Thievery Use Manual

Equipment Needed

- Minimum:
    - Oculus Rift headset
    - Oculus left controller
    - 2 Oculus cameras
    - 2.5m^2 space
- Extra:
    - 3rd camera
        - Used for more precise rotation especially when turning away from 2 front cameras

Setup

1) Setup equipment
2) Setup oculus full setup configurations
    a) This is important more specifically when the setup asks for your height as that is taken into account when playing the game

Instructions

Action	Button Execution	Description
Teleport	1) Hold grip button 2) Point downwards until a green raycast laser appears 3) Let go of grip button 4) If a red laser appears then you cannot teleport to specified area	Teleport to specified point when laser turns green
Move Incrementally	1) Move analog stick upwards (forward movement) 2) Move analog stick rearwards (backwards movement)	Move more precisely if need be (ex: get more close to a table to avoid hitting real world barriers)

Turn Incrementally	1) Move analog stick leftwards (left turn) 2) Move analog stick rightwards (right turn)	Turn in increments to avoid spinning around if it's more comfortable to the user
Crouching	Physically crouch down	Player crouches in the game in order to dampen player sound and to reduce player visibility
Grabbing (Door)	1) Move hand (gray sphere) towards door and **intersect** with its handle or door frame 2) Hold index button 3) Swing door open like a traditional door 4) Note: you do not need to step back as the door will phase through you	Open a door
Grabbing (Loot)	1) Move hand (gray sphere) towards loot and **intersect** with it 2) Hold index button 3) Pull loot towards your chest 4) Loot will disappear and add to your score	Grab loot
Grabbing (Chest)	1) Move hand (gray sphere) towards chest and **intersect** with its handle or top frame 2) Hold index button 3) Swing chest open like a chest door	Open a chest

How To Start

1) Start the .exe titled "Thievery VR"
2) Wait until you notice a networking GUI located in the top left corner
3) Click or the button that says "Host (H)" or press H
4) Enjoy

Goal

1) Steal at least 1000 loot
2) After your score is 1000 or more, head back to the front door that connects to the lobby and go through it
3) Every time you get caught, you lose a life. Lose all 3 lives and you lose the game

Scripts

Enemies

EnemyAI.cs

- The AI algorithm for the enemy
- Patrolling: enemy is patrolling and does not hear/see/barely hear/barely hear the player
- Suspicious: enemy has barely heard or barely seen the player and is suspicious of contact. 2 timers will count down. If the first timer counts down to 0 first then go back to patrolling. This timer gets reset every time a contact occurs. If the second timer counts down then go to investigative state
- Investigative: enemy will go to last contact point of player. A timer is counting down and if it reaches 0 then go back to patrolling. This state triggers from patrolling state if player is heard by the enemy
- Alert: Enemy has seen the player and running towards them. Any contact with the enemy updates the last player position and will go towards that position. A timer starts and if it reaches 0, then go back to patrolling
- This AI's navigation is using unity's navmesh agent system
- Plays the appropriate sound whenever a state is entered

EnemyFootController.cs

- Detects what the enemy is stepping on and plays the appropriate sound of the floor (ex: wood flooring)

EnemyHearing.cs

- The collider that checks whether the player hears or barely hears the player depending on the distance to the collider

EnemySight.cs

- The collider that checks whether the player sees or barely sees the player depending on the distance to the collider

PatrolPathNode.cs

- The next path of the patrolling path for the enemy
- Raycast a green line to the next path node in edit mode

Game State

GameStateController.cs

- Updates the score of the player
- Updates the players lives
- Checks if player has won or lost

GameWinRegion.cs

- Checks if player is colliding with the game win region. If they have enough loot then it tells the game state controller that the player can win now

Mechanics

Loot.cs

- Stores the score of the loot
- If it gets taken by the player then increase the players score

Door.cs

- Handles how the physics of the door is being calculated when the player is using it

LightIndicator.cs

- A collider is attached to the light and triggers code when the player touches it
- Tells the player how close they are to the light which is later calculated to find player brightness level

GUI

Fader.cs

- Fades a black screen in and out depending on the game state

HealthSystem.cs

- Checks with the game state controller to know how much live the player has. Once it has that information then display the health gui onto the canvas

BrightnessIndicator.cs

- Displays the appropriate brightness level of the player

LoudnessIndicator.cs

- Displays the appropriate loudness level of the player

StandingCrouchingIndicator.cs

- Displays an image to indicate if the player is standing or crouching

Player

PlayerController.cs

- Gives physics to the player
- Allows for movement to the player
- Allows for rotation to the player
- Updates the main cameras world from the VR headset local position
- Checks if the player is crouching or standing
- Determines the brightness level of the player
- Determines the loudness level of the player
- Sends the player to the start of the game when caught
- Allows the player to restart granted they have enough lives

PlayerFoot.cs

- Check to see what the player is standing on

PlayerFootController.cs

- Alternates between left and right foot when moving. The speed of the colliders cycling between each other is correlated to the speed of the player

PlayerHand.cs

- Checks to see what it's grabbing
    - If it's loot then create a joint between the hand and the loot. That gets destroyed once the loot has been acquired
    - If it's a door or a chest, then call the door script to attach or detach to the hand
- Allows the player to teleport
    - If the requested teleportation spot is invalid and/or too far away then display a red line renderer
    - If the requested teleportation spot is valid and close then display a green line renderer and teleport the player there

PlayerSoundMixer.cs

- Plays the appropriate sound that reflects on what surface the player is currently walking on

SwagBag.cs

- Checks to see if loot collides with it. If it does then add to the score with the loot score

Special Assets

Sounds

- All the sounds are taken from a game called Thief: The Dark Project
- It's important to have it since it contributes significantly to the immersion of the game
- It also tells the player where a guard is and what state they're currently in

How to Extend

Dynamic Sound Maps

- In the game Thief: The Dark Project, a sound map was created with respect to the structure of the buildings and their materials
- This is then used to emit sound more realistically which then adds more immersion and expected results from the guards senses

Realistic Visibility:

- Add a better way to calculate how bright a player is by not only taking in on how close they are to the light source but the intensity of the light, whether the player is in the shadows, if the player is standing/crouching, etc…

Items:

- By adding items into the game, the complexion of the game goes up but it allows for more mechanics to be introduced and more ways to play the game
- For example:
    - A lockpicking set that the player can use in order to pick through doors
    - A blackjack that disables guards if hit at the right spot