Project WindHaven

Technical Design Document

2D Top-Down Action Roguelike Dungeon Crawler

Aaron Gilbert – 18015521

Jordie Muljana – 17993858

Sam Ward – 17004205

Yuan Hao Li – 17982806

Spring 2020

Table of Contents

Bibliography

Nintendo. (1986) *The Legend of Zelda* [NES].

Overview

Game Concept

Windhaven is a single player top down dungeon crawler, where the player's goal is to escape with treasure, utilising the environment around them to defeat enemies.

Platform

The game will be developed for Windows PC

Development Environment

Game will be developed in c++ using the SDL Library

Third Party Tools

FMOD for audio integration
Sfxr for audio creation
Box2d for physics engine

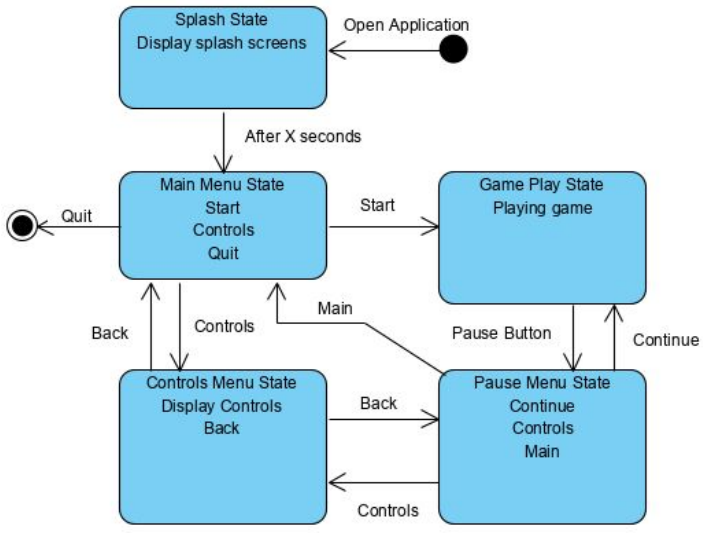
Software

Visual Studio Enterprise 2017
SVN

Game Play

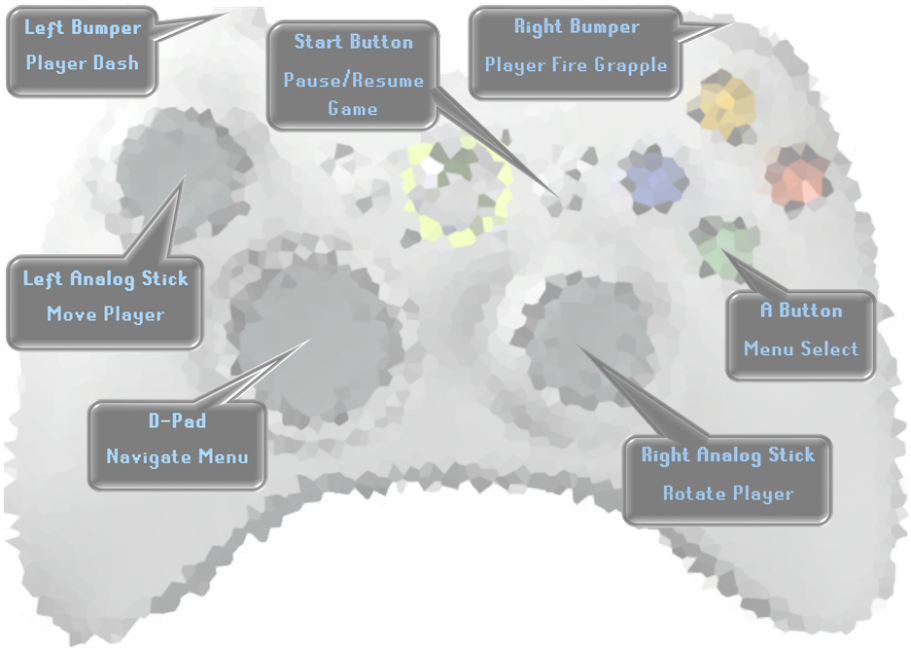
Game Flow

The game has two primary states: a menu state and a playing state. In a menu state, players can navigate through options in a menu using keyboard arrow keys or the directional pad on a controller. The core gameplay is contained in the playing state. The State system follows a hierarchy with menu state consisting of three possible sub-states, Main, Pause, and Controls.



State Machine Diagram for game flow

Game Controls



Core Mechanics

Physics

Game Physics consists of basic vectors for movement and direction. Box2d will be utilized for collision detection as well as contributing to the physics environment with entities being dynamic bodies.

Player Movement

Player movement is achieved through the use of both analog joysticks. Left analog stick moves the player in all directions, and the right analog stick controls player rotation, with the player facing the stick direction.
Smooth rotation of the player is achieved by using the arctan function on the analog stick values.

```
float angle = atan2((float)m_angleY, (float)m_angleX);
angle = RadiansToDegrees(angle);
//change angle from -180 - 180, to 0 - 360
angle = fmod(angle + 360, 360);
SetTargetAngle(angle);
```

Algorithm using current angle, target angle, and rotation speed in order to give the rotation a more realistic feel, rather than snapping to joystick position.

```
if (m_angle > m_targetAngle) {
    if (m_angle - m_targetAngle > 180)
        m_angle += m_rotationSpeed * deltatime; //goright
    else
        m_angle -= m_rotationSpeed * deltatime; //goleft
}
else if (m_angle < m_targetAngle) {
    if (m_targetAngle - m_angle > 180)
        m_angle -= m_rotationSpeed * deltatime; //goleft
    else
        m_angle += m_rotationSpeed * deltatime; //goright
}
//stop player from spinning continuously
if (m_angle > 360)
    m_angle = 0;
if (m_angle < 0)
    m_angle = 360;
```

Grappling Hook

The grapple is used to move either the player or objects around the map. Certain entities will be grappleable and depending on their density will react in accordance with the box2d physics environment. Grapple is fired in the direction of the right analog stick.

Player Dash

Player dash will be a short increase in movement speed in the direction of the right analog stick. During this dash other input is ignored.

Obtaining Treasure

Treasure chests are objects that appear inside Rooms, with each Room having only one Treasure each. Upon stepping onto a tile with a treasure chest, if the player has more than one Key, a Key is automatically consumed and the Treasure chest is opened, granting the player the amount of gold stored inside.

Enemy Behaviour

Artificial Intelligence

Enemy is one of the main obstacles to let the player win this game , with each room there are multiple enemies. The enemy itself would walk toward the player around the room and when the player is in its attack range then the enemy would get into attack mode , damaging the player.

Victory Conditions

The game is won when the player reaches the end of the level. A score is awarded based on the amount of treasure remaining and the time. The game is lost if the play loses all of their treasure. After winning or losing, a message is displayed on the screen and after a short time the game returns to the main menu.

Level Design

Map Generation

Each room will be comprised of a 12 x 16 array, with each element of the array being a 50 x 50 pixel level tile. The top two and bottom two rows are primarily empty, or consisting of ‘hole’ tiles, with gaps in the middle for bridges. This is also the same for the two columns on each side.
The tile array will be initialised by file presets, if provided, otherwise they will randomly generate, with a low rate for holes and a very high rate for floor tiles.
Enemy and object spawning will also use the file presets if provided.

Graphics and Animation

Texture loading and Sprite creation using SDL.
Game consists of animated and non-animated sprites as well as alpha blending.
Game is played in a fixed window.

Sprites

Sprites sheets are to be in png format.

Audio

Audio will be implemented using FMOD and created using sfxr in .wav format.

Sign-Off

Aaron Gilbert	
Sam Ward	
Jordie Muljana	
Yuan Hao Li