Diablo 1 (1996) – Retro game

Kevin Absen – Kernmodule Game Development jaar 2

Introduction:
Diablo is an action role-playing hack and slash video game. The player moves and interacts with the environment primarily by way of a mouse. Other actions, such as casting a spell, are performed in response to keyboard inputs. The player can acquire items, learn spells, defeat enemies, and interact with non-player characters (NPC)s throughout the game.

Twist:
While Diablo is originally played in isometric-perspective, my game will be top-down perspective with rotatable horizontal camera controls (third person). I also want to redesign the art style to 3D low poly style which makes it easier for me to control the environment (such as roofs disappearing when entering a building). The game will have a linear story but with many "benefits" that give you rewards (such as gold or items). There will only be one playable class for the time being which is a combination of Barbarian and Wizard, but with a focus on sword combat. I will scrap a lot of stats that Diablo 1 has to make it a little easier for me. The following is a list of the available stats in the game:

- Strength: Increases damage done per hit.
- Defense: Decreases damage taken by enemies.
- Vitality: Increases maximum health points.
- Stamina: Increases maximum mana points.

The skills/abilities will also vary a lot from Diablo 1. Below is a list of the attainable abilities:

- Primary: Slash (deal 100% damage, single target).
- Secondary: Fire Strike (deal 150% damage, multi target).
- Defensive: Shield Bubble (decreases damage taken by 50%, duration 5 seconds).
- Power: Sprint (increases walkspeed by 50%).
- Summoning: Minions (summons 2 clones of yourself that each deal 50% damage, duration 10 seconds).
- Ultimate: Immortality (increases health by 500%, damage by 500%, duration 10 seconds).

Plan of action:
For this project I will use Unity Engine 2018 because I feel that this engine is most accessible and reliable for designing this type of game, considering it has friendly support for user interfaces (combat and inventory UI), three dimensional controls, storing values (items, stats, prefabs) and Artificial Intelligence (navigation).

Design patterns:
Code-based I will make use of C# (Unity supported), with emphasis on a few design patterns. The first one is a *singleton* for my game manager which will take care of storing items and stats for the game. Second are *object pools* and *states* for spawning enemies and destroyable objects like crates and bone piles. These items will appear and disappear in the game continuously, so to increase performance I want to use object pools.

My priority plan is to first create the navigation so that the player can move around by clicking the mouse on the map. I will use the NavMeshAgent for this because this is by far the easiest method of detecting colliders that the player cannot collide with such as trees and buildings. Secondly I want to create the inventory system that will be one of the main mechanics of the game. This will be used to store weapons, armor, stats and other (unwearable) items. I will mimic the original Diablo interface for this, but with a different art style (vector art) that I will design in Illustrator. Once this is finished and working, I want to focus on the combat system and AI. The player has to be able to cast attacks that require mana and have a cooldown when casted. The player also has health points that need to be decreased when hit by an enemy. All these elements will be implemented into the user interface.

Of course the player also needs to be able to attack enemies, and enemies need to be able to attack the player. For this I will create advanced AI, which will also make use of the NavMeshAgent just like the player. When a player kills an enemy, the enemy has to drop items on the floor that can be picked up and/or stored in your inventory (%chance). Same applies for destroying objects.

My third priority is to create a shop system where the player can sell and buy items (weapon, armor, potions, etc…). This will require NPC interactions with perhaps a simple dialogue system.

Last priority is finetuning and creating extra features like graphics integration (main menu, animations, better models, map layout).