# 2D Game Design Workshop - Part 2

**Unity Fundamentals: Movement, Physics, and Collisions**

Lucas P. Cordova, Ph.D.

In this workshop, you'll practice the fundamentals of 2D game development in Unity. By the end of Part 1, you'll have created a simple car game with movement controls, physics interactions, and a dynamic camera system.

This workshop is divided into three parts: - Part 1: Unity Fundamentals (this part) - Part 2: Enhancing Your Game with Physics and Collisions - Part 3: Polishing and Expanding Your Game

Each part builds upon the previous one, so be sure to complete them in order.

## Table of contents

# 1 Workshop Overview

In this three-part series, you'll learn the fundamentals of 2D game development in Unity. By the end of Part 2, you'll have created a simple car game with movement controls, physics interactions, and a dynamic camera system.

## 1.1 Learning Objectives

By completing Part 2, you will: - Work with Rigidbody2D and Collider2D components - Handle collision and trigger events - Set up a Cinemachine camera to follow your player

# 2 Part 2 Starter Project

If you haven't already, please complete Part 1 of the workshop to create the starter project for Part 2. Alternatively, you can clone the completed Part 1 project from here.

# 3 Step 1: Understanding Physics Components

## 3.1 Key Physics Concepts

### 3.1.1 Colliders

- Define the physical boundaries of GameObjects
- Determine what can be "hit" or "touched"
- Come in various shapes: Box, Circle, Polygon, Capsule
- Can be triggers (detect overlap) or solid (prevent overlap)

### 3.1.2 Rigidbodies

- Add physics simulation to GameObjects
- Enable gravity, forces, and realistic movement
- Required for physics-based collisions
- Control mass, drag, and other physical properties

# 4 Step 2: Collision Detection Rules

## 4.1 When Collisions Are Detected

| GameObject A | GameObject B | Result |
|---|---|---|
| Collider only | Collider only | No collision detected |
| Collider + Rigidbody | Collider only | Collision detected |
| Collider + Rigidbody | Collider + Rigidbody | Collision detected |

Best Practice

At least one GameObject in a collision must have a Rigidbody component for Unity to detect the collision.

# 5 Step 3: Add Physics to Your Car

## 5.1 Adding Components to Your Car

1. **Select the Car** in the Hierarchy

2. **Add a Rigidbody2D**:

   - Click "Add Component" in the Inspector
   - Search for "Rigidbody2D"
   - Click to add it

3. **Configure the Rigidbody2D**:

   - Set **Gravity Scale** to 0 (we don't want the car falling)
   - Set **Linear Damping** to 1 (adds some resistance)
   - Set **Angular Damping** to 1 (slows rotation)

# 6 Step 4: Add a Collider to Your Car

## 6.1 Completing the Physics Setup

1. **With the Car still selected**
2. **Add a Collider2D**:

   - Click "Add Component"
   - Choose "Capsule Collider 2D"
   - It should automatically fit your sprite

# 7 Step 5: Create a Static Wall

## 7.1 Static Obstacle (Wall)

1. **Create a Square sprite**:

   - Right-click in Hierarchy → 2D Object → Sprites → Square
   - Rename to "Wall"

2. **Position and scale**:

   - Position: (5, 0, 0)
   - Scale: (1, 3, 1) to make it tall

3. **Add a Box Collider 2D**:

   - Add Component → Box Collider 2D
   - No Rigidbody needed (static object)

4. **Change color** (optional):
    - Set Sprite Renderer color to gray5

# 8 Step 6: Create a Dynamic Box

## 8.1 Dynamic Obstacle (Box)

1. **Create another Square sprite**:
    - Right-click in Hierarchy → 2D Object → Sprites → Square
    - Rename to "Box"
2. **Position**:
    - Position: (-3, 2, 0)
3. **Add physics components**:
    - Add Component → Box Collider 2D
    - Add Component → Rigidbody2D
    - Set Gravity Scale to 0
4. **Change color** (optional):
    - Set Sprite Renderer color to brown

# 9 Step 7: Create a Trigger Zone

## 9.1 Trigger Zone (Checkpoint)

1. **Create a Circle sprite**:
    - Right-click in Hierarchy → 2D Object → Sprites → Circle
    - Rename to "Checkpoint"
2. **Position and scale**:
    - Position: (0, 5, 0)
    - Scale: (2, 2, 1)
3. **Add and configure collider**:
    - Add Component → Circle Collider 2D

- Check “Is Trigger” checkbox

4. **Change appearance**:

   - Set Sprite Renderer color to green
   - Set Color alpha to 0.5 for transparency

# 10 Step 8: Add Collision Detection Code

## 10.1 Implementing Collision Methods

Create a script called Collision to include collision detection:

```
using UnityEngine;
using UnityEngine.InputSystem;

public class Collision : MonoBehaviour
{
    [SerializeField] float moveSpeed = 5f;
    [SerializeField] float steerSpeed = 100;

    // Called when this collider/rigidbody hits another collider/rigidbody
    void OnCollisionEnter2D(Collision2D collision)
    {
        Debug.Log($"Car crashed into: {collision.gameObject.name}");

        // Different reactions based on what we hit
        if (collision.gameObject.name == "Wall")
        {
            Debug.Log("Ouch! Hit a wall!");
        }
        else if (collision.gameObject.name == "Box")
        {
            Debug.Log("Pushed a box!");
        }
    }
}
```

# 11 Step 9: Add Trigger Detection Code

## 11.1 Complete Collision and Trigger Detection

Add these methods to your Cruise script (after the Update method):

```
    // Called when collision ends
    void OnCollisionExit2D(Collision2D collision)
    {
        Debug.Log($"Car separated from: {collision.gameObject.name}");
    }

    // Called when entering a trigger zone
    void OnTriggerEnter2D(Collider2D other)
    {
        Debug.Log($"Car entered trigger: {other.gameObject.name}");

        if (other.gameObject.name == "Checkpoint")
        {
            Debug.Log("Checkpoint reached!");
            // You could add score, play sound, etc.
        }
    }

    // Called when exiting a trigger zone
    void OnTriggerExit2D(Collider2D other)
    {
        Debug.Log($"Car exited trigger: {other.gameObject.name}");
    }
```

# 12 Step 10: Test Collision Detection

## 12.1 Testing Your Physics

1. **Open the Console Window**:
   - Window → General → Console
   - Dock it somewhere visible
2. **Play the game** and test:
   - Cruise into walls - see "crashed into" messages

- Push boxes around - see collision messages
- Cruise through checkpoints - see trigger messages

> **Collision vs Trigger Events**
>
> **OnCollisionEnter2D / OnCollisionExit2D**:
>
> - Physical collisions where objects bounce/push
> - Both objects need colliders, at least one needs Rigidbody2D
> - Objects cannot pass through each other
>
> **OnTriggerEnter2D / OnTriggerExit2D**:
>
> - Detection zones where objects can pass through
> - Collider must have "Is Trigger" checked
> - Used for checkpoints, power-ups, detection areas

# 13 Step 11: Install Cinemachine

## 13.1 Adding the Cinemachine Package

1. **Open Package Manager**:
   - Window → Package Manager
2. **Select Unity Registry**:
   - Dropdown at top-left → Unity Registry
3. **Search for Cinemachine**:
   - Type "Cinemachine" in search bar
4. **Install**:
   - Click on Cinemachine package
   - Click "Install" button
   - Wait for installation to complete and then close the package manager

# 14 Step 12: Setup Cinemachine Brain

## 14.1 Configure Main Camera

1. **Add CinemachineCamera**:
    - Right-click the Scene in the Hiearchy and select Cinemachine → Cinemachine Camera
    - Set the Tracking Target to the Car GameObject
    - Change the Orthographic Size to 6 (zoom level)
    - Change the Positional Control to "Positional Composer"
    - Leave default settings

# 15 Step 13: Create Virtual Camera

## 15.1 Adding the Follow Camera

1. **Create Virtual Camera**:
    - Right-click in Hierarchy
    - Cinemachine → Targeted Camera → Follow Camera
    - Rename to "CarFollowCamera"

# 16 Step 14: Test the Camera

## 16.1 Verify Camera Following

1. Press Play
2. Cruise your car around using WASD
3. Notice how the camera smoothly follows your car
4. Try adjusting Damping values (1-3) to see different follow speeds

# 17 Testing Checklist

Before submitting your project, ensure all features work correctly:

### 17.1 Movement

- ☐ Car moves forward with W key or Up Arrow
- ☐ Car moves backward with S key or Down Arrow
- ☐ Car turns left with A key or Left Arrow
- ☐ Car turns right with D key or Right Arrow
- ☐ Movement is smooth and frame-rate independent

### 17.2 Physics

- ☐ Car collides with walls and stops
- ☐ Car can push boxes around
- ☐ Boxes collide with walls
- ☐ Console shows collision messages

### 17.3 Triggers

- ☐ Car can Cruise through checkpoints
- ☐ Console shows checkpoint messages
- ☐ Checkpoints appear semi-transparent

### 17.4 Camera

- ☐ Camera follows the car smoothly
- ☐ Car stays centered in view
- ☐ Camera movement has appropriate damping

## 18 Troubleshooting Guide

### 18.1 Common Issues and Solutions

#### 18.1.1 Car doesn't move / Keyboard not responding

- Check if the Input System is enabled (Edit → Project Settings → Player → Active Input Handling)
- Ensure Cruise script is attached to Car
- Verify speed values aren't 0
- Ensure Time Scale is 1 (Edit → Project Settings → Time)
- Make sure Game view has focus (click inside it)

#### 18.1.2 NullReferenceException with Keyboard.current

- The Input System Package may not be installed
- Go to Window → Package Manager
- Search for "Input System" and install it
- Restart Unity when prompted

#### 18.1.3 Collisions not detected

- Ensure at least one object has Rigidbody2D
- Check that both objects have Collider2D components
- Verify colliders aren't set as triggers when they shouldn't be

#### 18.1.4 Car falls or flies away

- Set Rigidbody2D Gravity Scale to 0
- Check Body Type is "Dynamic"
- Freeze Z rotation if car spins uncontrollably

#### 18.1.5 Camera doesn't follow

- Verify Car is assigned to Virtual Camera's Follow field
- Check Cinemachine Brain is on Main Camera

## 19 Resources and References

### 19.1 Unity Documentation

- Unity 2D Documentation
- Input System Documentation
- Transform Component
- Rigidbody2D
- Collider2D Overview
- Cinemachine Documentation

### 19.2 Keyboard Shortcuts

- **Play/Stop**: Ctrl/Cmd + P
- **Pause**: Ctrl/Cmd + Shift + P
- **Save Scene**: Ctrl/Cmd + S
- **Duplicate**: Ctrl/Cmd + D
- **Delete**: Delete key
- **Undo**: Ctrl/Cmd + Z
- **Focus GameObject**: F key (with object selected)

### 19.3 Best Practices Learned

1. Always use `Time.deltaTime` for movement
2. Use modern Input System (`Keyboard.current`) for input handling
3. Organize your Hierarchy with empty GameObjects
4. Test frequently during development
5. Use Debug.Log for troubleshooting
6. Save your work often

## 20 Conclusion

Congratulations on completing Part 2 of the 2D Game Design Workshop! You've learned fundamental Unity concepts including transforms, physics, collisions, and camera systems. These skills form the foundation for any 2D game development project.

In Part 3, you'll enhance your game further by adding art assets, sound effects, and polishing gameplay mechanics. Keep experimenting and building on what you've learned!