Corey Amoruso
Ryan Bergquist
Zacharias Shufflebarger
CS 447
November 10, 2014

I. Game Overview

Gameplay:

The game will be an isometric RPG style game with co-op multiplayer not unlike the Diablo series. The player will control a single character who needs to navigate through a series of dungeons in order to progress through the game. The dungeons will consist of open areas filled with obstacles and enemies that will try and prevent the player from reaching the next area. Each level will be made up of several rooms containing a variety of different enemy types. Like any RPG the player will gain experience by battling enemies, this experience can be used to upgrade the players abilities. These abilities include character strength, stamina, and defense. Defeated enemies will also have a chance to drop items for the players disposal such as health potions. In co-op mode two players will be allowed to play through the game clearing dungeons together. To make the game more challenging with more than one player enemies will have increased health and defense. Here is a gameplay video of Diablo 2 for reference: https://www.youtube.com/watch?v=-EhReGR6n6k.

UI:

The User will control their character with the keyboard as well as the mouse. The keyboard will be used to direct character movement using the W, S, A, and D keys. The mouse position will orient the character so that it faces the mouse pointer. Clicking the left or right mouse buttons will make the character attack in the direction of the mouse. Clicking the left mouse button on items will also let the User collect items into their inventory. There will be obstacles and objects that will cause collisions with actors, keeping them from moving into a tile. Actors will also collide with each other, but Items will have no collision detection. From the inventory menu, the User may select items to use, equip, or drop. This will be done using the mouse.

Actors, Items, Obstacles, and a portion of the map will all be visible entities to the User. The screen will show a certain amount of the map around the character. As the character moves around, the screen will ‘follow’ the character’s movements, scrolling the map around as the character explores it. The screen represents a field of view around the character, so anything within that field of view is visible to the User, however there will be things outside of this field of view, or off screen, that exist.

The goal of the User is to not die. Actors will attempt to attack and overwhelm the User as they navigate the map and come near to enemies. The User will find items or gain levels to increase their attributes and be able to handle stronger enemies. There will be at least a few levels so there will be some progression.

This style of game is fun because it calls on the human instinct to survive. The real-time element makes it fast paced and keeps the User on their toes. A strong level of difficulty makes games like this entertaining because the User feels pressure to do well, but has limited resources like time and health points. Some mechanics can help alleviate some of the difficulty, such as items that regenerate health points, or other items that make the character more formidable. This game will attempt to achieve these same game goals; it will be fast paced with intuitive controls, there will be a challenging number of challenging enemies, and there will be items to collect to help the character get farther into the map and levels.

II. Development Strategy

Starting Point:

We will start with Zach's code (from his first game) for representing the map in the game process. We might also use his code for map generation although this has not been decided on for sure yet. The overall structure of the game will also be based on Zach's previous work but will be expanded to include the new features.

There will have to be some changes to the base code, but those will be small. The first change will be to calculate in the isometric projection from world coordinates to screen coordinates and back during the render phase. A second major change will be to remove the turn-based logic and make the game real-time. With this second change will also come some changes to Actor movement as well as pathfinding and execution. For the most part, these changes should be relatively simple to make, but are each very important.

Sticking Points:

The hardest part of this project will be the networking as the game state may be large depending on how many players are playing and the number of enemies/objects that are presently active in the game. This problem is easiest to remedy by planning ahead and organizing our data and classes in such a way that it is easy to serialize and transmit.

Another challenge we may face is making the game feel like a true RPG complete with character classes and skill trees. This problem arises from the time constraint we are faced with and will be handled by keeping the amount of skills and upgrades to a minimum.

Milestones:

1.
    1.1. Map data structure completed. (Zacharias)
    1.2. Isometric graphics rendering. (Zacharias)

1.3. Simple player movement (walking/running). (Zacharias)
1.4. Game States completed (start, running, game over, paused). (Ryan)

2. (Alpha Release)
   2.1. Enemies and objects spawning. (Ryan)
   2.2. Player offense and defense moves rendering. (Ryan)
   2.3. Multiple rooms rendering. (Zacharias)
   2.4. Basic Collision detection. (Zacharias)
   2.5. Mini Map completed. (Ryan)
   2.6. Player skill tree implemented. (Corey)
   2.7. Player upgrade screen complete. (Corey)

3.
   3.1. Networking completed. (Corey)
   3.2. Complete level rendering. (Zacharias)
   3.3. Add more enemy types (Ryan)
   3.4. Add more player skills. (Corey)
   3.5. Add more objects and player upgrades.(Corey)

III. Technical Showpiece

Our Technical Showpiece is going to be networked multi-player co-op. This will not only add complexity to the gameplay and programming of the game, but will also force us to spend more time designing the structure of the game in order to efficiently serialize the game data for server to client transfer. Since we are going to be using a Client-Server architecture for our game the server will need to receive messages from each client updating the corresponding player's position, status. and game state (in the event one player pauses). In turn the server needs to update each client by serializing the current game state and sending it to each client.

This will most likely be accomplished by creating a serializable object that holds the need to know information for the clients such as other player positions/actions, enemy positions, object positions, and game state. This object will then be sent through a stream from the server to each client. We are planning on using the Apache Commons Library for socket creation and for opening the stream between client and server.

IV. High Bar

1. Refined visuals: Make the HUD look nicer or add more animations.

2. Level progression: Add more levels with increasing difficulty, and increasingly interesting items.
3. Increased Player complexity: Variable damage, critical chance, chance to hit/dodge, or more skills.
4. Class system for Player control: More interesting classes or deeper developed classes.
5. Abilities and Spells for more complex game-play: Special effects or attacks with unique animations.
6. Item diversity: Items that take advantage of other High Bar implementations. A wider variety of items with a wider variety of effects.

V. Low Bar Checklist

1. Isometric projection: Render the cartesian world coordinates to the screen represented as an isometric game. (Alpha release requirement)
2. Level Map: At least one map rendered to screen with the Player able to move around on it. (Alpha release requirement)
3. Mini map: A small, zoomed out overview of the map showing what has been explored. (Alpha release requirement)
4. Basic Actor/Player RPG attributes: Health points, attack damage, armor rating, all used to handle attacks between the Actors and the Player.
5. Objects or terrain: Collidable objects to hinder free movement around the map.
6. Basic items: Items used to increase the Player's health, attack, or armor. Consumable items to heal the Player or bestow some other benefit.
7. Simple HUD: Show the User how their character is doing.
8. Actor pathfinding: Actor AI will chase the Player when it gets near enough.
9. Screen scrolling: The screen will only show a portion of the world and will follow the Player as it moves around the map.
10. Networked multiplayer: Allow multiple Users to play the game together.