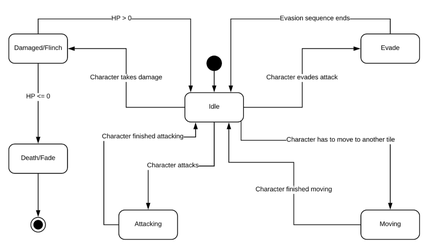
Class Diagram:

Designed in a way that modulates the characters created. Our method for inserting the sprites into the base sprite class ensures that we will not need to create multiple characters with redundant data types. Each character inheriting the base sprite allows ease of development and lower coupling since the sprite data types do not need to interact with the base stats data type.



Combat Sequence Diagram:

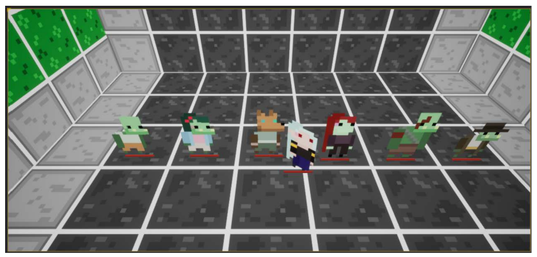
Shows the steps it takes to go through combat with a character an the results that should occur when an action is taken.



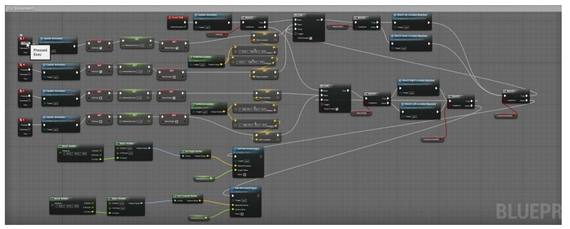
Sprite State Diagram:

Shows the different states of a sprite and what triggers the sprite to advance to its next animation.

Screenshots



Screenshot of the characters on the playing map.



Unreal blueprint interface for movement.

Testing

The Unreal test environment is called Environment Query System. This system tests actor and pawn locations as well as the artificial intelligence system in Unreal.

For testing, we mainly focused on Unit Testing. Movement control for the player was tested by utilizing the Unreal Engine 4's "Play" tool. Specific keyboard inputs were used to check if character movement corresponded in the correct manner.