

Workshop 2: Critical Thinking in Game Development

Introduction to Unity Editor & C# Scripting

Lucas P. Cordova, Ph.D.

Table of contents

1	Game Development Skills	2
2	Unity Roadmap	3
3	Unity Editor Interface	4
4	Layout Best Practices	4
5	Benefits of Tall Layout	5
6	Today's Demo: Bearcat Wheel of Fate	6
7	Project Setup	6
8	Scene Setup	7
9	Introduction to C# Scripting	8
10	The WheelSpin Script	9
11	Understanding the Script - Part 1	10
12	Understanding the Script - Part 2	10
13	Understanding the Script - Part 3	10
14	Time.deltaTime Explained	11
15	Adding Debug Output	11

16 Testing Your Game	11
17 Wheel Segments Decoded	12
18 Wheel Segments (continued)	12
19 Input System Resources	13
20 Key Takeaways	13
21 Next Lecture Preview	13
22 Questions & Practice Time	14

1 Game Development Skills

1.1 Core Competencies

- **Learning** - Continuous adaptation
- **Critical Thinking** - Design decisions
- **Problem Solving** - Debug & iterate
- **Mathematics** - Probability & functions
- **Programming** - Unity/C# mastery
- **Creativity** - Unique experiences

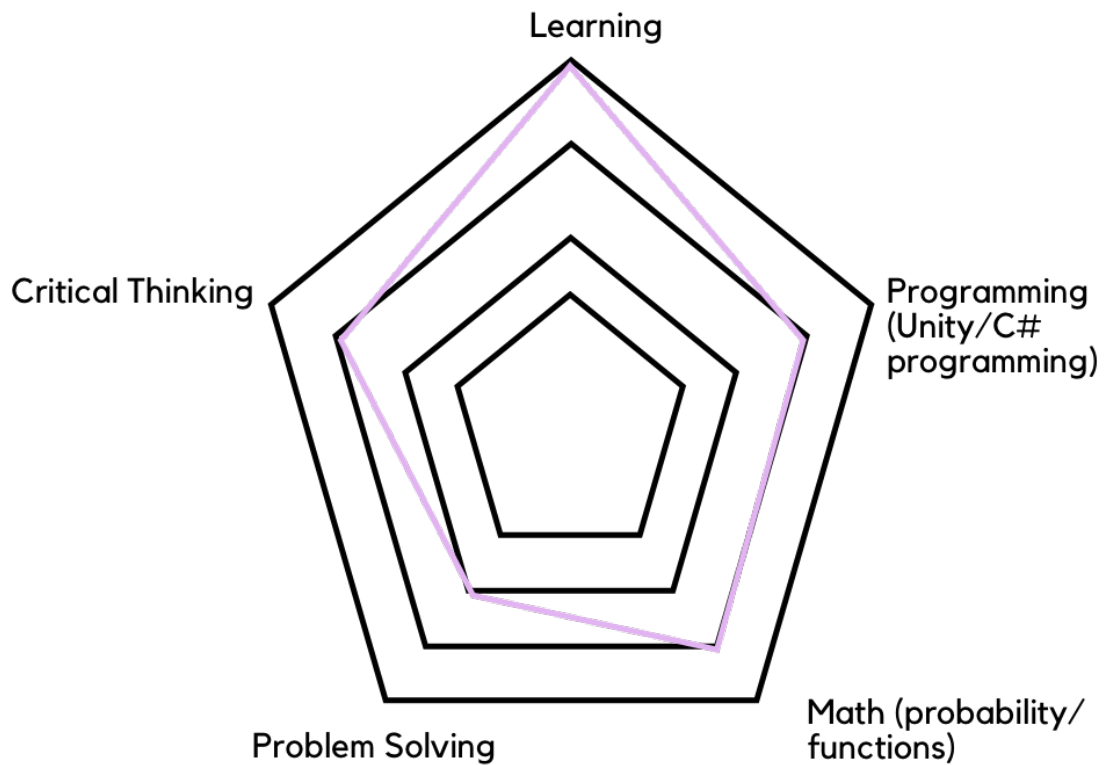


Figure 1: Game Development Competencies

2 Unity Roadmap

2.1 Where We're Going in Unity

- **Weeks 2-5:** 2D Game Development
 - Sprites, Physics2D, Tilemaps
 - 2D Game Mini-Project
- **Weeks 6-10:** 3D Game Development
 - 3D Models, Lighting, Physics
 - Advanced Interactions

- 3D Game Mini-Project
 - **Final Project:** Your Choice!
 - Any Unity game or simulation
 - 2D, 3D, or hybrid approach
-

3 Unity Editor Interface

3.1 Version Compatibility

- Unity Editor interface varies by version
- **Best Practice:** Match tutorial version exactly
- If versions differ:
 - Expect UI differences
 - Research feature locations
 - Check Unity documentation

You should have completed the Editor Interface tutorial

4 Layout Best Practices

4.1 Recommended “Tall” Layout Setup

1. **Window** → **Layouts** → **Tall**
2. Move **Game View** below Scene View
3. Enable **Console** (Window → General → Console)
4. Arrange for maximum workspace

4.2 Recommended “Tall” Layout

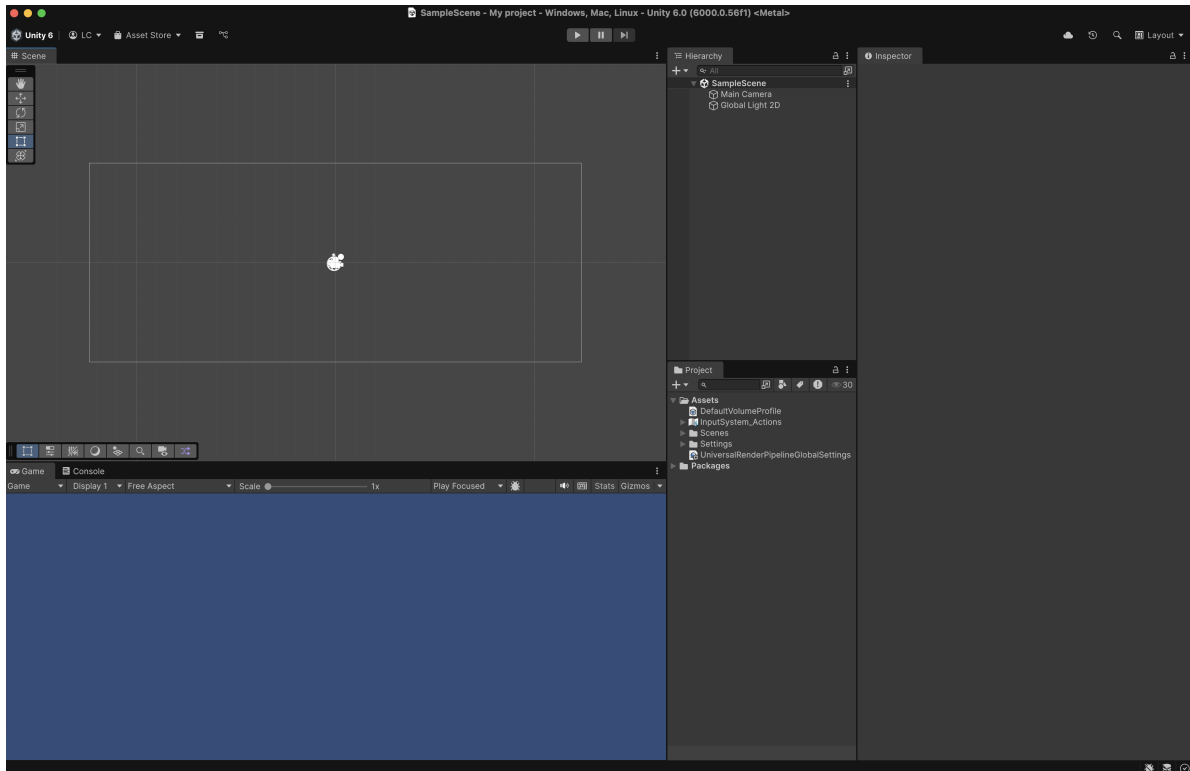


Figure 2: Unity Tall Layout

5 Benefits of Tall Layout

5.1 Scene View

- Maximum vertical space
- Better for level design
- Easier object placement

5.2 Game View + Console

- Test gameplay immediately

- See debug output live
 - Catch errors quickly
-

6 Today's Demo: Bearcat Wheel of Fate

6.1 What We'll Build

- Interactive spinning wheel game
- Introduction to C# scripting
- Basic Unity 2D setup
- Input handling with new Input System

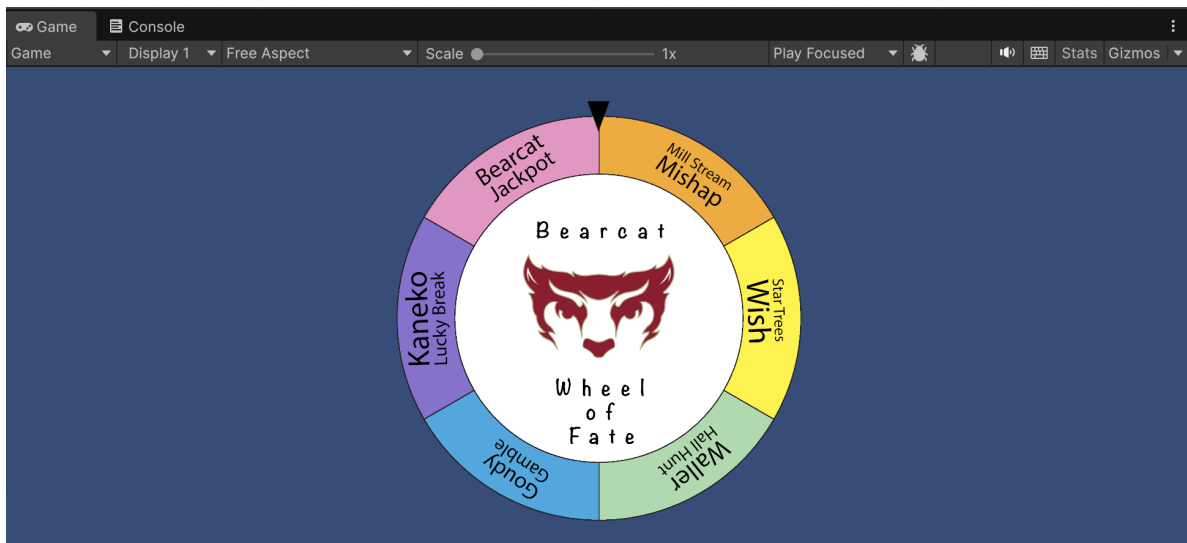


Figure 3: Bearcat Wheel of Fate

7 Project Setup

7.1 Creating a New 2D Project

1. **Unity Hub** → New Project

2. Select **2D (URP)** template
3. Name: “BearkatWheelOfFate”
4. Choose location → Create

7.2 Import Assets

- Wheel sprite (provided)
 - Marker/pointer sprite (provided)
 - Drag into Assets folder
-

8 Scene Setup

8.1 Positioning GameObjects

1. **Wheel GameObject**
 - Position: (0, 0, 0)
 - Scale as needed
2. **Marker GameObject**
 - Position above wheel
 - Ensure proper sorting order

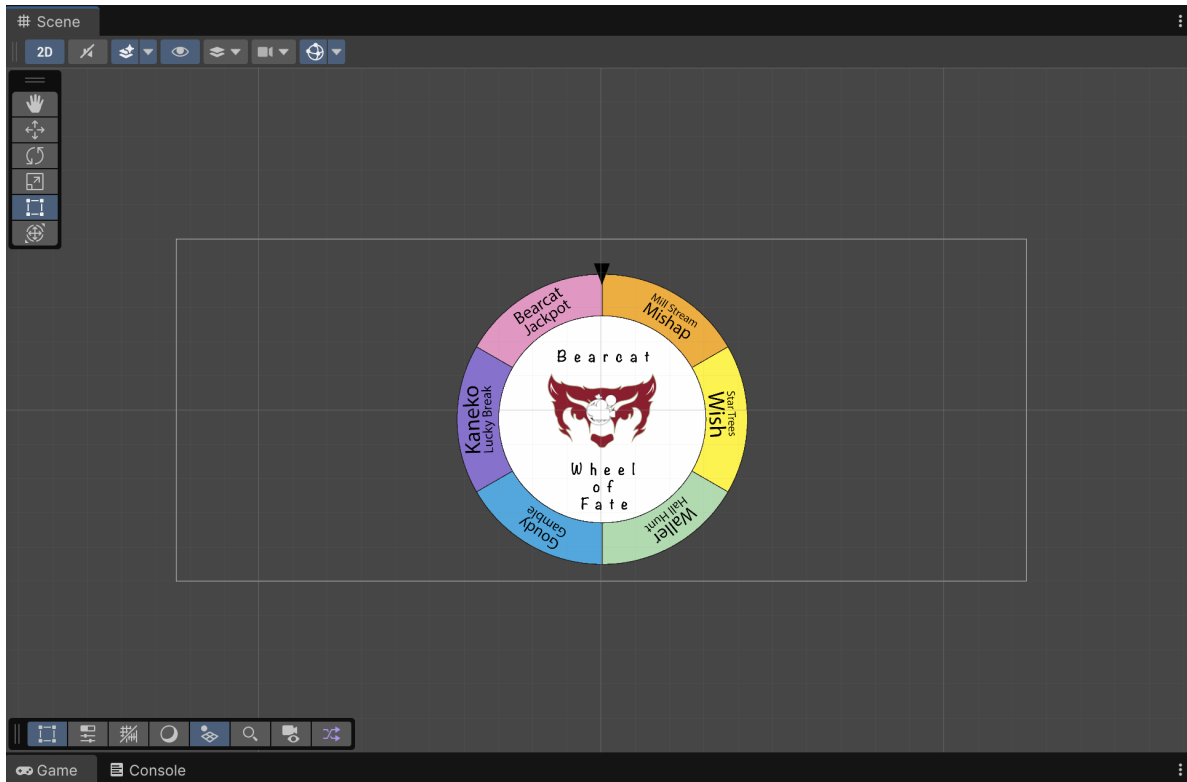


Figure 4: Scene Setup

9 Introduction to C# Scripting

9.1 Creating Our First Script

1. **Assets** → Create → C# Script
 2. Name: **WheelSpin**
 3. Attach to Wheel GameObject
 4. Open in code editor
-

10 The WheelSpin Script

```
1  using System;
2  using UnityEngine;
3  using UnityEngine.InputSystem;
4
5  public class WheelSpin : MonoBehaviour
6  {
7      private float _rotationSpeed = 0.0f;
8
9      // Start is called once before the first execution of Update
10     void Start()
11     {
12         _rotationSpeed = 0.0f;
13     }
14
15     // Update is called once per frame
16     void Update()
17     {
18         // Check if the space bar was pressed this frame
19         if (Keyboard.current.spaceKey.wasPressedThisFrame)
20         {
21             _rotationSpeed = new System.Random().Next(2000, 5000);
22         }
23
24         if (_rotationSpeed > 0)
25         {
26             float rotationAmount = _rotationSpeed * Time.deltaTime;
27             transform.Rotate(0, 0, rotationAmount);
28
29             // Decrease rotation speed over time
30             _rotationSpeed -= 500 * Time.deltaTime;
31         }
32     }
33 }
```

11 Understanding the Script - Part 1

11.1 Key Components

- **using statements** - Import Unity functionality
 - **MonoBehaviour** - Base class for Unity scripts
 - ****private float _rotationSpeed****
 - Private: Only accessible within this class
 - float: Decimal number type
 - Underscore prefix: C# convention for private fields
-

12 Understanding the Script - Part 2

12.1 Unity Lifecycle Methods

void Start() - Called ONCE when GameObject becomes active - Initialization code goes here - Runs before first Update()

void Update() - Called EVERY frame (30-120+ times/second!) - Game logic and input checking - Performance critical code

13 Understanding the Script - Part 3

13.1 Transform & Rotation

```
1 transform.Rotate(0, 0, rotationAmount);
```

- **transform** - Built-in reference to GameObject's Transform
 - Controls position, rotation, scale
 - Rotate(x, y, z) - Rotates around each axis
 - Z-axis rotation for 2D spinning
-

14 Time.deltaTime Explained

14.1 Frame-Independent Movement

```
1 float rotationAmount = _rotationSpeed * Time.deltaTime;
```

- **Time.deltaTime** = Time since last frame
 - Makes movement consistent across different framerates
 - 60 FPS: deltaTime 0.0167 seconds
 - 30 FPS: deltaTime 0.0333 seconds
-

15 Adding Debug Output

15.1 Debugging with Console

```
1 void Update()
2 {
3     if (Keyboard.current.spaceKey.wasPressedThisFrame)
4     {
5         _rotationSpeed = new System.Random().Next(2000, 5000);
6         Debug.Log($"Spinning! Initial speed: {_rotationSpeed}");
7     }
8
9     if (_rotationSpeed > 0)
10    {
11        // ... rotation code ...
12        Debug.Log($"Current speed: {_rotationSpeed:F2}");
13    }
14 }
```

16 Testing Your Game

16.1 Play Mode Testing

1. Save your script (Ctrl/Cmd + S)

2. Return to Unity Editor
 3. Press **Play** button
 4. Press **Spacebar** to spin
 5. Watch **Console** for debug messages
-

17 Wheel Segments Decoded

17.1 Your Fate Awaits...

1. **Bearcat Jackpot**
You landed big! (good fortune)
 2. **Mill Stream Mishap**
Oops, you slipped by the ducks (bad luck)
 3. **Star Trees Wish**
Make a lucky wish under the trees
-

18 Wheel Segments (continued)

4. **Waller Hall Haunt**
Unlucky... you met a campus ghost
 5. **Kaneko Lucky Break**
You found free snacks in the lounge
 6. **Goudy Gamble**
Mystery food choice, will it be lucky or not?
-

19 Input System Resources

19.1 Exploring Keyboard.current

```
1 // Other useful inputs:
2 Keyboard.current.enterKey.wasPressedThisFrame
3 Keyboard.current.escapeKey.isPressed
4 Keyboard.current.wKey.wasPressedThisFrame
```

19.2 Documentation

- [Unity Input System - Keyboard](#)
 - [Input System Quick Start Guide](#)
-

20 Key Takeaways

- Unity Editor layout affects productivity
 - C# scripts bring GameObjects to life
 - Start() initializes, Update() runs game logic
 - Time.deltaTime ensures smooth movement
 - Debug.Log() is your debugging friend
 - Input System provides modern input handling
-

21 Next Lecture Preview

21.1 Gearing Up Towards the 2D Mini-Game Project

- Sprite animations
 - Tilemap environments
 - Player movement & collision
 - Enemy AI basics
 - Collectibles & UI
-

22 Questions & Practice Time

22.1 Your Turn!

1. Modify the spin speed range
2. Add different input keys
3. Experiment with rotation direction
4. Add sound effects (stretch goal)

22.2 Remember

- Save scripts before testing
- Check Console for errors
- Experiment and have fun!