**Going in Blind**

Game Design Document

**Goal**

Demonstrate the concepts and tools learned over the course of the Spring 2022 semester in Advanced Game Development.

**Overview**

Produce a game that appears to use a 3D space using 2D visuals. For this project, it is recommended that we use Godot as our 3D engine, but we are not limited to just using it. The purpose of this project is to showcase what we have learned over the semester and to allow us to express some creativity.

**Game Description**

Going in Blind is a pretty standard platformer with the added twist that the player cannot see the environment normally. In order to get a grasp of their surroundings, the player must throw some object to creating sound that will illuminate the surrounding area. The players own footsteps will also cause sound waves allowing them to see the ground beneath them.

The game is to be in a first-person POV and will use mouse and keyboard controls as well as using a gamepad as an alternative control scheme. The player will encounter enemies in the form of a ghost that they will be able to see without needing to use their sonar abilities. The ghosts will pursue the player but can be defeated by using some sort of item. To win, the player must reach the exit and the play loses if they get caught by the ghosts too many times.

**Game Assets**

Player: Standard looking human. Simple and 3D. Walking and throwing animations needed.

Thrown Object (rock?): Some sort of rock-like object that the player can find and throw. Should be illuminated so it can be located after thrown.

Thrown Object (weapon): A different type of object that players can find that allow them to damage the ghost enemies. Should be distinct from the other Thrown Object.

Ghost: Shaped like the Pac-Man ghosts but in a 3D space. A floating animation when idle and one when moving.

Environment: Maze like space with moving platforms that the player must navigate. All black to give the illusion of not being able to see.

Audio: Footsteps and some rough objects hitting surfaces will be needed. The ghosts will also have some spooky sounds they will make.

UV maps: Ghosts should have a simple UV map that gives some character to them (maybe cute or spooky). Throwable objects will have some textures to them.

**UI**

The game’s UI must be simple and not cluttered so that the player can “see” as much of the environment as possible. The player’s health, in the form of 3 hearts, will be shown along with any throwable objects the player has in their inventory and whichever one is currently equipped. A crosshair of some kind might be a good option so that players do not get too confused on their orientation in the environment.

**Game Object Mechanics**

Player Mechanics:

The player is controlled using WASD to move, Spacebar to jump and the mouse to look around and left click to throw and probably some button ‘E’ to interact with objects. On a gamepad the left joystick will control movement, some face-button designated to jump and another face-button to interact with objects and a last face-button to throw. Only one player object is created and will stay persistent.

Thrown Objects Mechanics:

The thrown objects will have a consistent velocity for the sake of not making the system too complex. The objects adhere to some sort of gravity. The throwable objects will have collision boxes that, when they come into contact with the environment, will release a wave of sound that allows the player to perceive the world. The thrown objects will not have any complex collision boxes and simple spheres will work.

Ghost Mechanics:

Ghosts will have various walking patterns as well as a chasing pattern. Ghosts will be sprinkled around the world to act as obstacles for the player to either defeat or outrun. The ghosts will have two collision boxes, a standard collision box and a smaller hurtbox. The former is used when detecting if they have been hit with an object and the latter is used when coming into contact with the player. This is to make it feel a little fairer and give the player more chance to escape their pursuer.

**States and Transitions**

Launching the game will start the gameplay state (no need for a fancy title screen just yet). Player instructions will be displayed on walls/signs/various objects to assist the player. Winning leads to a win state where an overlay is placed that says “you won” but could be changed to lead into the next level. When taking too much damage the game will go into a lose state and ask the player if they want to restart or quit. Restart will return the player back to the start of the level while quitting will close the game.

**Milestone Plan**

Basic models and environment – Simple, less-detailed, player and enemy models. Simple level design for testing will be included in this step.

Player motion – Get a working motion scheme working as well as being able to interact with throwable objects (not yet throwable).

Working enemies – Enemies with some walking patterns and collision detection allowing them to interact with the player model. Implementing taking damage will take place here but not losing the game.

Throwing objects – Add physics to throwable objects and a throw action for the player model. These objects should be able to interact with the environment as well as the enemies. If an object hits an enemy, the enemy is deleted.

Increase level complexity – Design some interesting and fun obstacles for players to run around in. This could include moving platforms, small maze, lock-and-door puzzles, etc.

Implement blind mode – Take all preexisting models and adjust them for “blind mode”. Ghosts will be white and glowing. Throwable objects will be outlined and have a faint glow. The level will now be entirely black and only viewable when sound is created.

State management – Create states for starting the game, winning, and losing the game.