

Kevin's Tests Scripts

Complete Documentation & Hookup Guide

Guide for Unity 6000.3.0f1

Private Eye Pierce Demo No. 1

Table of Contents

1. Introduction and File Structure
2. Before You Begin: Creating Event Flags
3. Dialogue Folder Scripts
4. Flags Folder Scripts
5. HUD Folder Scripts
6. Interaction Folder Scripts
7. Player Folder Scripts
8. Puzzles Folder Scripts
9. Rooms Folder Scripts
10. Demo-Specific Folder Scripts
11. Audio
12. Complete Demo Walkthrough
13. Troubleshooting

1. Introduction, File Structure, & Animations

Welcome! This guide will teach you how to use the Kevin Tests scripts to build stuff in **Private Eye Pierce**.

Your Folder Structure

All the scripts live inside: **Assets > Scripts > Kevin Tests**

Inside that folder, you'll find these subfolders:

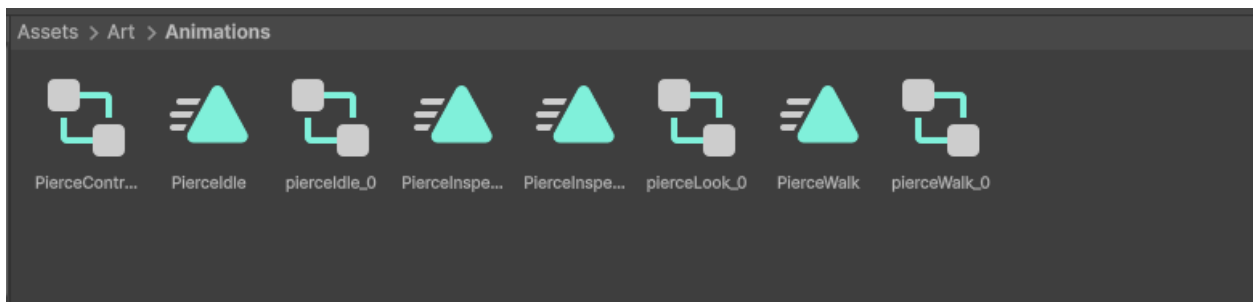
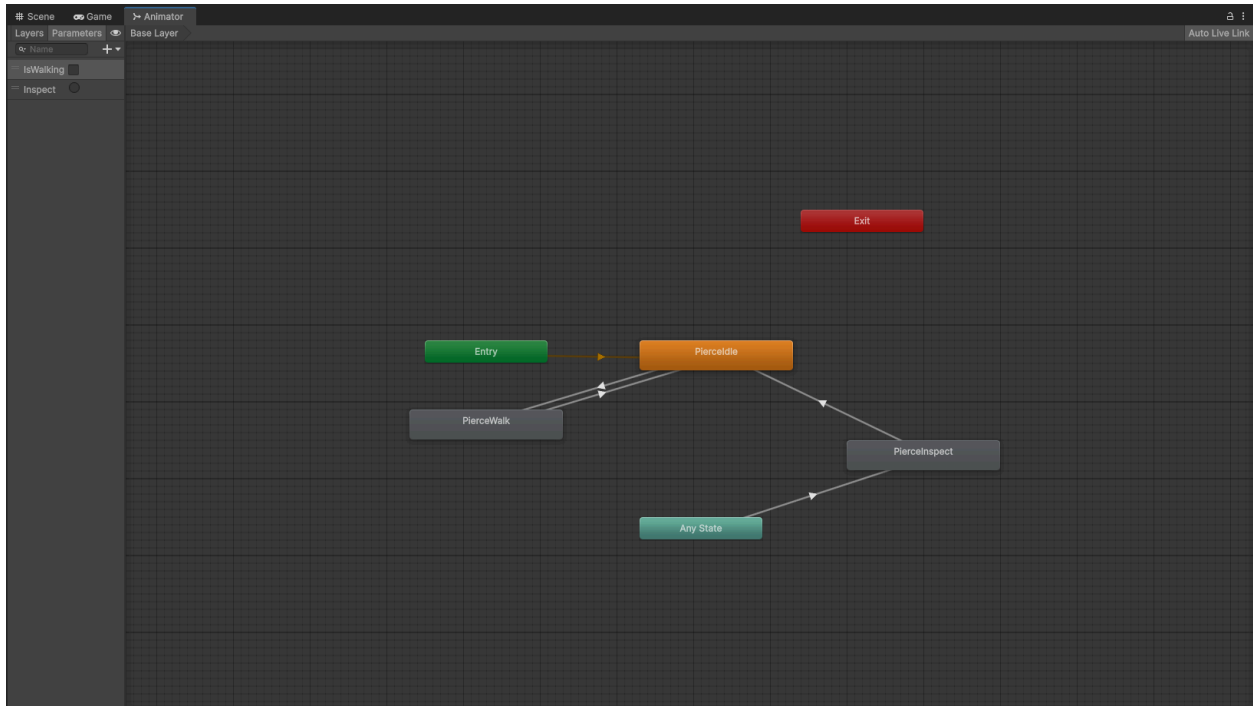
Folder	What's Inside
Demo-Specific	Scripts only used for the demo ending
Dialogue	Scripts that make characters talk
Flags	Scripts that remember yes/no information
HUD	Scripts for on-screen displays (cursor, highlights)
Interaction	Scripts for clicking on things
Player	Scripts that control the player character
Puzzles	Scripts for game puzzles
Rooms	Scripts for different areas/rooms

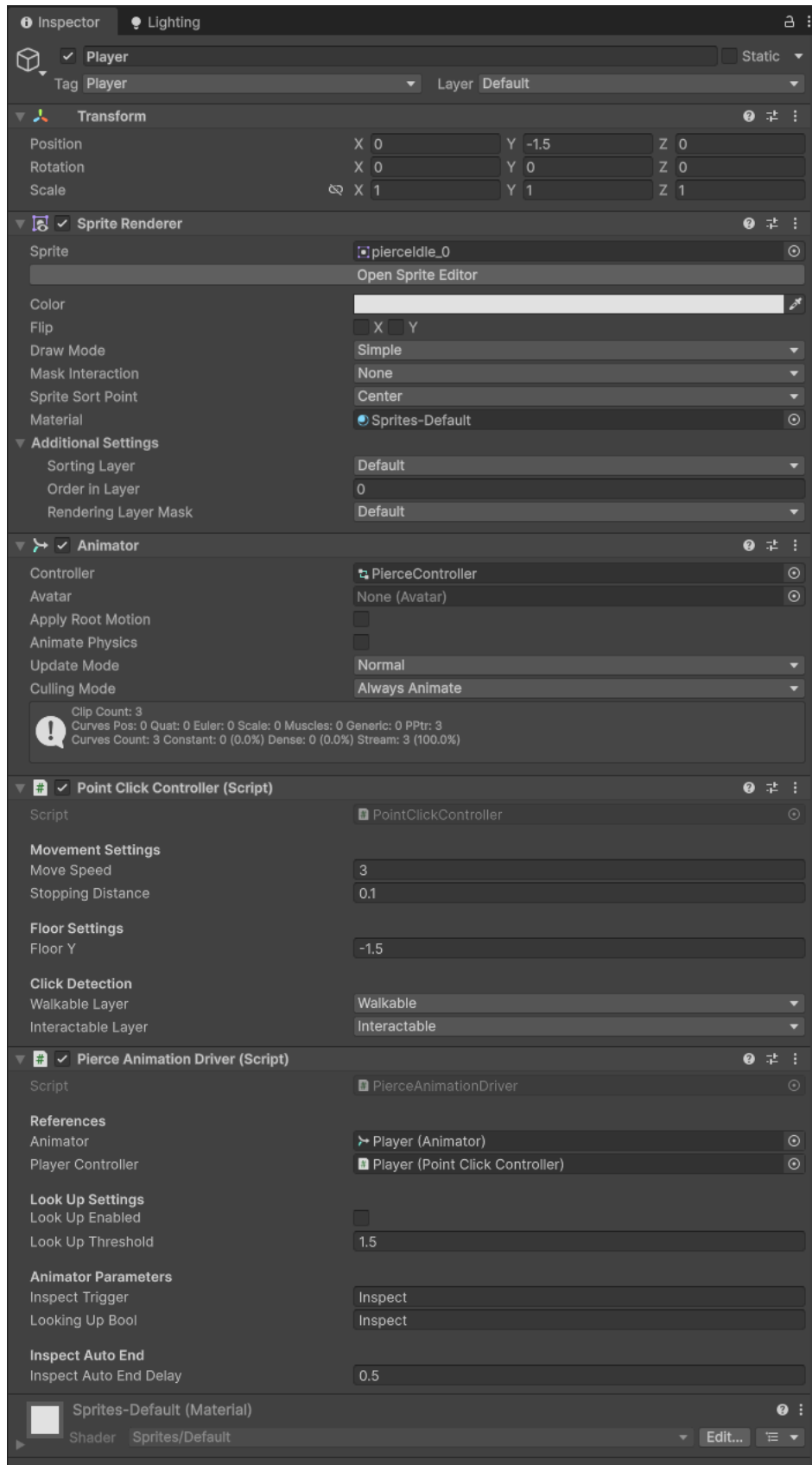
For dialogue snippets and audio, please see the audio folder in the project (**Assets > Audio**) to view Ashley's amazing work! You can also find it [here](#) and [here](#). Any dialogue references made in this doc are just hypotheticals for showing how the scripts work.

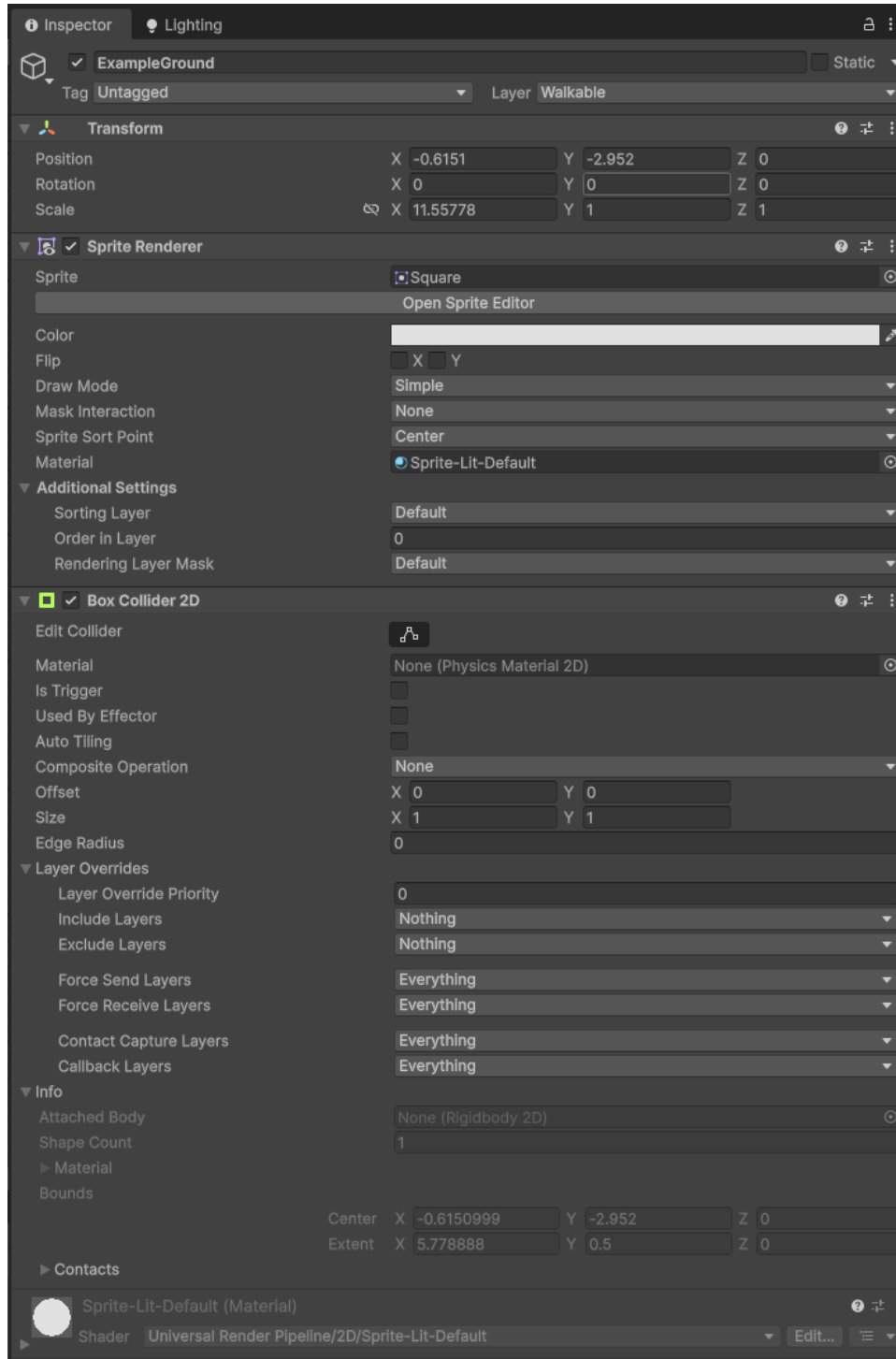
Animations

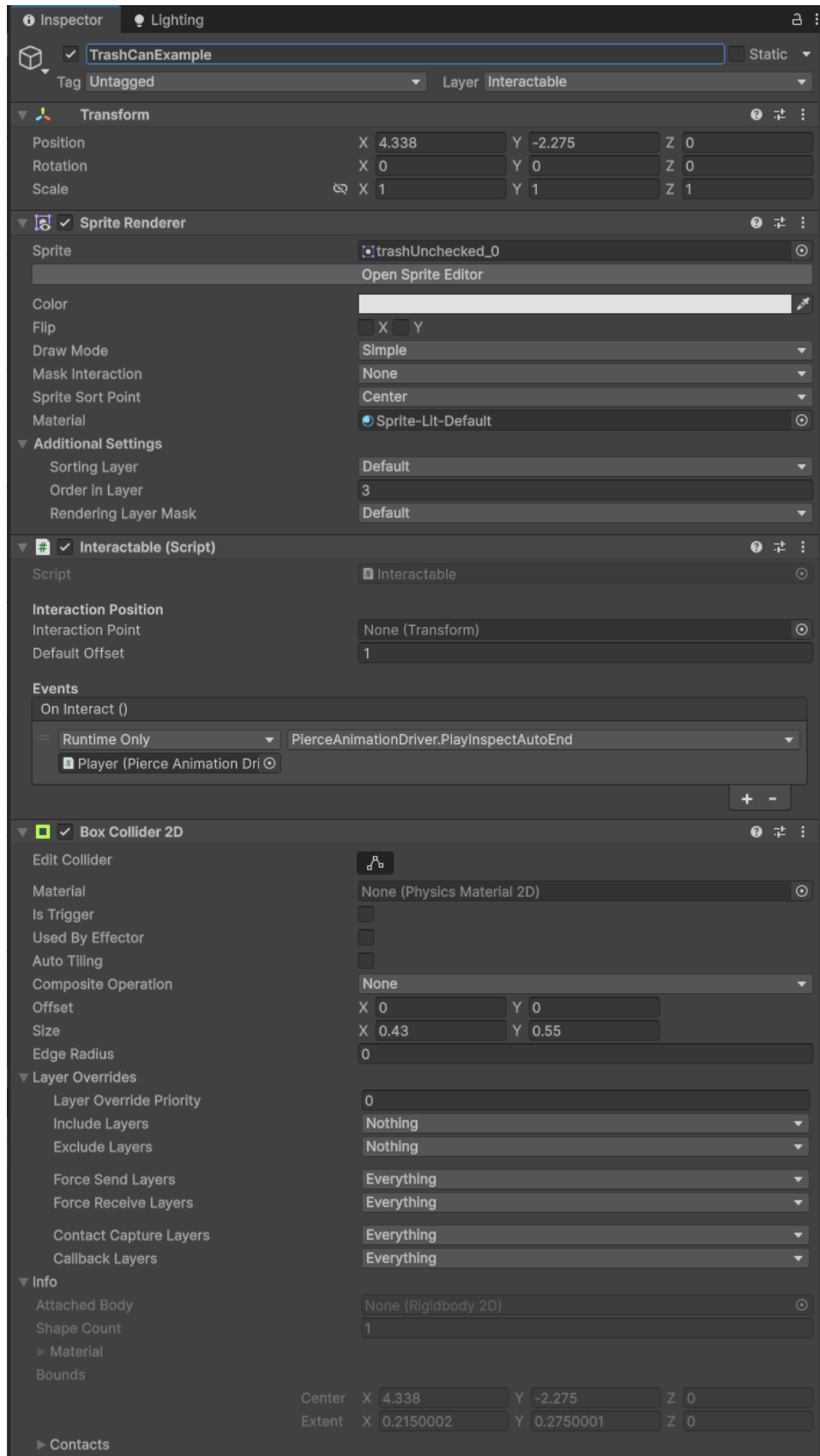
All animations should live inside: **Assets > Art > Animations**

These are set up and ready to go. They should look something like this in Unity - following each of these examples should get all of the animations to work (shoutout Bella for the amazing sprites):









2. Before You Begin: Creating Event Flags

IMPORTANT: Before setting up most scripts, you need to create Event Flags. These are like sticky notes that the game uses to remember things.

What is an Event Flag?

Think of an Event Flag like a light switch. It's either ON or OFF. The game uses these to remember things like:

- Did the player pick up the keys? (ON = yes, OFF = no)
- Is the break room door unlocked? (ON = yes, OFF = no)
- Did the player complete the tutorial? (ON = yes, OFF = no)

The 3 Event Flags You Need to Create

For the demo, you need exactly 3 Event Flags:

Flag Name	What It Remembers
TutorialComplete	Did Pierce break the office door window?
BreakRoomUnlocked	Did Pierce complete the vending machine puzzle?
HasKeys	Did Pierce find the janitor's keys?

How to Create an Event Flag (Step by Step)

Step 1: Open Your Project Window

Look at the bottom of Unity. You should see a panel called "Project". This shows all your files, like folders on your computer.

Step 2: Navigate to Where You Want to Save Flags

Click through the folders to get to: **Assets > ScriptableObjects > Flags**

(If this folder doesn't exist, create it by right-clicking and selecting Create > Folder)

Step 3: Right-Click in the Empty Space

Right-click anywhere in the empty area of the Project window (not on a file).

Step 4: Create the Event Flag

In the menu that appears, click: **Create > Game > Event Flag**

Step 5: Name Your Flag

A new file appears. Type one of these names exactly:

- TutorialComplete
- BreakRoomUnlocked
- HasKeys

Step 6: Configure the Flag

Click on the flag you just created. In the Inspector panel (usually on the right side), you'll see:

- **Is Active:** Leave this UNCHECKED (the flag starts OFF)
- **Reset On Play:** CHECK this box (so the flag resets when you test the game)
- **Default Value:** Leave this UNCHECKED (the flag should start as OFF)

Step 7: Repeat for All 3 Flags

Do steps 3-6 three times, once for each flag name.

3. Dialogue Folder Scripts

Location: **Assets > Scripts > Kevin Tests > Dialogue**

DialogueAsset.cs

What Does It Do?

This is like a script for a play. It holds all the words that characters say during a conversation. You create one of these for each conversation in your game.

How to Create a Dialogue Asset

Step 1: Right-click in your Project window

Step 2: Click: Create > Dialogue > Dialogue Asset

Step 3: Name it something like "Carmen_Dialogue" or "Janitor_Greeting"

Step 4: Click on it and fill in the fields in the Inspector:

- **Dialogue Name:** A friendly name like "Carmen First Meeting"
- **Default Player Portrait:** Drag in Pierce's face picture
- **Default Other Portrait:** Drag in the NPC's face picture
- **Player Name:** Type "Pierce"
- **Other Name:** Type the NPC's name like "Carmen"

Step 5: In the Lines section, click the + button to add dialogue lines

- **Speaker:** Choose who's talking (Player or Other)
- **Text:** Type what they say

DialogueRunner.cs

What Does It Do?

This is like a stage manager. When it's time for a conversation, this script tells the DialogueUI "Hey, start showing this conversation!"

How to Hook It Up

Step 1: Create an empty object: Right-click in Hierarchy > Create Empty

Step 2: Name it "DialogueSystem"

Step 3: Click Add Component, type "DialogueRunner", and click it

Step 4: The DialogueUI component will be added automatically (it's required)

Note: You only need ONE DialogueRunner in your entire scene!

DialogueTypes.cs

What Does It Do?

This is a behind-the-scenes helper. It defines what a "dialogue line" is and what a "speaker" is. You don't need to do anything with this file - it just needs to exist.

Hookup: None needed! Just make sure the file is in your project.

DialogueUI.cs

What Does It Do?

This is the TV screen for conversations. It shows the dialogue box, the character portraits, and types out the text letter by letter.

How to Hook It Up

Step 1: Create a Canvas: Right-click in Hierarchy > UI > Canvas

Step 2: Inside the Canvas, create a Panel: Right-click Canvas > UI > Panel

Step 3: Name it "DialoguePanel" and design your dialogue box

Step 4: Add these inside DialoguePanel:

- An Image for the player portrait (name it "PlayerImage")
- An Image for the NPC portrait (name it "NPCImage")
- A TextMeshPro text for dialogue (name it "DialogueText")
- A TextMeshPro text for speaker name (name it "SpeakerNameText")

Step 5: On your DialogueSystem object (with DialogueRunner), find DialogueUI and drag in:

- Dialogue Panel: your DialoguePanel
- Player Image: your PlayerImage
- Npc Image: your NPCImage
- Dialogue Text: your DialogueText
- Speaker Name Text: your SpeakerNameText

Step 6: Add a PlayerInput component to the DialogueSystem object

Step 7: Create an action called "AdvanceDialogue" mapped to left-click or a key

4. Flags Folder Scripts

Location: **Assets > Scripts > Kevin Tests > Flags**

EventFlag.cs

What Does It Do?

Remember how we talked about light switches earlier? This script IS that light switch. It's a tiny piece of data that says "yes" or "no" and other scripts can check it.

Example: The break room door asks "Is BreakRoomUnlocked ON?" If yes, let Pierce through. If no, say "The door is locked."

How to Hook It Up

You already learned how to create Event Flags in Section 2! Here's a quick reminder:

Step 1: Right-click in Project window

Step 2: Click: Create > Game > Event Flag

Step 3: Name it (like "HasKeys")

Step 4: Click on it and check "Reset On Play"

That's it! Now you can drag this flag into other scripts that need it.

5. HUD Folder Scripts

Location: **Assets > Scripts > Kevin Tests > HUD**

AdventureHUDController.cs

What Does It Do?

This is your game's control panel for on-screen stuff. It handles:

- A custom mouse cursor (instead of the boring arrow)
- A highlight that appears when you hover over clickable things
- An arrow that points to exits

How to Hook It Up

Step 1: Create a Canvas if you don't have one: Right-click Hierarchy > UI > Canvas

Step 2: Name it "HUDCanvas"

Step 3: Add the AdventureHUDController component to HUDCanvas

Step 4: Create a Cursor Image:

- Right-click HUDCanvas > UI > Image
- Name it "CursorImage"
- Drag your cursor picture into Source Image
- Set the Pivot to X: 0, Y: 1 (top-left corner)
- IMPORTANT: Uncheck "Raycast Target" so it doesn't block clicks

Step 5: Create a Highlight Image:

- Right-click HUDCanvas > UI > Image
- Name it "HighlightImage"
- Drag your highlight picture into Source Image
- Uncheck "Raycast Target"
- DISABLE the GameObject (uncheck the box by its name)

Step 6: Create an Arrow Image (same process as Highlight)

Step 7: On AdventureHUDController, drag in:

- Cursor Image: your CursorImage
- Highlight Image: your HighlightImage
- Arrow Image: your ArrowImage
- Interactable Layer: Check "Interactable" (see Layers setup below)
- Hide System Cursor: Check this box

ButtonMashPromptUI.cs

What Does It Do?

During the vending machine puzzle, this shows text like "Click to escape!" that bobs up and down. When you click, the text bounces bigger for a moment.

How to Hook It Up

Step 1: Inside HUDCanvas, create a Panel: Right-click > UI > Panel

Step 2: Name it "MashPromptPanel"

Step 3: Inside that panel, create text: Right-click > UI > Text - TextMeshPro

Step 4: Name it "MashPromptText" and type "Click to escape!"

Step 5: Add ButtonMashPromptUI to MashPromptPanel

Step 6: Drag MashPromptText into "Tmp Text" field

Step 7: Drag MashPromptPanel into "Panel Root" field

Step 8: Set Bob Speed to 2, Bob Amount to 10, Punch Scale to 1.2

Step 9: DISABLE MashPromptPanel (the puzzle will enable it when needed)

HoverHighlightTarget.cs

What Does It Do?

This is optional! Add it to any object that should glow or show a highlight when the mouse hovers over it.

How to Hook It Up

Step 1: Click on any object you want to highlight (like a door or item)

Step 2: Add Component > type "HoverHighlightTarget"

That's it! The defaults work fine.

SearchPromptUI.cs

What Does It Do?

When Pierce searches a trash can or desk, this shows "Searching..." on screen while he looks.

How to Hook It Up

Step 1: Inside HUDCanvas, create a Panel named "SearchPanel"

Step 2: Inside that panel, create TextMeshPro text named "SearchingText"

Step 3: Type "Searching..." in the text

Step 4: Add SearchPromptUI component to SearchPanel

Step 5: Drag SearchPanel into "Panel Root"

Step 6: Drag SearchingText into "Tmp Text"

Step 7: DISABLE SearchPanel

6. Interaction Folder Scripts

Location: **Assets > Scripts > Kevin Tests > Interaction**

DialogueInteraction.cs

What Does It Do?

Put this on NPCs (people Pierce can talk to). When Pierce clicks on them and walks over, a conversation starts. It can even show DIFFERENT conversations depending on game progress!

Example: The janitor says "Help, I'm stuck!" normally, but says "You found my keys! Thank you!" after Pierce gets the keys.

How to Hook It Up

Step 1: Click on your NPC sprite (like Carmen or the Janitor)

Step 2: Add Component > Box Collider 2D (so it can be clicked)

Step 3: Set the Layer to "Interactable" (dropdown near top of Inspector)

Step 4: Add Component > Interactable

Step 5: Add Component > DialogueInteraction

Step 6: Find your DialogueRunner in the scene and drag it into "Dialogue Runner"

Step 7: Drag your DialogueAsset into "Dialogue Asset"

For conditional dialogue (different conversation based on flag):

Step 8: Drag an EventFlag into "Conditional Flag" (like HasKeys)

Step 9: Drag a different DialogueAsset into "Conditional Dialogue Asset"

Step 10: On the Interactable component, find "On Interact", click +, drag this NPC in, and choose DialogueInteraction > StartDialogue

Interactable.cs

What Does It Do?

This is the magic "I can be clicked" key. Put it on ANYTHING you want Pierce to be able to click on - doors, items, NPCs, puzzles, everything.

How to Hook It Up

Step 1: Click on the object you want to be clickable

Step 2: Make sure it has a Collider2D (Box Collider 2D is most common)

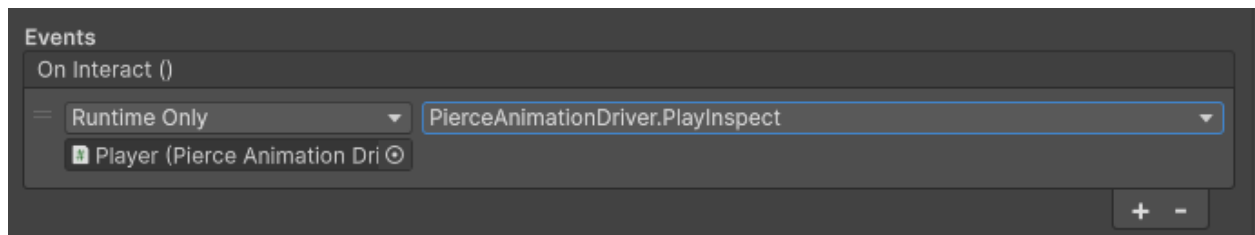
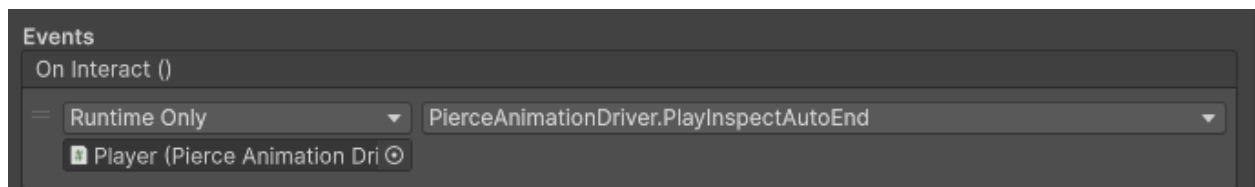
Step 3: Set Layer to "Interactable"

Step 4: Add Component > Interactable

Step 5: Set Default Offset to 1 (Pierce stands 1 unit away)

Step 6: In "On Interact", click + and connect what should happen

NOTE: For any interactable that you want Pierce to be able to move after using, use **PlayInspectAutoEnd**. For any that you would want movement to be disabled for (such as a puzzle), use **PlayInspect**. The puzzle scripts *should* enable movement again after the puzzle is completed.



LockedDoor.cs

What Does It Do?

A door that can be LOCKED. It checks an EventFlag to see if Pierce is allowed through. Perfect for the break room door that only opens after the vending machine puzzle.

How to Hook It Up

Step 1: Click on your door sprite

Step 2: Add Box Collider 2D

Step 3: Set Layer to "Interactable"

Step 4: Add Component > Interactable

Step 5: Add Component > LockedDoor

Step 6: Fill in LockedDoor fields:

- Target Room: Drag the Room this door leads to
- Target Entry Point Name: Type "Left" or "Right" etc.
- Room Manager: Drag in your RoomManager
- Unlock Flag: Drag in the EventFlag that unlocks this door (like BreakRoomUnlocked)
- Require Flag True: CHECK this box

Step 7: On Interactable's "On Interact", click +, drag this door in, choose LockedDoor > UseDoor

7. Player Folder Scripts

Location: **Assets > Scripts > Kevin Tests > Player**

AdventureCameraController.cs

What Does It Do?

This controls your camera! It can:

- Stay perfectly still (for small rooms)
- Follow Pierce around (for big rooms)
- Add black bars for a cinematic film look

How to Hook It Up

Step 1: Click on Main Camera in your Hierarchy

Step 2: Add Component > AdventureCameraController

Step 3: For the classic film look, leave Target Aspect Width at 1.37

Step 4: For small rooms: Check "Use Room Mode" and set Room Position

Step 5: For big rooms: Uncheck "Use Room Mode", drag Pierce into Target

FootstepLoop.cs

What Does It Do?

Plays footstep sounds when Pierce walks. Automatically starts when he moves and stops when he stops.

How to Hook It Up

Step 1: Click on Pierce (your player)

Step 2: Add Component > Audio Source

Step 3: Add Component > FootstepLoop

Step 4: Drag the Audio Source into "Audio Source" field

Step 5: Drag your footstep sound file into "Footstep Clip"

Step 6: Set Volume to 0.5 (adjust to taste)

PierceAnimationDriver.cs

What Does It Do?

Controls Pierce's special animations:

- Makes him look up when your mouse is above him (we never got to working on the art asset for this)
- Plays an "inspect" animation when examining things

How to Hook It Up

Step 1: Click on Pierce

Step 2: Add Component > PierceAnimationDriver

Step 3: Drag Pierce's Animator into "Animator" field

Step 4: In your Animator Controller, add these parameters:

- "Inspect" (type: Trigger)
- "IsLookingUp" (type: Bool) - not needed, we never got to working on the art asset for this

PointClickController.cs

What Does It Do?

THIS IS THE MOST IMPORTANT SCRIPT! It makes Pierce walk when you click. Click on the floor, Pierce walks there. Click on an object, Pierce walks to it and interacts with it.

Setting Up Layers First!

Before hooking up this script, you need to create two Layers:

Step 1: Go to Edit menu > Project Settings

Step 2: Click on "Tags and Layers"

Step 3: Scroll to "Layers" section

Step 4: Find an empty slot (like User Layer 6) and type "Walkable"

Step 5: Find another empty slot and type "Interactable"

Setting Up a Walkable Area

Pierce can only walk on "Walkable" areas:

Step 1: Right-click Hierarchy > 2D Object > Sprites > Square

Step 2: Name it "WalkableArea"

Step 3: Scale it to cover your floor (like Scale X: 20)

Step 4: Position it at your floor level

Step 5: Set its Layer to "Walkable"

Step 6: Add Box Collider 2D

Step 7: Make it invisible: Set SpriteRenderer's Color alpha to 0

How to Hook It Up

Step 1: Click on Pierce

Step 2: Make sure he has a SpriteRenderer and Collider2D

Step 3: Set his Tag to "Player" (dropdown near top)

Step 4: Add Component > PointClickController

Step 5: Set Move Speed to 3

Step 6: Set Stopping Distance to 0.1

Step 7: Set Floor Y to your floor's Y position (look at Pierce's Y when standing)

Step 8: For Walkable Layer, click dropdown and check "Walkable"

Step 9: For Interactable Layer, click dropdown and check "Interactable"

8. Puzzles Folder Scripts

Location: **Assets > Scripts > Kevin Tests > Puzzles**

Key Hunt Manager.cs

What Does It Do?

This is optional! It can randomly pick which searchable object has the keys, so players can't just memorize where they are.

How to Hook It Up

Step 1: Create Empty object named "Key Hunt Manager"

Step 2: Add Component > Key Hunt Manager

Step 3: Drag ALL your searchable props into "Searchable Props" list

Step 4: Drag Has Keys Event Flag into "Has Keys Flag"

Step 5: Check "Randomize Key Location" for random placement

Searchable Prop.cs

What Does It Do?

Put this on things Pierce can search, like trash cans, desks, or filing cabinets. When clicked, Pierce "searches" it and might find the keys!

How to Hook It Up

Step 1: Click on your searchable object (like a trash can)

Step 2: Add Box Collider 2D

Step 3: Set Layer to "Interactable"

Step 4: Add Component > Interactable

Step 5: Add Component > Searchable Prop

Step 6: Set Search Duration to 1.5 (seconds)

Step 7: For the ONE object that has keys, check "Contains Keys"

Step 8: Drag Has Keys Event Flag into "Has Keys Flag"

Step 9: On Interactable's "On Interact", click +, drag this object in, choose SearchableProp > Search

TutorialPuzzle.cs

What Does It Do?

This runs the first puzzle: Pierce is trapped in his office. He needs to find a bat and use it to break the door's window.

How to Hook It Up

Step 1: Create Empty object named "TutorialController"

Step 2: Add Component > TutorialPuzzle

Step 3: Set up the BAT:

- Create bat sprite
- Add Box Collider 2D, set Layer to Interactable
- Add Interactable component
- Drag bat into TutorialPuzzle's "Bat" field
- On bat's Interactable "On Interact", connect to TutorialPuzzle > InteractWithBat

Step 4: Set up the DOOR:

- Create door sprite
- Add Box Collider 2D, set Layer to Interactable
- Add Interactable component
- Drag door into "Door" field
- Drag broken door sprite into "Broken Door Sprite"
- On door's Interactable "On Interact", connect to TutorialPuzzle > InteractWithDoor

Step 5: Create 4 DialogueAssets and drag them in:

- Door Locked Dialogue: "It's locked."
- Need Bat Dialogue: "I need something to break this."
- Bat Picked Up Dialogue: "This could work."
- Door Opened Dialogue: "That did it!"

Step 6: Drag DialogueRunner into "Dialogue Runner"

Step 7: Drag TutorialComplete EventFlag into "Tutorial Complete Flag"

VendingMachinePuzzle.cs

What Does It Do?

Pierce's hand gets stuck in a vending machine! A fullscreen image of his arm appears and shakes. Click fast to escape! The faster you click, the more it shakes.

How to Hook It Up

Step 1: Create your vending machine sprite

Step 2: Add Box Collider 2D, set Layer to Interactable

Step 3: Add Interactable component

Step 4: Add VendingMachinePuzzle component

Step 5: Create the fullscreen arm:

- In HUDCanvas, create UI > Image
- Name it "FullscreenArmImage"
- Drag your arm picture into Source Image
- Hold Alt and click bottom-right anchor preset (stretches to fill screen)
- Set Left, Right, Top, Bottom all to 0
- DISABLE the object

Step 6: Create the hand reveal image (same process)

Step 7: Fill in VendingMachinePuzzle fields:

- Fullscreen Arm Image: your arm image
- Hand Reveal Image: your hand image
- Base Shake Intensity: 5
- Max Shake Intensity: 30
- Break Room Unlocked Flag: drag in BreakRoomUnlocked EventFlag

Step 8: On Interactable's "On Interact", connect to VendingMachinePuzzle > StartPuzzle

9. Rooms Folder Scripts

Location: **Assets > Scripts > Kevin Tests > Rooms**

Door.cs

What Does It Do?

A simple automatic door. When Pierce WALKS INTO it (touches it), he teleports to another room. Use this for unlocked doors that don't need clicking.

How to Hook It Up

Step 1: Create an invisible trigger area (or use a door sprite)

Step 2: Add Box Collider 2D

Step 3: CHECK "Is Trigger" on the collider

Step 4: Add Door component

Step 5: Fill in:

- Target Room: the Room Pierce goes to
- Target Entry Point Name: "Left", "Right", "Top", or "Bottom"
- Room Manager: your RoomManager object

InteractDoor.cs

What Does It Do?

This is your existing script for doors that work with the new Input System. Keep using it if you prefer input-based door interaction.

Room.cs

What Does It Do?

This defines what a "room" is. It knows the room's boundaries and where Pierce should appear when entering from different doors.

How to Hook It Up

Step 1: Create an empty object named "Room_OfficeName"

Step 2: Put ALL objects for this room as children (background, props, NPCs, doors)

Step 3: Add Room component

Step 4: Set Room ID to something like "Office" or "Hallway"

Step 5: Create room bounds:

- Create 2D Square sprite as child
- Scale it to cover the ENTIRE room
- Add Box Collider 2D, check "Is Trigger"
- Make it invisible (alpha = 0)
- Drag it into "Room Bounds" field

Step 6: Create entry points (empty objects at positions where Pierce enters):

- Create Empty child, name it "EntryPoint_Left", position it
- Drag into "Entry Point Left" field
- Repeat for Right, Top, Bottom as needed

RoomManager.cs

What Does It Do?

The traffic controller for rooms! When Pierce goes through a door, RoomManager turns OFF the old room and turns ON the new room.

How to Hook It Up

Step 1: Create Empty object named "RoomManager"

Step 2: Add RoomManager component

Step 3: Drag your starting Room into "Current Room"

Step 4: Drag Main Camera (with AdventureCameraController) into "Camera Controller"

Note: Only ONE RoomManager per scene!

10. Demo-Specific Folder Scripts

Location: **Assets > Scripts > Kevin Tests > Demo-Specific**

DemoEndSequence.cs

What Does It Do?

When the demo ends, this plays a nice ending:

1. Screen fades to black
2. "To be continued..." appears
3. Game returns to main menu

How to Hook It Up

Step 1: In HUDCanvas, create UI > Image named "FadeImage"

- Set color to BLACK
- Set alpha to 0 (transparent)
- Stretch to fill screen (Alt + bottom-right anchor)
- DISABLE it

Step 2: Create UI > Text - TextMeshPro named "ToBeContinuedText"

- Type "To be continued..."
- Set color to white, alpha to 0
- Center it on screen
- DISABLE it

Step 3: Create Empty object named "DemoEndManager"

Step 4: Add DemoEndSequence component

Step 5: Fill in:

- Fade Image: your FadeImage
- Tmp Text: your ToBeContinuedText
- Fade Duration: 1.5

- Text Display Duration: 3
- Main Menu Scene Name: exact name of your menu scene

Step 6: On the janitor's DialogueInteraction "On Dialogue Ended", connect to DemoEndSequence > PlayEndSequence

FadeGroup.cs

What Does It Do?

Makes sprites fade in or out smoothly. Useful for making things appear or disappear gracefully.

How to Hook It Up

Step 1: Add FadeGroup to any object with sprites you want to fade

Step 2: Drag SpriteRenderers into "Renderers" list (or leave empty to auto-find children)

Step 3: Set Fade Duration to 0.5

Call FadeOut() to fade away, FadeIn() to fade in

10. Audio Setup

Overview

Our game needs two types of audio:

Type	Description	Where It Plays
Music	Background music that sets the mood	Main Menu and In-Game
Sound Effects (SFX)	Sounds that respond to game events	Various moments during gameplay

Available Audio Files

Music Options

There are **six music tracks** available in the project. You need to choose **two** of them:

- **One for the Main Menu** - This plays on the title screen
- **One for In-Game** - This plays while Pierce explores and tries to escape the office (our gameplay scene)

Listen to all five tracks and pick which two fit best for each purpose. The Main Menu music should feel inviting and set the noir mood. The In-Game music should be atmospheric without being distracting during puzzles and exploration.

Sound Effects Available

SFX Name	File Name	What It's For
Footsteps	Dress Shoes Walk	Pierce walking around
Window Breaking	Breaking Window	Tutorial puzzle - when Pierce smashes the door window
Machine Shaking	Shaking Machine	Vending machine puzzle - during the button mash

Setting Up Main Menu Music

What Does It Do?

Music that plays automatically when the player reaches the main menu screen.

Step 1: Create a Music Object

1. Open your **Main Menu scene**
2. Right-click in Hierarchy
3. Click "Create Empty"
4. Name it "MainMenuMusic"

Step 2: Add Audio Source Component

1. With MainMenuMusic selected, click "Add Component"
2. Type "Audio Source" and click it

Step 3: Configure the Audio Source

Fill in these settings:

- **AudioClip:** Drag your chosen main menu music track here
- **Play On Awake:** CHECK this box (music starts when scene loads)
- **Loop:** CHECK this box (music repeats forever)
- **Volume:** 0.5 (adjust to your preference)
- **Spatial Blend:** Set to 0 (2D sound)

Done! The music will now play automatically when the Main Menu scene loads.

Setting Up In-Game Music

What Does It Do?

Background music that plays while the player is actually playing the game.

Step 1: Create a Music Object

1. Open your **Game scene** (where Pierce walks around)
2. Right-click in Hierarchy
3. Click "Create Empty"
4. Name it "InGameMusic"

Step 2: Add Audio Source Component

1. With InGameMusic selected, click "Add Component"
2. Type "Audio Source" and click it

Step 3: Configure the Audio Source

Fill in these settings:

- **AudioClip:** Drag your chosen in-game music track here
- **Play On Awake:** CHECK this box
- **Loop:** CHECK this box
- **Volume:** 0.3 (lower than menu music so it doesn't overpower sound effects)
- **Spatial Blend:** Set to 0 (2D sound)

Done! The music will now play during gameplay.

Setting Up Footsteps (Dress Shoes Walk)

What Does It Do?

Plays the sound of Pierce's dress shoes when he walks. Automatically starts when he moves and stops when he stops.

Step 1: Add Audio Source to Pierce

1. Click on Pierce (your player character) in the Hierarchy

2. Click "Add Component"
3. Type "Audio Source" and click it
4. Configure the Audio Source:
 - **AudioClip:** Leave this empty (FootstepLoop will handle it)
 - **Play On Awake:** UNCHECK this box
 - **Loop:** CHECK this box
 - **Volume:** Set to 1

Step 2: Add FootstepLoop Component

1. With Pierce still selected, click "Add Component"
2. Type "FootstepLoop" and click it

Step 3: Configure FootstepLoop

Fill in these fields:

- **Audio Source:** Drag Pierce's AudioSource component here
- **Footstep Clip:** Drag the "Dress Shoes Walk" audio file here
- **Volume:** 0.5 (adjust to taste)
- **Stop Delay:** 0.1
- **Footsteps Enabled:** CHECK this box

Step 4: Test It

1. Press Play
2. Click somewhere to make Pierce walk
3. You should hear dress shoe footsteps while he moves
4. Footsteps should stop when he stops

Setting Up Window Breaking Sound

What Does It Do?

Plays when Pierce successfully breaks the office door window with the bat in the tutorial puzzle.

Step 1: Create an Audio Source for Sound Effects

1. In your Tutorial Room, create an empty object named "TutorialAudio"
2. Add an Audio Source component
3. Configure it:
 - **Play On Awake:** UNCHECK this box
 - **Loop:** UNCHECK this box

Step 2: Connect to TutorialPuzzle

1. Find your TutorialPuzzle component (on TutorialController or similar)
2. Find the **On Puzzle Completed** event
3. Click the + button
4. Drag your TutorialAudio object into the object slot
5. Click the dropdown and choose: **AudioSource > PlayOneShot**
6. Drag the "Breaking Window" audio file into the AudioClip parameter

Now when Pierce breaks the window, the sound plays!

Setting Up Shaking Machine Sound

What Does It Do?

Plays a looping shake/struggle sound during the vending machine button mash puzzle.

Step 1: Add Audio Source to Vending Machine

1. Click on your Vending Machine object
2. Click "Add Component"
3. Type "Audio Source" and click it
4. Configure it:
 - **AudioClip:** Drag the "Shaking Machine" audio file here
 - **Play On Awake:** UNCHECK this box
 - **Loop:** CHECK this box (it loops during the puzzle)
 - **Volume:** 0.6

Step 2: Start Sound When Puzzle Begins

1. Find your VendingMachinePuzzle component
2. Find the **On Puzzle Started** event
3. Click the + button
4. Drag the Vending Machine object into the slot
5. Click the dropdown and choose: **AudioSource > Play**

Step 3: Stop Sound When Puzzle Ends

1. Find the **On Puzzle Completed** event
2. Click the + button
3. Drag the Vending Machine object into the slot
4. Click the dropdown and choose: **AudioSource > Stop**

Now the shaking sound loops during the puzzle and stops when Pierce escapes!

Audio Troubleshooting

No music playing:

- Is Play On Awake checked?
- Is the AudioClip assigned?
- Is Volume above 0?

Music restarts when changing rooms:

- This is normal - the music object is part of the room
- If you want continuous music, keep the music object outside of Room objects

Footsteps not playing:

- Is FootstepLoop component on Pierce?
- Is the AudioSource component on Pierce?
- Is "Dress Shoes Walk" clip assigned to Footstep Clip?
- Is Footsteps Enabled checked?

Footsteps too loud/quiet:

- Adjust the Volume field in FootstepLoop component

Sound effects not playing on events:

- Is the AudioSource assigned in the event?
- Is the correct function selected (PlayOneShot for one-time sounds, Play for loops)?
- Is the AudioClip assigned?

12. Complete Demo Walkthrough

Here's how the whole demo flows from start to finish:

Part 1: Tutorial Room (Pierce's Office)

Pierce starts trapped in his office. The door is locked!

1. Player clicks the door - Pierce says "It's locked"
2. Player clicks the door again - "I need something to break this"
3. Player clicks the bat - Pierce picks it up (bat disappears)
4. Player clicks the door - Pierce breaks the window!
5. **TutorialComplete** flag turns ON

Part 2: Hallway

Pierce explores the hallway and can:

- Talk to Carmen (optional dialogue)
- Talk to the Janitor (trapped in elevator)
- Click the vending machine - triggers the button mash puzzle
- After completing the puzzle, **BreakRoomUnlocked** flag turns ON
- Now the break room door works!

Part 3: Break Room

Pierce searches for the janitor's keys:

- Multiple searchable objects (trash can, desk, etc.)
- Most say "Nothing here..."
- ONE has the keys - **HasKeys** flag turns ON

Part 4: Ending

Pierce returns to help the janitor:

- Because HasKeys is ON, the janitor has NEW dialogue
- "You found my keys! Thank you!"
- When dialogue ends, DemoEndSequence plays
- Fade to black... "To be continued..." ... Main menu

13. Troubleshooting

Pierce Won't Move

- Is PointClickController on Pierce? Add it.
- Did you create the "Walkable" layer? Check Tags and Layers.
- Is your WalkableArea on the "Walkable" layer?
- Does WalkableArea have a Collider2D?

Can't Click on Objects

- Does the object have a Collider2D?
- Is the object's Layer set to "Interactable"?
- Is "Interactable" checked in PointClickController's Interactable Layer?

Nothing Happens When Clicking

- Does the object have an Interactable component?
- Is "On Interact" connected to something?
- Is the function name spelled correctly?

Dialogue Doesn't Start

- Is there a DialogueRunner in the scene?
- Is the DialogueRunner dragged into the DialogueInteraction?
- Is a DialogueAsset assigned?

Door Doesn't Work

- Does the target Room have a Room component?
- Is RoomManager assigned on the door?
- Does the entry point name match exactly? ("Left" not "left")

Locked Door Always Locked

- Is the EventFlag actually turning ON?
- Is "Require Flag True" checked?
- Is "Reset On Play" checked on the flag?

Vending Machine Arm Not Shaking

- Is Base Shake Intensity greater than 0?
- Is the arm Image assigned?
- Did the puzzle actually start? (Did you click the vending machine?)

Event Flag Not Working

- Does the EventFlag file exist?
- Is it dragged into the correct slot?
- Is "Reset On Play" checked?
- Are you testing in Play mode? (Flags don't change in Edit mode)

End of Documentation