Testing Grounds

A Game by Braeden Kurfman, Chris Gnat, and Joseph Wenger

Software Requirements

&

Game Development Document

Table of Contents

Introduction

Testing Grounds, our project game, is going to be a 3D platformer using the Unity Engine. The code that will be added to add game elements will be tied into Unity via Visual Studio Code. The Game itself will consist of a player in first person, a finish line, obstacles, Items and a monster chasing you to the exit. The player will use the mouse to move where the player looks, and the keyboard for movement. Left click will be how the player uses the grappling hook. Some of the obstacles will be disappearing platforms, moving platforms, a gravity warper, and jumps that require a grappling hook, one of the items the player has in the game. There are also mario-style items the player can collect that will increase movement speed, jump height, and more . The monster is going to look like a giant octopus that will chase the player through the level and give the game a sense of purpose and tension.

Outside of gameplay, the music and sound will be made entirely electronically and will feature digital synthesizers and drum machines. This will be our first time working with a DAW and composing a soundtrack. For the sounds, it will sound as digital as possible while still resembling the real sounds. Some of the soundtrack will be in 8-bit style as well. For the art style, it will be our first time using Blender and will feature lower poly models with simple textures. Some of the objects that will be used for platforms will be made from Unity's basic object system that allows for shaping of objects. This would also include any sculpting done to the ground, walls, and ceiling to make the surroundings feel a bit more immersive.

Some changes that have been made: it originally didn't have a monster. It was originally just a timer to try and get to the end as fast as possible, but we thought a monster could be more interesting and we can still keep the timer to keep the game competitive. We didn't originally think about having a grappling hook in the game, but our philosophy on game design is "If it has a grappling hook, it is immediately a better game". And the Gravity changer went from a small idea to a big selling point of our game. We have decided the game will be rated E

for Everyone according to ESRB’s rating system since there is very minimal “violence” similar to most plaformers where the most violent event is falling or getting a failure screen from the enemy.

Some of our inspirations include: Super Mario 64 (Nintendo,1996), Mirror's Edge Catalyst (EA, 2016), and the Portal series (Steam, 2007 & 2011).

Requirement Specifications

User Requirements

Movement - player will be able to strafe left, and right as well as move forward and backward. Player will have a base state (walking) and a sprint state (running). The player will be able to jump in both base and sprint state.

Interactions - Player will be able to interact with a grappling hook, Doors, and ‘modifier’ objects.

Game Design - a First Person Perspective will be used for the player. The environment will be designed with 3D platformer principles and the artstyle will be based on ‘portal’ with simple and blocky objects and a void to encourage a virtual world feel (similar to BOIII and its speedrun mode).

UI Design - Player will be able to access a ‘pause’ menu at any time. Player will be able to see a current run timer and will be able to view a leaderboard as well as a death counter.

Clear Objective - The player’s goal should be to finish the level.

Enemy - The enemy will chase the player, if the enemy ‘catches’ the player, they will restart from the beginning

Sound - Music and sounds will play throughout the game. Sound effects will play after certain actions occur, while different music will play depending on where the player is currently located.

Tutorial - A tutorial will help the player learn the controls and the goal of the game.

Good Performance - The game should be able to run smoothly on most hardware and keep up with the player’s actions.

Offline Playability - The game should be playable without the need to connect to the internet.

System Requirements

Movement

Function: allow the player to move via various methods

Description: movement methods are 4-way strafe, sprint, and jump

Inputs: WASD, Spacebar, and Mouse

Source: player peripherals = Mouse and Keyboard

Outputs: movement to the player

Destination: Player's screen

Precondition: Player is in a instance where inputs from the keyboard and mouse are allowed for movement.

Postcondition: Player movement occurs.

Interactions

Function: manipulate player behaviors

Description: mario style items will modify player behavior (speed, jump height, etc) and Grappling hook which acts as a pendulum for the player to swing

Inputs: mario style items are triggered on contact - Grappling hook triggered by left click

Source: Left click for grappling hook and since the items are on contact the input would be keyboard and mouse

Outputs: modified player behavior for mario style items and a rope starting from the model to the the contact point

Destination: Player's screen

Precondition: Player interacts with an object.

Postcondition: The object's effect occurs to the player.

UI

Function: See stats related to the player's activity/ pause the game.

Description: Players will have a counter for the amount of times they have lost to the monster and/ or have fallen off the level on screen. Players will also have a timer shown on the screen for the current level. The player will also be able to pause the game at any time during gameplay allowing for access to leaving the game or continuing if desired.

Inputs: When a player starts a level, these HUD (heads up display) elements (the death counter and timer) will appear on the screen and start at 0. No player input is necessary other than starting the level. For the pause button, they will press the ESC key on their keyboard and use the mouse to interact with the buttons that appear.

Destination: Player's screen

Precondition: Player initiates the game start process.

Postcondition: Player receives information about their stats and the information appears correctly.

Clear Objective

Function: Ends the level for the player

Description: The finish line for the level will be clearly indicated by a large beacon of light that the player should stand in to complete the level.

Inputs: Player's current position. Finish line position.

Origin: Scripting, tracking player movement inside the scripts to indicate true/ false if the player is inside the finish line.

Outputs: Level complete announcement. Time spent on level and deaths indicator.

Destination: Player's screen/ HUD

Enemy

Function: Causes a player death, moves in the same direction as the player.

Description: If the squid monster catches the player, the player will be reset to the beginning of the level and the death counter will increase by one.

Inputs: The squid is an entity object. The player starting the level spawns the squid, squid begins moving towards the player based on the movement script.

Outputs: Enemy creates sound effects when it gets closer.

Destination: Player's screen

Precondition: Player must be in a level unpaused.

Postcondition: Enemy is able to move towards the player and "kill" the player's character.

Side effects: The monster may be too difficult or easy to evade for some players depending on skill level.

Sound

Function: play sound effects and music

Description: Players will hear unique sound effects when doing an action, such as jumping, using an item, finishing a level, etc. Music will also play in the game and on the title screen.

Inputs: Starting the game, starting a level, player actions

Outputs: Sounds going into the user's audio system (headset, speakers)

Precondition: Event triggers sound effect/ music

Postcondition: Sound is played without error

Side effects: The player's audio might not be enabled or too quiet to hear, or it may be too loud and hurt their hearing.

Tutorial

Function/ Description: Introduce the player to controls, objectives, what HUD icons mean, basic instructions.

Inputs: Players move forward into the level, tutorial comes from preplaced signs.

Outputs: Visual display of what the controls are.

Destination: On signposts within the levels.

Precondition: Player starts the first level and moves forward in it.

Postcondition: The player completes the first level and fully understands the controls.

Side effects: The player may forget or skip one of the tutorials so they are confused as to which button to press later on for an action.

Good Performance

Function: Allows the player to make quick decisions and rely on their inputs to make the mistakes of the player not come down to poor framerate/ freezes. This will allow for a better player experience. Button clicks should be registered within 200 milliseconds.

Inputs: Player inputs from mouse and keyboard as well as in game actions and events. Unity Engine and code.

Outputs: Frame rate, load times, potential crashes, Unity's game testing log

Destination: Player's screen

Side effects: Removing something that causes frame rate issues could cause the game to be less complete. Certain hardware could respond in a different way to specific events. Some systems may receive higher performance than others on the same level of performance otherwise.

Offline Playability

Function/ Description: Player will be able to access the game while not connected to the internet while maintaining all aspects of the game.

Inputs: player inputs from keyboard and mouse and game stored on hard drive/ SSD.

Outputs: Game is rendered

Destination: Player's screen

Precondition: Player has the game installed to their hard drive, SSD, or has it available to download from an external drive.

Postcondition: Player is able to load the game on their hardware as they would normally.

References

Original Plan Document:
https://docs.google.com/document/d/19YjF4SExoG6Ingm8yobcJ4VjKjX309huD7KZppPWFuw/edit?usp=sharing

Google Document Presentation:
https://docs.google.com/presentation/d/1oomw2bpqdmUzMdilltA9UJvldlfh2JLyIRE0ESgJLRU/edit?usp=sharing

Unity Desktop Requirements:
https://docs.unity3d.com/Manual/system-requirements.html#desktop

ESRB Rating Guide:
https://www.esrb.org/ratings-guide/

Unity Coding Standards:
https://unity.com/how-to/organizing-your-project#code-standards

Contributions

Braeden Kurfman: Programming, Level design

Christian Gnat: 3D Models, Programming assistance

Joseph Wenger: Music and Sound, Programming assistance, ‘concept art’