Software Requirement Specification

Tower of Hanoi

Prepared by Diego Vela and Alyssa Barrientos

September 9, 2024

Table of Contents

Purpose	3
Introduction	3
Team Roles	3
Scripts	3
Description of Major Interactions in Gameplay	5
Excel Sheet Link(Sources, Bugs, and Backlog)	6

Purpose

The purpose of this project is to familiarize the team with the tools and features of unity while designing a simple tower of hanoi game.

Introduction

Donut on a Stick is a simple tower of hanoi game that uses a donut theme to provide a user-friendly experience. The game will display three poles, with the leftmost pole having a tower of donuts organized in decreasing order by size from bottom to top.

The objective of the game is to take all the rings in the left pole to the right pole in the same order. The rings in each pole must always be in decreasing order from the bottom and will not allow the player otherwise.

The game will include two scenes: a title screen and a gameplay screen. It will also include an exit menu that allows the player to quit the game when in the gameplay screen.

Team Roles

Developers[2]: In charge of the whole project from start to finish including design, documentation, and testing.

Scripts

Disk: Detects disk collisions and disk size and sends data to verify game state.

DiskSpawner: Handles disk spawning at the beginning of the scene and sets win conditions.

DragAndDrop2D: Handles the drag and drop function in the scene and prevents dragging through other objects with collisions. Created with AI.

DragAndDrop(Diego): Diego’s version of the drag and drop function. Works the same but it allows for dragging through obstacles. Not used in this version but it was used as the basis for DragAndDrop2D.

EscMenuController: Controls the escape menu panel when the user presses the esc key.

InvalidTowerCheck: Handles valid move logic using a stack and has a call to the LoseMenuController.

LoseMenuController: Handles the lose menu panel when an invalid move is detected.

MainMenu: Handles all button logic on the title screen.

SetUpController: Handles the setUp Panel that allows the user to choose how many disks to spawn in.

WinCondition: Sets and checks the win condition on the game and has a call to WinMenuController.

WinMenuController: Handles the Win Menu Panel when the user wins the game.

Game Structure

TitleScreen: This scene handles the title menu and allows the user to ‘Play’ or ‘Quit’ the game.

1. TitleMenuCanvas: Contains all Options in opening menu
    a. BackgroundImage
    b. GameName
    c. MainMenu
        i. QuitButton
            1. MainMenu Script
            2. Quits Game
        ii. PlayButton
            1. MainMeni Script
            2. Plays Game

GamplayScene: This scene is the main gameplay in which the user must get all the disks to the right tower but the disks must always be placed in descending order.

2. MenuContainer: Contains all the menu panels
    a. SetUpCanvas

- i. SetUpController Script
- ii. Displays the SetUpPanel

- b. EscMenuCanvas
  - i. EscMenuController Script
  - ii. Displays the EscMenuPanel
- c. WinMenuCanvas
  - i. WinMenuController Script
  - ii. Displays the WinMenuPanel
- d. LoseMenuCanvas
  - i. LoseMenuController Script
  - ii. Displays the LoseMenuPanel

3. StructureContainerCanvas
   - a. OuterBounds
     - i. Contains the base and off-screen edges of the scene.
   - b. Towers
     - i. Contains 3 tower objects, each with InvalidTowerCheck Script. ‘isTrigger’ objects.
   - c. TowerCollisions
     - i. Contains 3 towerC objects that collide with the disks.
4. RingContainer
   - a. DiskSpawner Script
   - b. Contains 8 square objects that are the disks for gameplay.
     - i. Disk Script, DragAndDrop2D Script
     - ii. Each disk has two child objects that are the outer bounds of the disk that will collide with the towerC objects.
5. WinCondition
   - a. WinCondition Script

Description of Major Interactions in Gameplay

Disk Drag&Drop: Basically when the user clicks a disk, it will get the position of the disk and move the position of the disk. Whenever the disk collides with another object, it will lock either/both the horizontal and/or vertical transformations so that you cannot drag it through a structure

TowerStack: Whenever a diskCenter object enters a collision with a tower object, the tower will add the size of the disk to a stack list. If the list is non-empty, it will compare the size of the top of the stack, and the disk that just entered to check if the move is valid or not.

Excel Sheet Link

TowerOfHanoi_CPSC386_HW1

Sources Sheet: Shows all outside sources used and how.

KnownBugs Sheet: Shows all found bugs during testing

Backlog: Shows work schedule and contributions of team members