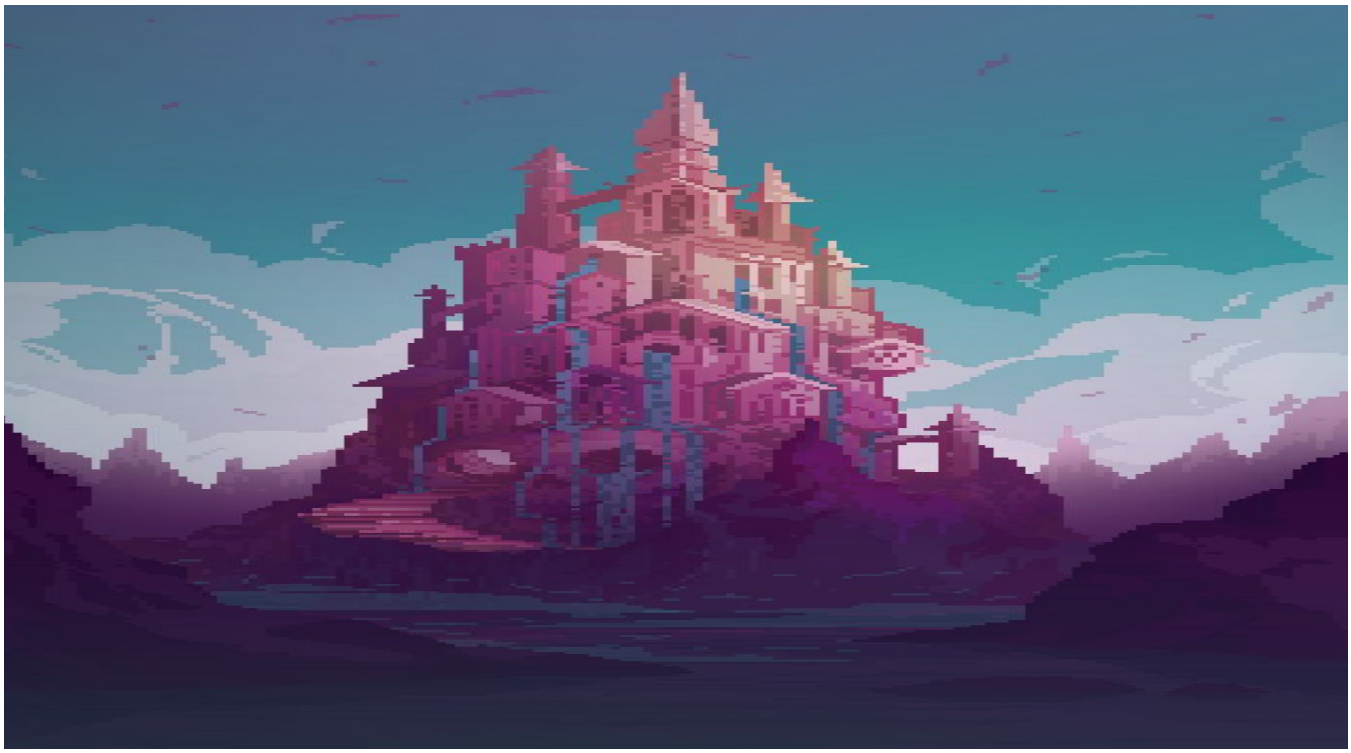
Invader of the Void

Testing and Inspection Report



A document of testing and inspection for Invaders of the Void

Prepared by
Cecilia Avila
Eugenio Perez
Kaveesha Weerasiri
Adrian Zavala
in CS 440
at the
University of Illinois Chicago

September 2018

Table of Contents

List of Figures

List of Tables

I Project Description

1 Project Overview

The project, “Invaders of the Void” is a 2D platformer puzzle game. It was inspired by Mario, Terraria, and Celeste to bring a 2D puzzle experience. It has five different levels to complete. Each level has its own theme and puzzle to solve with keys to unlock the door to the next level. The player starts in a space theme level, then transitions to a castle with four levels.

2 Project Domain

The domain of the project is video games and Mario and Celeste. It needed to have some puzzle aspects as well.

3 Relationship to Other Documents

This project is modeled after the Group 16 Final Design Document named “Invaders of the Void”

4 Naming Conventions and Definitions

4a Definitions of Key Terms

AI: Artificial Intelligence, the heuristics for an ingame object, and how it acts in the context of the current level.

Boss: a bigger version of the enemy that the user must deftly overcome.

Door: an ingame object that serves as a trigger when the player comes into contact with it. Assuming that the player has a key or certain conditions are met, it will automatically transition into the next level.

Enemy: an AI controlled object that serves to damage the player or send them back to the start of the level should they touch them.

Game object: refers to things that have multiple game components. They are more or less containers.

Game components: Things that add properties to game objects so that they can function in the actual game.

Lever: An interactable ingame object that causes an event to occur

Movement: How the user is able to control the player.

Player: A user controlled object that serves as their avatar in this game

Ray cast: A vector in that it projects in one direction. It has a line renderer component that makes it visible to the user..

Key: A game object that the player must collect before being able to interact with the door

Sconce: an ingame object that can hold torches, serving as a lever whenever a torch is placed or removed from it

Torch: An ingame object that illuminates its immediate radius while serving as a lever whenever placed or removed on a sconce.

Trampoline: An ingame object that is able to propel the player upward with varying force, usually allows them to reach places they otherwise wouldn't with their normal jump.

4b UML and Other Notation Used in This Document

N/A

4c Data Dictionary for Any Included Models

N/A

II Testing

1 Items to be Tested

1. Player Movement
2. Player Animation
3. General AI movement
4. Enemy movement left to right

5. Enemy animation
6. Platforms
7. Trampoline
8. Key pick-up
9. Unlock Door
10. Respawn of player
11. Music
12. Space: Unlock door (same as Unlock door, but has 3 keys)
13. Player Interaction: Lever & Build Mode
14. Minecraft/Terraria: Placement and Deletion of Blocks
15. Torch Puzzle:
16. Boss Battle: Laser animation and hit detection
17. Boss Battle: Bullets animation and hit detection
18. Boss Battle: Apple eating and animation
19. Boss Battle: Health Bar updates according to what happens
20. Boss Battle: Falling platform

2 Test Specifications

ID#3- General AI Movement

Description: AI tracks player and moves accordingly towards the player

Items covered by this test: player, enemy

Requirements addressed by this test: N/A

Environmental needs: Unity, Libraries: Pixel Adventure 1, Pixel Adventure 2, and Free Pixel Space Platform Pack, Medieval_pixel_art_asset_FREE

Intercase Dependencies: The A* map is placed correctly on the tilemap so that the bat can avoid the different platforms. The platforms will be in their own layer

Test Procedures: The A* map is set up. The bat has a seeker script as one of its game components. The distance that the AI should calculate and the loop are set up correctly

Input Specification: The distance to calculate the path is specified by the user. The player's position is given to the AI script. The transform point is specified for where the player will be spawned, enemy sprite

Output Specifications: The bat flies in a path towards the player. The enemy plays an animation and flips it's sprite based on how fast it is going one way

Pass/Fail Criteria: The player is tracked down 95% of the time as long as platforms are not in the way

ID#4- Enemy Movement Left to Right

Description: Common enemy moves left and right

Items covered by this test: Enemy: bat and ghost

Requirements addressed by this test: N/A

Environmental needs: Unity, Libraries: Pixel Adventure 1, Pixel Adventure 2, and Free Pixel Space Platform Pack, Medieval_pixel_art_asset_FREE

Intercase Dependencies: N/A

Test Procedures: Enemy

Input Specification: Float value

Output Specifications: Enemy moves left and right depending on float value

Pass/Fail Criteria: The enemy continuously moves left and right by the distance provided

ID#5- Enemy Animation

Description: Common enemy animate

Items covered by this test: Enemy: bat and ghost

Requirements addressed by this test: N/A

Environmental needs: Unity, Libraries: Pixel Adventure 1, Pixel Adventure 2, and Free Pixel Space Platform Pack, Medieval_pixel_art_asset_FREE

Intercase Dependencies: N/A

Test Procedures: Enemy

Input Specification: Float value

Output Specifications: Enemy moves left and right depending on float value

Pass/Fail Criteria: The enemy continuously moves left and right by the distance provided

ID#7- Trampoline

Description: Trampoline projects player upwards when the player hits it and plays an animation

Items covered by this test: Trampoline, player

Requirements addressed by this test: N/A

Environmental needs: Unity, Libraries: Pixel Adventure 1, Pixel Adventure 2, and Free Pixel Space Platform Pack, Medieval_pixel_art_asset_FREE

Intercase Dependencies: Trampoline is upright

Test Procedures: Trampoline gives the player rigidbody a force

Input Specification: Player hits the trampoline box collider 2d, player rigidbody, trampoline animation, force specified by the user

Output Specifications: Animation of the trampoline ends, player is lifted off the ground

Pass/Fail Criteria: The player is projected upward depending on the force inputted

ID#8- Key pick-up

Description: Player picks up key and key disappears

Items covered by this test: Player, key

Requirements addressed by this test: N/A

Environmental needs: Unity, Libraries: Pixel Adventure 1, Pixel Adventure 2, and Free Pixel Space Platform Pack, Medieval_pixel_art_asset_FREE

Intercase Dependencies: Player Movement

Test Procedures: Player needs to move to the key and make contact with it

Input Specification: Keyboard input to move the player: W, A, D

Output Specifications: The key should disappear

Pass/Fail Criteria: The key should disappear and the boolean value of "hasKey" should turn true.

ID#9- Unlock Door

Description: Moves player to the next scene.

Items covered by this test: Player, key

Requirements addressed by this test: N/A

Environmental needs: Unity, Libraries: Pixel Adventure 1, Pixel Adventure 2, and Free Pixel Space Platform Pack, Medieval_pixel_art_asset_FREE

Intercase Dependencies: Key pick-up

Test Procedures: Player picks up key, and makes contact with door

Input Specification: Keyboard input to move the player: W, A, D

Output Specifications: Player is transitioned to a new level

Pass/Fail Criteria: Player is transitioned to the next chronological order scene

ID#10 - Respawn of Player

Description: Move player back to initial start position for that level.

Items covered by this test: Player

Requirements addressed by this test: N/A

Environmental needs: Unity, Libraries: Pixel Adventure 1, Pixel Adventure 2, and Free Pixel Space Platform Pack, Medieval_pixel_art_asset_FREE

Intercase Dependencies: Player Movement

Test Procedures: Player must come into contact with an enemy, spikes, or water.

Input Specification: Player moves towards these objects with keyboard input: W, A, D.

Output Specifications: The player is moved to the beginning of level location

Pass/Fail Criteria: Player is moved to the beginning of level location when in contact with enemy, spikes, or water.

ID#11 - Music

Description: Music plays in the background of each scene and changes with the change of scene as well.

Items covered by this test: player camera

Requirements addressed by this test: N/A

Environmental needs: Unity, Libraries: Pixel Adventure 1, Pixel Adventure 2, and Free Pixel Space Platform Pack, Medieval_pixel_art_asset_FREE

Intercase Dependencies: Player Movement, Key pick-up, and Unlock door

Test Procedures: Pass into the next level to hear music transitions related to each scene.

Input Specification: Player movement: W, A, D

Output Specifications: A song transitions along with a level/scene transition and plays for that duration of that level.

Pass/Fail Criteria: When playing a level, there should be a song playing in the background, once you pass that level you transition to a new level and hear a new song for that level. Each level has its own theme song, along with the ending and game over scenes.

ID#12 - Space: Unlock door

Description: Space door transitions player to the next scene. Same logical test for door unlock, but with more keys

Items covered by this test: player, 3 crystal keys

Requirements addressed by this test: N/A

Environmental needs: Unity, Libraries: Pixel Adventure 1, Pixel Adventure 2, and Free Pixel Space Platform Pack, Medieval_pixel_art_asset_FREE

Intercase Dependencies: Player Movement, Key pick-up

Test Procedures: Player makes contact with 3 crystal keys

Input Specification: Player movement: W, A, D

Output Specifications: Player is transitioned to a new level

Pass/Fail Criteria: Player is transitioned to the next chronological order scene

ID#13 - Player Interaction: Lever & Build Mode

Description: Initiate player interaction with objects.

Items covered by this test:

Requirements addressed by this test: N/A

Environmental needs: Unity, Libraries: Pixel Adventure 1, Pixel Adventure 2, and Free Pixel Space Platform Pack, Medieval_pixel_art_asset_FREE

Intercase Dependencies: Player Movement

Test Procedures: Press "F" to initiate player interaction with lever in level two or to activate building mode if in level 3.

Input Specification: Keyboard input: F

Output Specifications: Allow interaction between objects in level 2, 3, and 4

Pass/Fail Criteria: Interaction with object not available if F was not pressed. If F is pressed then the player can pull the lever in level 2 and activate building mode for level 3. Lastly to place and remove a torch in level 4.

ID#14 - Placement and Deletion of Blocks

Description: Block placement for Minecraft/Terraria room.

Items covered by this test: The player can place a block by left mouse clicking and delete by right mouse clicking.

Requirements addressed by this test: N/A

Environmental needs: Unity, Libraries: Pixel Adventure 1, Pixel Adventure 2, and Free Pixel Space Platform Pack, Medieval_pixel_art_asset_FREE

Intercase Dependencies: Player Interaction

Test Procedures: Press "f" to activate building mode. With a mouse or mouse pad, left click anywhere in the black screen to place a block and right click anywhere where you had previously placed a block to delete it.

Input Specification: Boolean based, mouse click input from user

Output Specifications: A block is placed where the click was placed on the tilemap. If a block or object is currently at that location then you cannot place a block. A block is deleted where the click was placed on the tilemap. If a block is currently at that location then you can only delete that block.

Pass/Fail Criteria: Pass if a block can be placed at that clicked location on the tilemap if nothing is currently there, otherwise fail. Pass if a block can be deleted at that clicked location on the tilemap if a block is currently there, otherwise it should do nothing. Fail if it does not delete the clicked on block.

17# - Boss Battle: Laser animation and hit detection

Description: Laser shows, hits, and updates objects appropriately

Items covered by this test: Walls, player, falling platform, moving platform, apple

Requirements addressed by this test: N/A

Environmental needs: Unity, Libraries: Pixel Adventure 1, Pixel Adventure 2, and Free Pixel Space Platform Pack, Medieval_pixel_art_asset_FREE

Intercase Dependencies: Boss is at half health

Test Procedures: Get the enemy health and get the time since the last laser

Input Specification: Timer that is set to go off based on the seconds indicated by the user, the players position, the enemy rigidbody, the explosion gameobject

Output Specifications: The laser is shown through the use of a ray cast. If the player is hit, then you decrease the player's health and it shows a hit animation. If something else is hit, the laser stops and displays an animation

Pass/Fail Criteria: The laser is shown. The player takes damage if it is hit and the laser gets destroyed, playing an animation. Otherwise, the laser hits the object and stops, playing an animation

18# - Boss Battle: Bullets animation and hit detection

Description: Bullets fire from player, hits, and updates objects appropriately

Items covered by this test: Walls, enemy, falling platform, moving platform, apple

Requirements addressed by this test: N/A

Environmental needs: Unity, Libraries: Pixel Adventure 1, Pixel Adventure 2, and Free Pixel Space Platform Pack, Medieval_pixel_art_asset_FREE

Intercase Dependencies: player hits the spacebar

Test Procedures: The player hits the spacebar and the game recognizes that input

Input Specification: The players position, the player rigidbody, the explosion gameobject, the enemy health, the velocity of the bullet, the cooldown that is specified by the user

Output Specifications: The bullet is shown and moves across the screen based on velocity. The enemy's health is decreased if hit and it has a hit animation. The bullet animation is played when it hits any of the game objects in the scene.

Pass/Fail Criteria: The bullet is shown. The enemy takes damage if it is hit and the bullet gets destroyed, playing an animation. Otherwise, the bullet hits the object, gets destroyed, and plays an animation.

19# - Boss Battle: Apple eating and animation

Description: Is the apple bobbing up and down when the scene plays and can the player eat it

Items covered by this test: player health, collision detection, player

Requirements addressed by this test: N/A

Environmental needs: Unity, Libraries: Pixel Adventure 1, Pixel Adventure 2, and Free Pixel Space Platform Pack, Medieval_pixel_art_asset_FREE

Intercase Dependencies: N/A

Test Procedures: The animation works for the apple. The player can detect objects. The apple has a circle collider 2d

Input Specification: The position of the player, the sprite of the apple, the user inputted bonus for eating the apple that would be added to the player, the players health

Output Specifications: The player gains health after eating the apple. The apple stays in idle animation until the player eats it.

Pass/Fail Criteria: Apple is destroyed after the player touches it. Player health goes up. Player can only eat apple if below full health

20# - Boss Battle: Health Bar updates according to what happens

Description: Health bar for enemy and player updates accordingly

Items covered by this test: player, enemy

Requirements addressed by this test: N/A

Environmental needs: Unity, Libraries: Pixel Adventure 1, Pixel Adventure 2, and Free Pixel Space Platform Pack, Medieval_pixel_art_asset_FREE

Intercase Dependencies: Health bar must accurately update when pressing spacebar

Test Procedures: Player gets hit by the bat, bat gets hit by player bullets, player eats apples

Input Specification: Player health, Enemy Health, player sprite, enemy sprite, enemy color, player renderer, health bar slider, amount of damage that user specifies the player or the enemy will take

Output Specifications: player health UI decreases if hit by the enemy or the laser that it shoots**,** player health UI increases if the player eats the apple, enemy health UI decreases if hit by the players bullets

Pass/Fail Criteria: Player health bar UI decreases when hit by the enemy or it's laser. Player health bar UI increases when eating an apple. Enemy health bar UI decreases when hit by the player bullet.

21# - Boss Battle: Falling Platform

Description: Platform falls only when player is on it

Items covered by this test: player

Requirements addressed by this test: N/A

Environmental needs: Unity, Libraries: Pixel Adventure 1, Pixel Adventure 2, and Free Pixel Space Platform Pack, Medieval_pixel_art_asset_FREE

Intercase Dependencies: The rigidbody 2d body type is set to kinematic

Test Procedures: Player gets hit by the bat, bat gets hit by player bullets, player eats apples

Input Specification: The fall delay that the user specifies, the rigidbody of the platform, and the animator of it

Output Specifications: When the player lands on the platform, the platform drops and plays an animation

Pass/Fail Criteria: The platform is not kinematic anymore after the player stands on the platform. Animation is played for the propellers

3 Test Results

ID#1 - Player Movement

Staff conducting tests: Eugenio Perez and Adrian Zavala

Expected Results: Player will move according to input

Actual Results: Player moved according to input

Test Status: Pass

ID#2 - Player Animation

Staff conducting tests: Adrian Zavala

Expected Results: Player animation will change with input

Actual Results: Player animations change accordingly, but at times the player animations for jump will animate and then stop.

Test Status: Pass - but need to find out why animation stops

ID#3 - General AI Movement

Date(s) of Execution: 4/01/2020 - 4/20/2020

Staff conducting tests: Kaveesha

Expected Results: Does the bat move towards the player, no matter its position (unless there is a moving platform, in which the route does not calculate fast enough for it to track down the player).

Actual Results: Bat moves toward the player when it can no matter it's position

Test Status: Pass

ID#4 - Enemy Movement left to right

Date(s) of Execution: 2/20/2020 - 4/20/2020

Staff conducting tests: Cecilia, Adrian, Kaveesha

Expected Results: Enemy will move on its own across the room from left to right.

Actual Results: Enemy moved from left to right across the room the entire time.

Test Status: Pass

ID#5 - Enemy Animation

Date(s) of Execution: 4/03/2020 - 4/18/2020

Staff conducting tests: Kaveesha, Adrian, Cecilia

Expected Results: The enemy continues to play animations even when it is hit or obstructed by something

Actual Results: Enemy continues to do the correct animation regardless of what is happening on screen

Test Status: Pass

ID#6 - Platforms

Date(s) of Execution: 4/01/2020 - 4/20/2020

Staff conducting tests: Eugenio, Kaveesha

Expected Results: The player does not move through the platform when it lands or stands on it and the enemies also does not move through the platforms

Actual Results: Neither the player nor the enemy can go through the platform

Test Status: Pass

ID#7 - Trampoline

Date(s) of Execution: 4/01/2020 - 4/20/2020

Staff conducting tests: Kaveesha

Expected Results: Trampoline shoots player in an upward trajectory given the inputted force

Actual Results: Player flies up as a result of the player hitting the trampoline

Test Status: Pass

ID#8 - Key pick-up

Date(s) of Execution: Tested at each level

Staff conducting tests: Cecilia Avila, Eugenio Perez, Kaveesha Weerasiri, Adrian Zavala

Expected Results: Key will disappear

Actual Results: Key disappeared and boolean turned true

Test Status: Pass

ID#9 - Unlock Door

Date(s) of Execution: Tested at each level

Staff conducting tests: Cecilia Avila, Eugenio Perez, Kaveesha Weerasiri, Adrian Zavala

Expected Results: Transition to next level

Actual Results: Transition was correct

Test Status: Pass

ID#10 - Respawn of Player

Date(s) of Execution:4/10/2020 - 4/20/2020

Staff conducting tests: Cecilia Avila, Adrian Zavala, Eugenio Perez, Kaveesha Weerasiri

Expected Results: Player will get placed at its initial spot from when the level begins every time it is in contact with enemies, spikes, or water.

Actual Results: Player is respawned to the correct location when in contact with specified objects

Test Status: Pass

ID#11 - Music

Date(s) of Execution: 4/05/2020 - 4/20/2020

Staff conducting tests: Cecilia, Adrian, Eugenio, Kaveesha

Expected Results: Music plays in all levels and transitions to certain songs according to level

Actual Results: Music plays in all scenes, including game over scene and winning scene. Music also matches the specified room it should be playing for.

Test Status: Pass

ID#12 - Space: Unlock Door

Date(s) of Execution: 04/03/2020 - 4/05/2020

Staff conducting tests: Cecilia, Adrian, Eugenio, Kaveesha

Expected Results: The player will collect all the crystal gems and then proceed to the door. Once the player gets to the door, it transitions the player to the next level.

Actual Results: Works correctly and transitions the player to the next level.

Test Status: Pass

ID#13 - Player Interaction

Date(s) of Execution: 04/01/2020 - 4/09/2020

Staff conducting tests: Cecilia Avila, Eugenio Perez

Expected Results: When in rooms: 2,3,4 then when pressing F player interaction should be allowed with objects like lever, building mode and torch. If at any other level, user input F should do nothing.

Actual Results: You can only use F for a specified room and functionalities don't mix. In specific, you can only pull the lever in level 2, can only activate building mode in room 3, etc.

Test Status: Pass

ID#14 - Minecraft/Terraria: Placement and Deletion of Blocks

Date(s) of Execution: 04/03/2020 - 4/05/2020

Staff conducting tests: Cecilia Avila

Expected Results: Place a couple of blocks randomly on the tilemap if space is empty by clicking the left mouse button. Then delete those blocks with the right mouse button.

Actual Results: Clicking on the left mouse button creates blocks in open space. If you try to place it over an existing object, it does not let you. You can only delete a block if you click on an existing block the player created.

Test Status: Pass

ID#15 -Boss Battle: Laser animation and hit detection

Date(s) of Execution: 4/10/2020 - 4/20/2020

Staff conducting tests: Kaveesha

Expected Results: The laser shows up. The laser stops moving and plays an animation when it hits an object in the scene.

Actual Results: The laser shows up and the animation plays on the scene

Test Status: Pass

ID#16 -Boss Battle: Bullet animation and hit detection

Date(s) of Execution: 4/10/2020 - 4/20/2020

Staff conducting tests: Kaveesha

Expected Results: The bullet moves in the direction the player is facing. When the bullet hits any object, it destroys itself and plays an animation

Actual Results: The bullet goes in the direction the player is facing, destroys itself and plays an animation

Test Status: pass

ID#17 -Boss Battle: Apple eating and animation

Date(s) of Execution: 4/10/2020 - 4/20/2020

Staff conducting tests: Kaveesha

Expected Results: The apple plays its idle animation. It destroys itself and gives the player health when the player is under full health

Actual Results: The animation plays when idle. It destroys itself and gives the player health when the conditions are right

Test Status: Pass

ID#18 -Boss Battle: Health bar updates according to what happens

Date(s) of Execution: 4/01/2020 - 4/20/2020

Staff conducting tests: Kaveesha

Expected Results: Health UI of the player decreases if hit by the enemy or the laser that it shoots**,** player health UI increases if the player eats the apple, enemy health UI decreases if hit by the players bullets

Actual Results: The same as expected results

Test Status: Pass

ID#19 -Boss Battle: Falling Platform

Date(s) of Execution: 4/07/2020 - 4/20/2020

Staff conducting tests: Kaveesha

Expected Results: Platform falls when the player stands on it and the enemy does not trigger the fall. Animation plays when platform falls

Actual Results: Same as expected

Test Status: Pass

ID#20 - Player can remove torch from sconce

Date(s) of Execution: 4/07/2020 - 4/20/2020

Staff conducting tests: Eugenio

Expected Results: When player presses 'F' over a sconce with a torch, **and** the player **isn't** holding a torch, the torch is removed from the sconce and placed in the player's hand

Actual Results: torch is properly removed from sconce and placed in player's hand

Test Status: Pass

ID#21 - Player can place torch on empty sconce

Date(s) of Execution: 4/07/2020 - 4/20/2020

Staff conducting tests: Eugenio

Expected Results: When player presses 'F' over an empty sconce, **and** the player **is** holding a torch, the torch is removed from the player's hand and placed in the sconce

Actual Results: torch is properly removed from player's hand and placed in player's hand

Test Status: Pass

4 Regression Testing

Anything that is working in our game had to have worked beforehand otherwise all of the different mechanics would not work correctly. Once we had finished the initial room, we settled on the workflow, and agreed to leave as many scripts as unchanged as possible so we could maximize compatibility when merging code in future sprints. Each level uses the scripts from the first level, as well as a collection of scripts of its own, as such, this means most levels have their own testing suite.

That being said, one script that did see some changes during development was the movement script. This was tested each time we loaded into a room. It was crucial we made sure movement worked, because it serves as the foundation for all of our levels. Without movement, there is no game.

ID#1 - Player Movement

Description: General movement for all players as the player was re-used for each scene.

Items covered by this test: Player

Requirements addressed by this test: N/A

Environmental needs: Unity, Libraries: Pixel Adventure 1, Pixel Adventure 2, and Free Pixel Space Platform Pack, Medieval_pixel_art_asset_FREE

Intercase Dependencies: This test should be successful before testing: Player Animation

Test Procedures: Press on keyboard. Press ‘D’ for right movement. Press ‘A’ for left movement. Press ‘W’ for jump movement.

Input Specification: Input from keyboard W, A, and D

Output Specifications: D: player moves the right while pressing, A: player moves the to left while pressing, W: player “jumps”/moves upwards then “falls”/moves back to place.

Pass/Fail Criteria: Player moves according to the input

ID#2 - Player Animation

Description: Animation for the all players across all levels.

Items covered by this test: Player

Requirements addressed by this test: N/A

Environmental needs: Unity, Libraries: Pixel Adventure 1, Pixel Adventure 2, and Free Pixel Space Platform Pack, Medieval_pixel_art_asset_FREE

Intercase Dependencies: Player Movement

Test Procedures: Press on keyboard. Press 'D' for right animation. Press 'A' for left animation. Press 'W' for jump animation.

Input Specification: Input from keyboard W, A, and D

Output Specifications: D: player moves and animates the right while pressing, A: player moves and animates the to left while pressing, W: player "jumps"/moves upwards and animates then "falls"/moves and animates back to place.

Pass/Fail Criteria: Player moves and animates according to the input

ID#3 - Darkness Room: Room properly goes dark when user grabs key

Description: When the user grabs the key from the circuit breaker, the "power goes out" and the room becomes enveloped in darkness.

Items covered by this test: Player

Requirements addressed by this test: N/A

Environmental needs: Unity, Libraries: Pixel Adventure 1, Pixel Adventure 2, and Free Pixel Space Platform Pack, Medieval_pixel_art_asset_FREE

Intercase Dependencies: Player Movement, ID#20, ID#21

Test Procedures: When the user comes into contact with the key in the room, the room becomes mostly black, with only the player, and two torches being visible in the room.

Input Specification: Player jumps when under the key to grab it

Output Specifications: a Black square will have it's layer promoted so that it covers the entire level, giving the illusion that the room is covered in total darkness. The only things that will be visible will be the player, and any torches in the player's view.

Pass/Fail Criteria: The room goes dark, and the torches properly illuminate a small radius.

III Inspection

1 Items to be Inspected

Adrian's item:

- playermovement.cs
  - Ensure the player can move right and animate
  - Ensure the player can move left and animate
  - Ensure the player can jump up and animate
  - Ensure that the playermovement.cs is communicating with Unity for animation effects

Cecilia's item:

Eugenio's item:

- torchInteract.cs
  - Ensure player is able to place and remove torch from sconce
  - Ensure player isn't able to replicate an infinite number of torches
  - Ensure an array of blocks moves upon interacting with the torchSconce

Kaveesha's item:

These scripts all interact with each other, so I thought it would be good to inspect all of them together. It's focus is only for the bullet and health aspects of these scripts

- bullet.cs, weapon.cs, healthbar.cs, enemyHealth.cs
  - ensure player is limited to firing bullets every .3 seconds
  - ensure bullet is instantiated from the way that the player is facing and that the bullet is applied the speed given by the user
  - ensure that when the enemy is hit, the enemy takes damage
  - ensure that when the enemy takes damage, the enemy's health bar decreases by a certain value given by the user
  - ensure that the enemy sprite turns red, the enemy shoots lasers, and the enemy becomes faster after the enemy gets to half health
  - ensure that the bullet plays the explosion animation and is destroyed when it hits a certain object

2 Inspection Procedures

Meetings were held weekly to assess progress and go over action items. Bugs that were found were often lumped into these action items, but for the most part testing was often done passively. There was never a backlog of bugs that justified the existence of some sort of test management system. Anytime one of us discovered a bug in each other's levels, we made it known in the groupchat. That being said, we each made a "Testing Procedure" for the last stretch of the semester, and supplied an item that should be tested exhaustively. This section contains the results of our findings, although at this point in development, most bugs had been ironed out.

Adrian's item Procedure:

1. In Unity, load any room as the player movement script is used for every player
2. Load "Adrian"
3. Press play
4. To ensure movement of player press the W, A, D keys
    a. ensure W makes player jump, and character animation should jump
    b. ensure A makes player move left and character animation should run
    c. ensure D makes player move right and character animation should run
5. If the character are animating correctly then the player movement script is communicating properly with Unity to display the correct animations

Cecilia's item Procedure:

1. In Unity, load up "Cecilia"
2. Press play
3. Press 'F' to turn on building mode
    a. Left-click on mouse or mouse pad to place a block on black background
        i. ensure block was placed at the location of the click
        ii. ensure block was not allowed to be placed on top of an existing block or object
        iii. ensure one block is placed per click
    b. Right-click on mouse or mouse pad to delete an existing block
        i. ensure block exists at the location of the click to delete it
        ii. ensure one block is deleted per click
        iii. ensure that nothing happens if click happens on empty space
4. Place blocks above water and in front of the ghost to avoid getting respawned
5. Place blocks to reach the key and create a path to the door to move on to the next level

Eugenio's item Procedure:

1. In Unity, load up the "DarknessRoom" scene
2. Press play
3. Press 'F' over the first torch you see
    a. ensure player is holding torch in his hand
    b. ensure torch sconce is empty
    c. ensure that the array of tiles blocking your path have moved
    d. ensure player doesn't slow down when walking over torch's "trigger area"
4. Repeat for all torch sconces you see (some tiles that move are located off screen relative to where you interact with the sconce).
5. Finish the level Once all sconces are tested.

Kaveesha's item Procedure:

1. In Unity, load up the "Boss Battle" scene
2. Press play
3. Press the spacebar to shoot bullets
    a. Ensure bullets stop on all surfaces that it hits
    b. Ensure bullets are destroyed on impact
    c. Ensure boss gets its health depleted when hit
    d. Ensure bullet hit animation plays
    e. Ensure that enemy's sprite changes, its UI health bar changes, its speed, and it shoots lasers at the player when the enemy gets to half health
    f. Ensure bullet does not spawn behind the player
4. Continue to shoot bullets at the enemy until the enemy is depleted of all of its health
5. If the enemy depletes the player's health either through touching the player or its laser hitting it, then display the game over screen
6. Otherwise, finish the level by picking up the key that is placed within the users reach and enter the door to win

3 Inspection Results

Invader of the Void is designed in such a way that it is convenient to test each and every level all in one run. Everyone tested everyone's code at one point or another during development. That being said, here are the results of the more focused tests that were created after the 2nd release.

Inspection #1

Procedure: Adrian's item

Inspected by: Kaveesha

Date of procedure: 4/22/2020

Notable Discoveries: A referenced script was missing from the player gameobject, but it was an easy fix. I removed the script from the players components

Inspection #2

Procedure: Cecilia's item

Inspected by: Eugenio

Date of procedure: 4/21/20

Notable Discoveries: Item passed inspection, and yielded no bugs

Inspection #3

Procedure: Eugenio's item

Inspected by: Cecilia

Date of procedure: 4/24/2020

Notable Discoveries: Items passed and did not encounter bugs

Inspection #4

Procedure: Kaveesha's item

Inspected by: Adrian

Date of procedure: 4/24/2020

Notable Discoveries: All items pass flawlessly and no bugs were found

IV Recommendations and Conclusions

As stated prior under III.2, testing was often done passively, and most bugs were squashed early in development. As a result, all inspections resulted in positive results, and no bug fixes are needed.

V Project Issues

1 Open Issues

Issue #1

Null Pointer exceptions do not seem to show up anymore in the console.This may affect the product later where the product just crashes unexpectedly

Not updating the library past a certain point may become a detriment as some of the things that we are using are from older versions of both the Unity and the A* pathfinding project.

Issue #2

Player horizontal running animation is no longer being triggered

2 Waiting Room

Requirement 4b

Going off original mario pixel placement, the character will not just remain in the center of the screen but shall move no further head of the 112 parallel pixel on the screen fixing its position.

The timer should be accurate up to a tenth of a second

Notes about Requirement 4b: We were unable to implement a camera with parallax, and opted to make the camera fixed on the middle of the screen, the effort required to add parallax was better suited to other requirements that were more crucial to functionality.

Test case in IV.6

Test Timer- Make sure timer works

Notes about Test Case in IV.6: We did not have time to implement a functional timer in the final game.

Requirement 8e

The game shall have a partially-sighted mode to make it more accessible

for users with poor eyesight. The game shall include a color-blind mode.

Notes about Requirement 8e: We wanted to focus on fleshing out the functionality of the game, but these are important to make sure that a wide range of users are happy with our game and will continue to use it in the future.

3 Ideas for Solutions

Solution for Requirement 4b

The solution definitely requires multithreading of some kind. The timer cannot operate in the main thread, as it will crash the game. TorchFlex.cs employs a very basic worker thread that ensures a loop runs every second rather than every frame, so we could look there for a solution on how to get a timer to properly count the seconds one is playing a game.

There is also the question of how to save the time variable across multiple scenes when the user transitions levels.

Solution for trampoline working at all angles

The solution would require accessing the z axis of the trampoline so that we do not hard code that value and have if statements. The force that is applied to the player should be calculated from that z axis angle.

4 Project Retrospective

At the start of the project, there was much debate over which language/framework to use for Invader of the Void. Straight java, libGDX, Phaser, node.js...In the end, we came to the conclusion that while these languages might lend themselves well to a static application, it would be like trying to screw a nail with a hammer when trying to design a game. We opted to use Unity because it is an engine designed to make games and it has lots of tutorials and documentation which makes it easier for us to develop without prior experience.

Besides the steep learning curve attached to understanding the inner workings of Unity, there was also great turmoil in trying to have version control in the free version of the app. We needed to make sure that everyone was using the same version of Unity, the same assets, the same libraries, or risk “exploding” the project. Once that

was all said and done, we designed a test room that would create a basis for the work that we would do for the other levels. The knowledge gained from creating that test room led the way to our individual ideas.

We opted to each create individual levels that were cohesive, but loosely coupled to expedite merging. When we tried merging code in the first release, it ended up being easier just dragging and dropping each other's files into one main branch, which was not ideal for the long run.

Unity often liked to stop working for no reason. This led to many headaches that often had esoteric fixes that nigh worked consistently. Switching branches, deleting all your local files, coupled with the ages it took Unity to load a project. There were many moments during development where we pondered where we'd be using another language or framework, but in the end, each engine carries its own strengths and weaknesses, and simply put, Unity's strengths greatly outweighed the inconveniences that occurred during development.

Although this project was developed during the height of the COVID-19 pandemic, we were able to maintain productivity in spite of having to work remotely. Although morale and motivation were sometimes impeded, due dates always prevail. We believe that the fruits of our labor were worth it. Although there were many trials and tribulations, our TA made a point to mention that this project was his "favorite one" out of all the ones that were demoed to him. There is no award for that, but it is still nice to know.

VI Glossary

AI: Artificial Intelligence, the heuristics for an ingame object, and how it acts in the context of the current level.

Boss: a bigger version of the enemy that the user must deftly overcome.

Blocks: squares that the player can collide with and use as platforms.

Darkness: An area that is not visible to the player unless under a light source

Door: an ingame object that serves as a trigger when the player comes into contact with it. Assuming that the player has a key or certain conditions are met, it will automatically transition into the next level.

Enemy: an AI controlled object that serves to damage the player or send them back to the start of the level should they touch them.

Game object: refers to things that have multiple game components. They are more or less containers.

Game components: Things that add properties to game objects so that they can function in the actual game.

Lever: An interactable ingame object that causes an event to occur

Movement: How the user is able to control the player.

Player: A user controlled object that serves as their avatar in this game

Ray cast: A vector in that it projects in one direction. It has a line renderer component that makes it visible to the user..

Key: A game object that the player must collect before being able to interact with the door

Sconce: an ingame object that can hold torches, serving as a lever whenever a torch is placed or removed from it

Script: A sequence of instructions for a particular object.

Sprite: A picture (usually in .png or .jpg format) that is used to give a visual representation to game objects.

Sprite Mask: An object that is used to hide or reveal a sprite.

Torch: An ingame object that illuminates its immediate radius while serving as a lever whenever placed or removed on a sconce.

Unity: A Game engine used to make video games

Trampoline: An ingame object that is able to propel the player upward with varying force, usually allows them to reach places they otherwise wouldn't with their normal jump.

VII References / Bibliography

[1] Abdul Rehman, Anatoly Tverdovsky, Cesar Lazcano, Curt Thieme, Group 16 Final Design Document, 2018

[2] Flynt, John P., and Omar Salem. Software Engineering for Game Developers. Thomson/Course Technology, 2005.

VIII Index

Sprite Mask	29