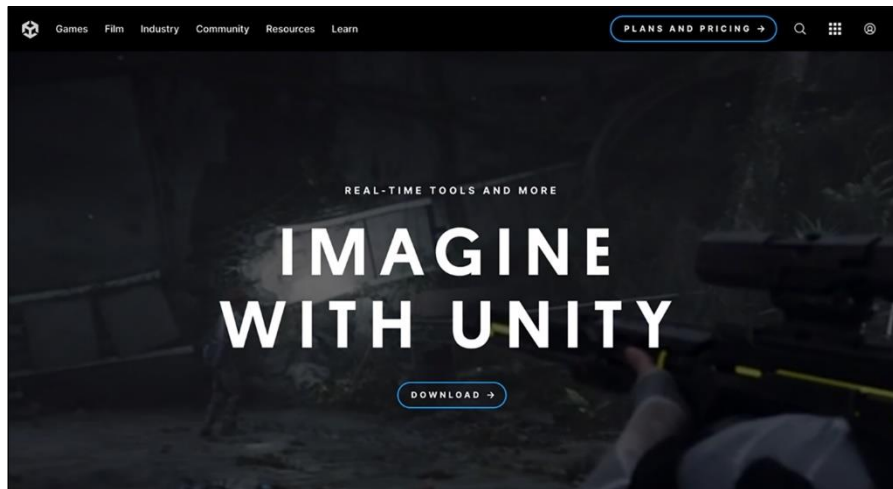


Chapter 1: Getting to Know Your Environment



Create with Unity in three steps

1. Download the Unity Hub

Follow the instructions onscreen for guidance through the installation process and setup.

[Download for Windows](#)

[Download for Mac](#)

[Instructions for Linux](#)

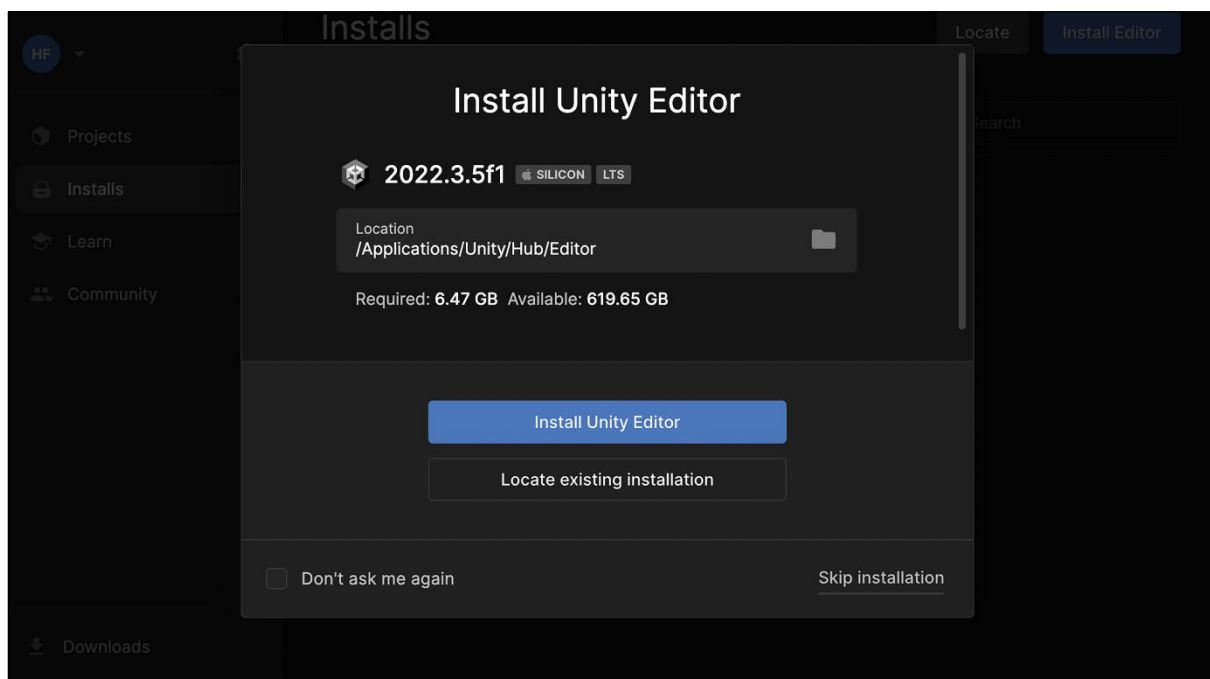
2. Choose your Unity version

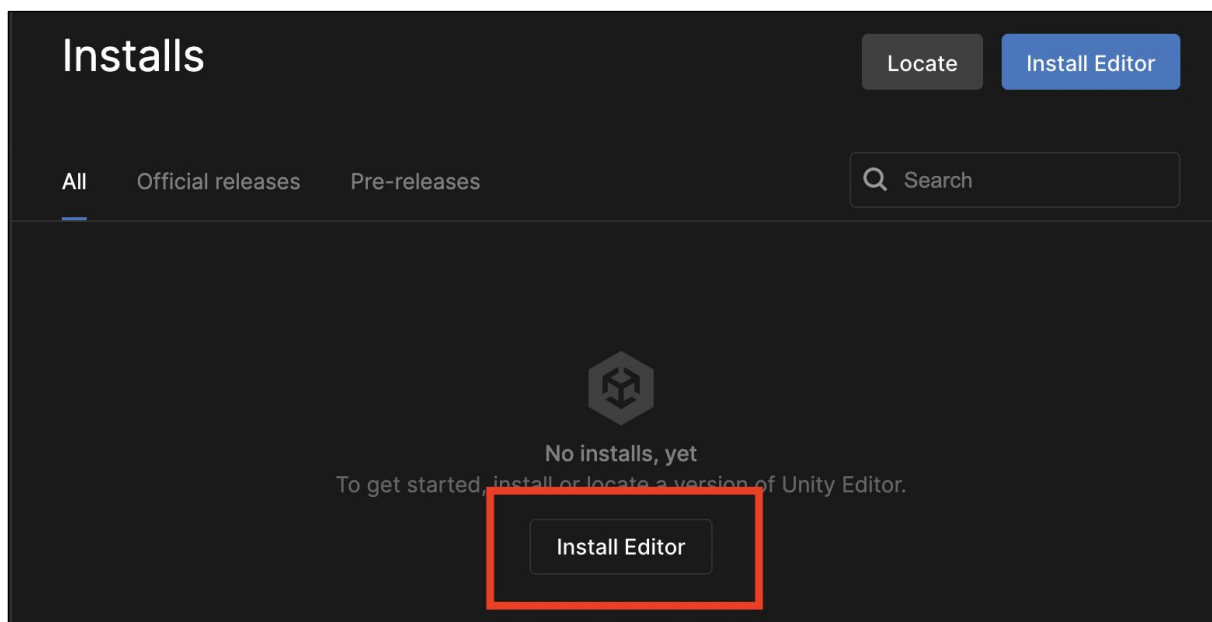
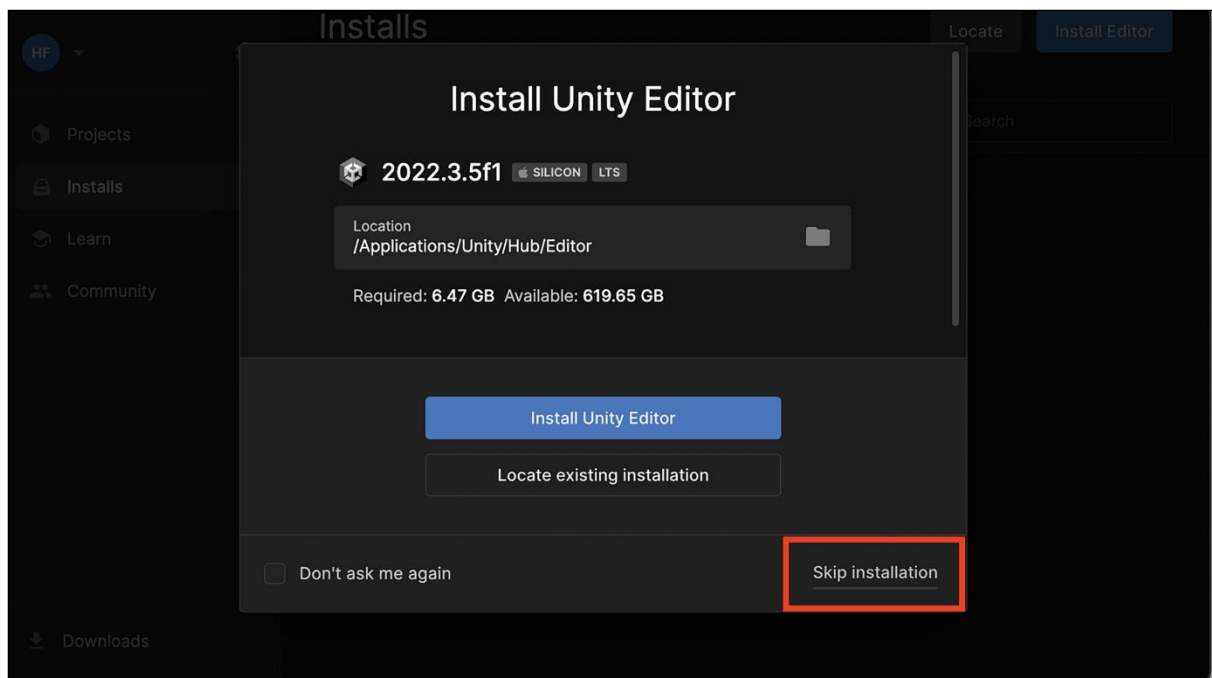
Install the latest version of Unity, an older release, or a beta featuring the latest in-development features.

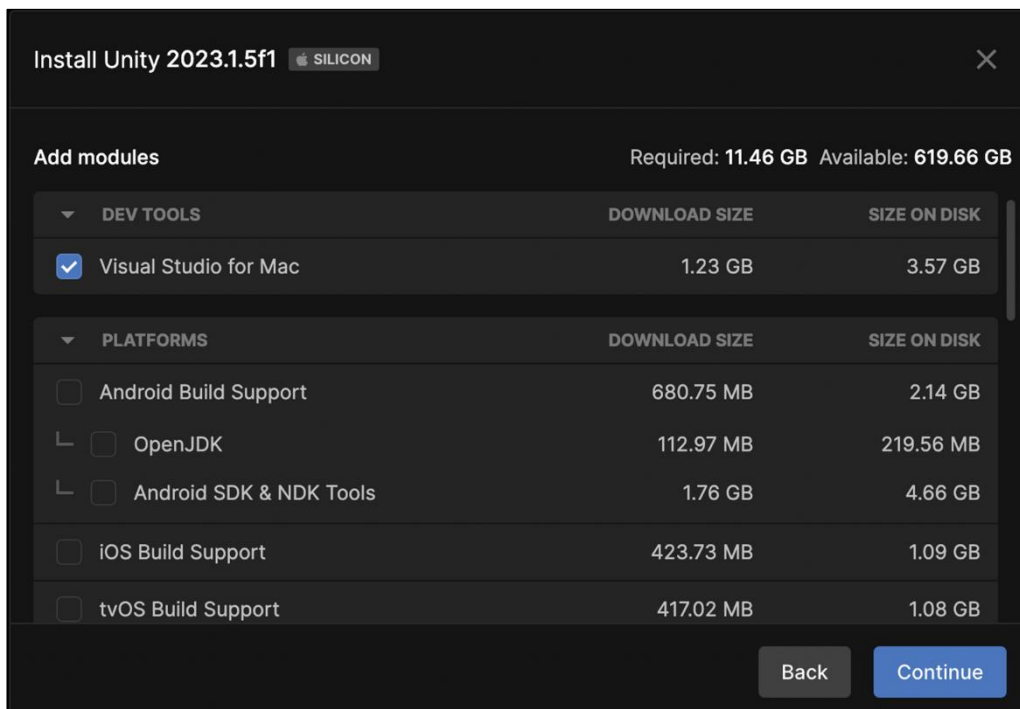
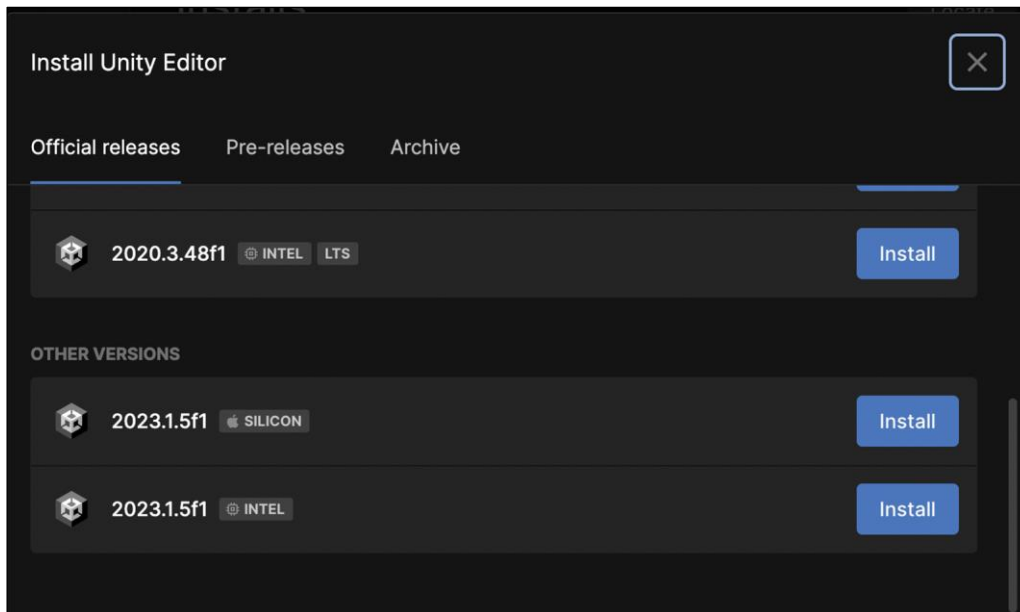
[Visit the download archive](#)

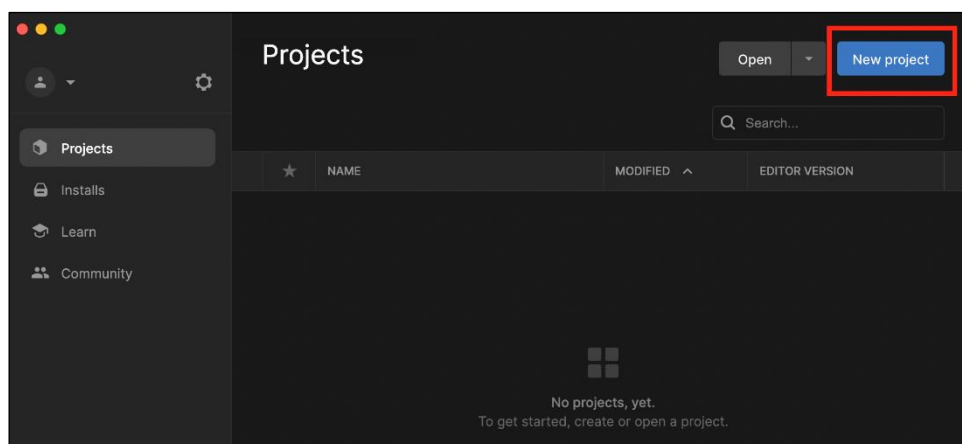
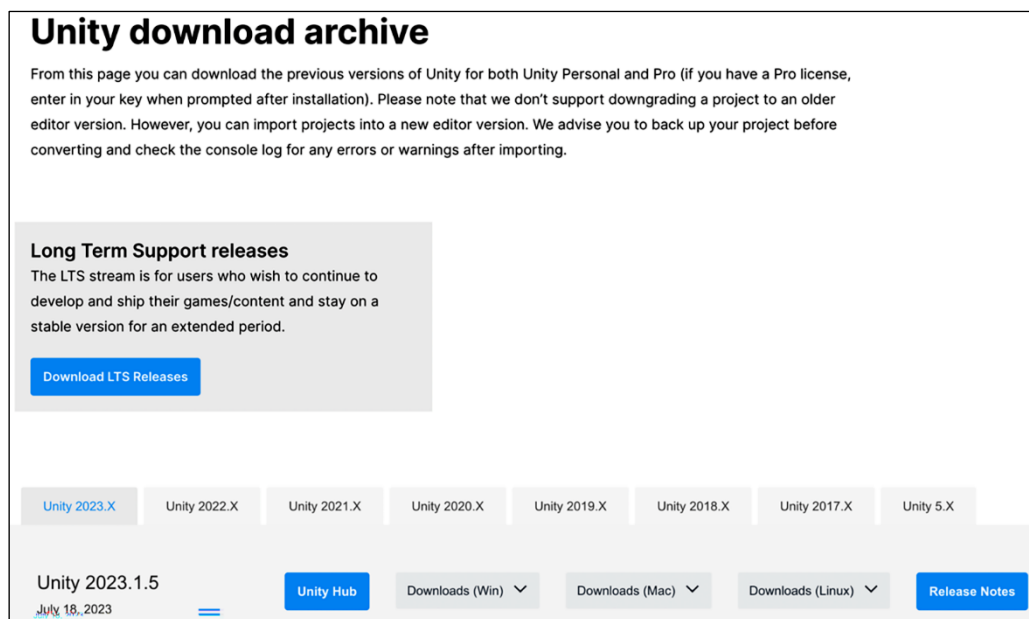
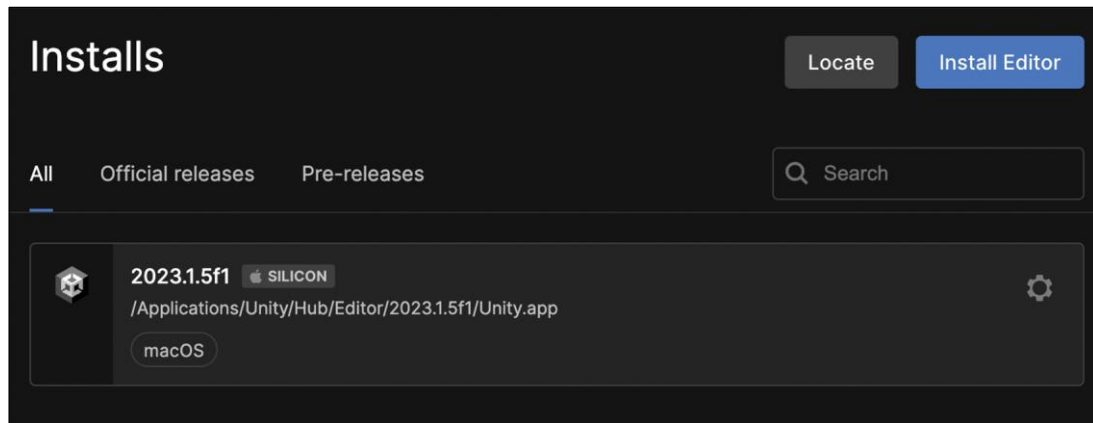
3. Start your project

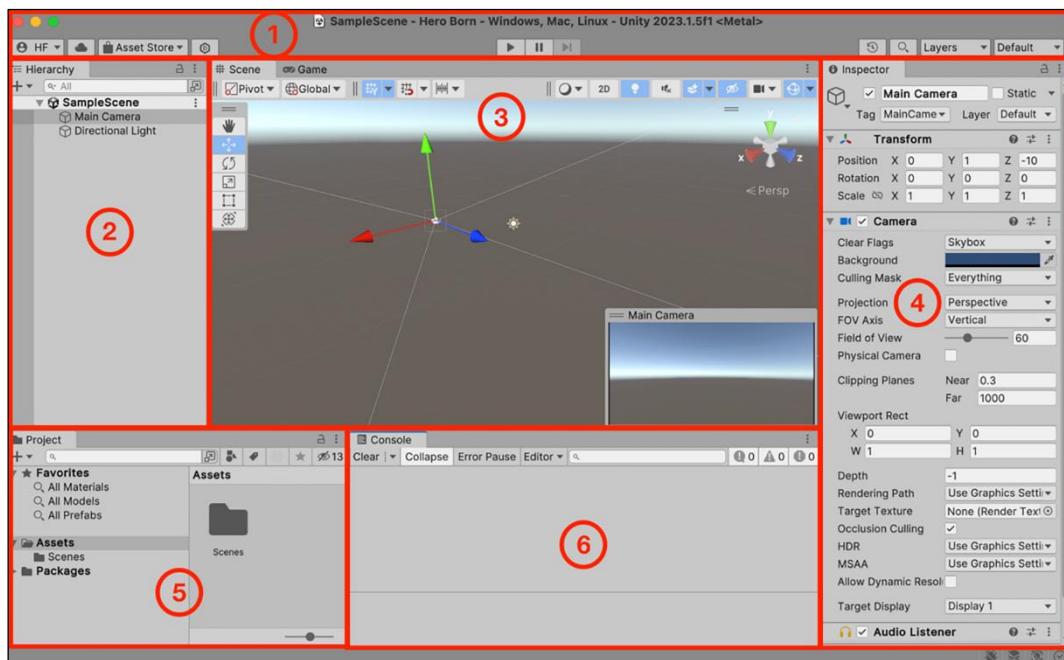
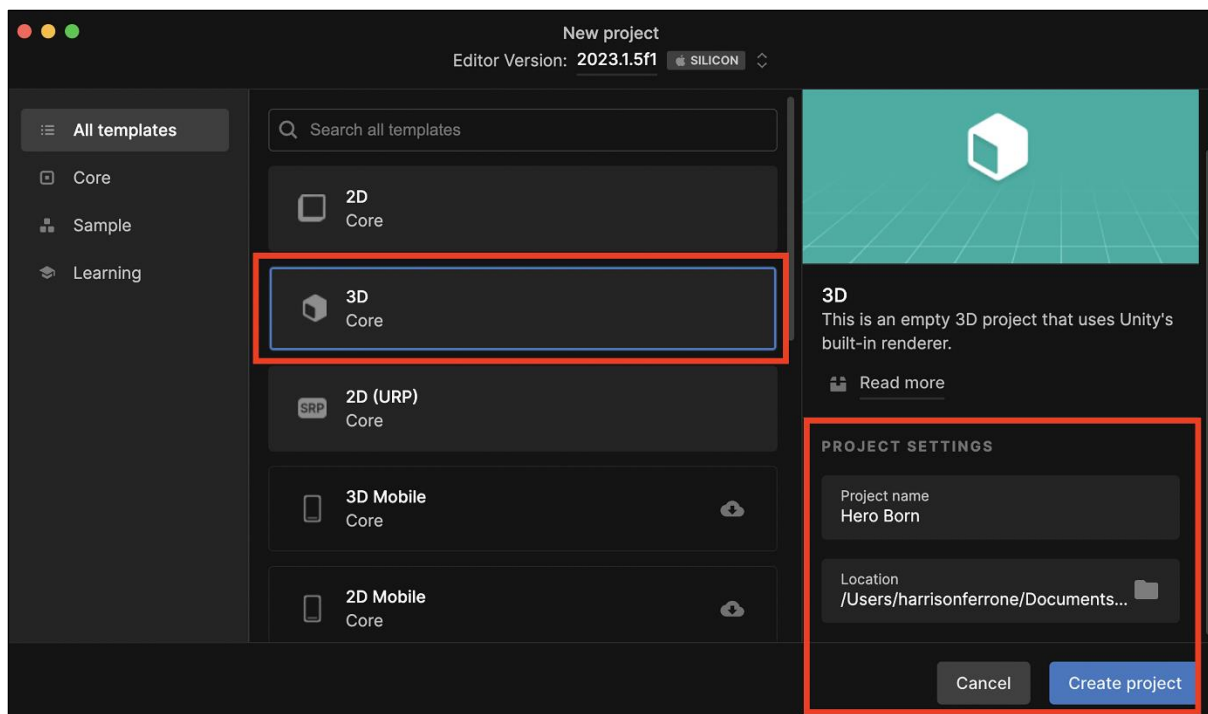
Begin creating from scratch, or pick a template to get your first project up and running quickly. Access tutorial videos designed to support creators, from beginners to experts.

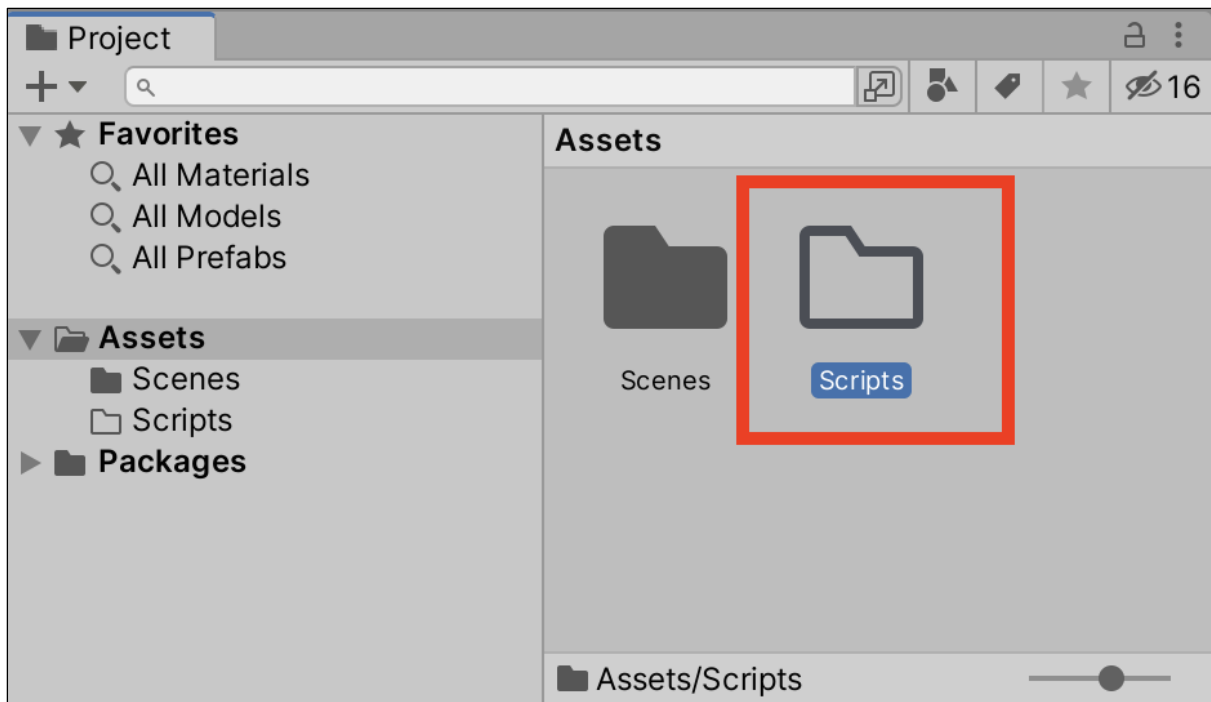
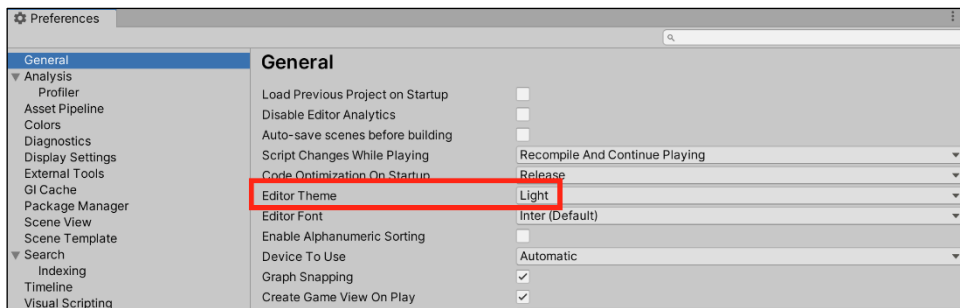
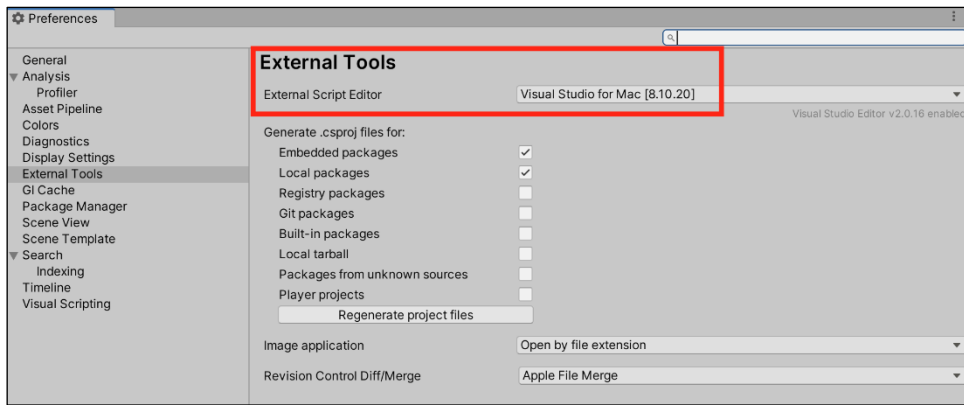


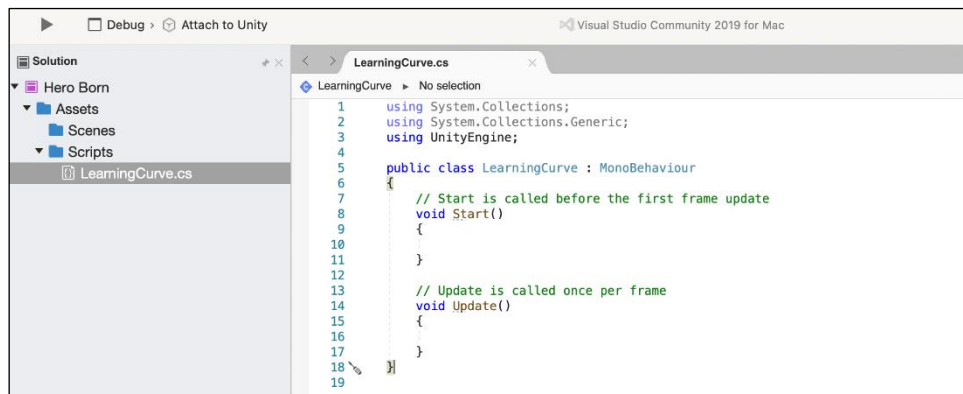
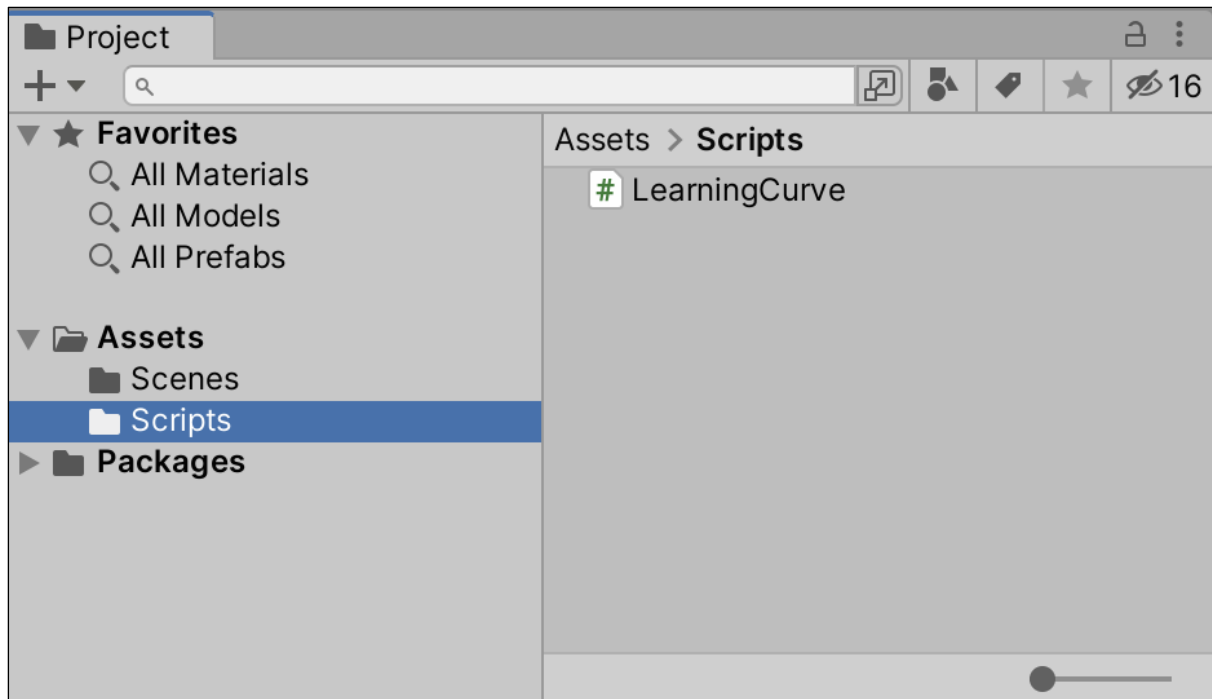


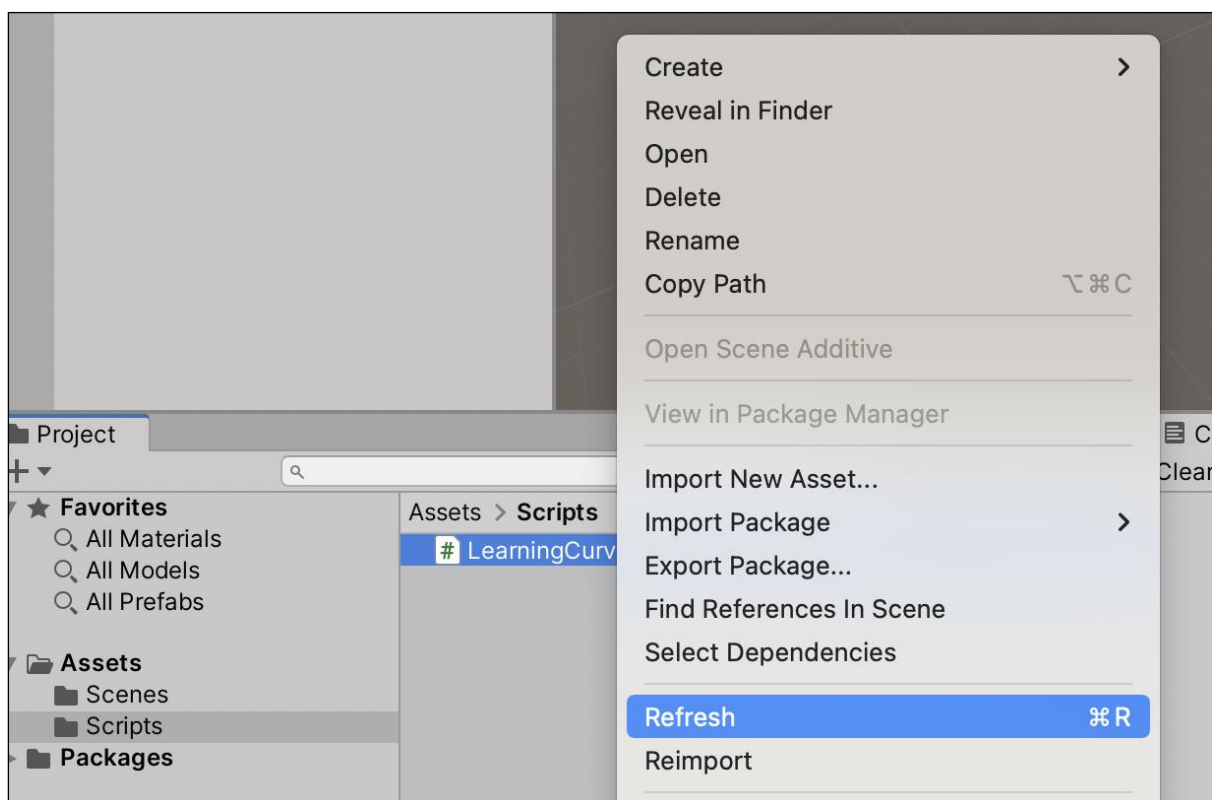
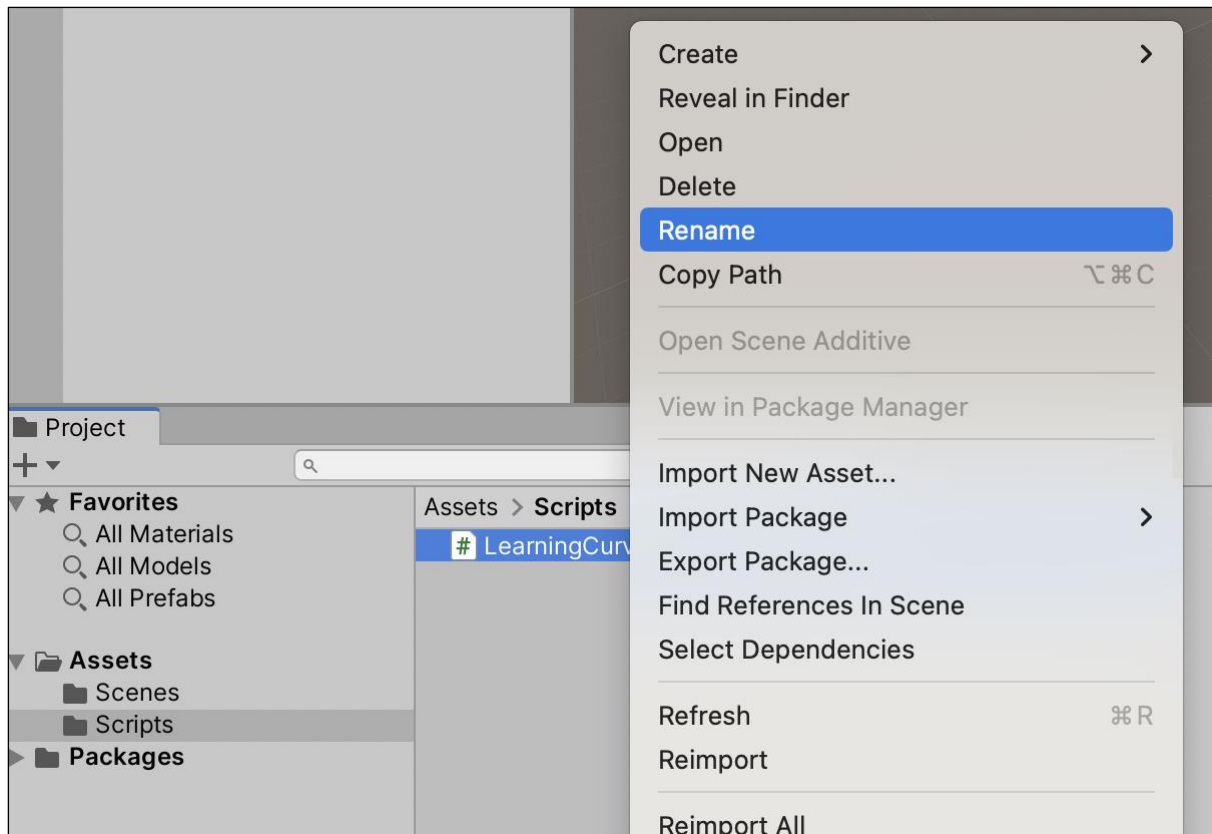


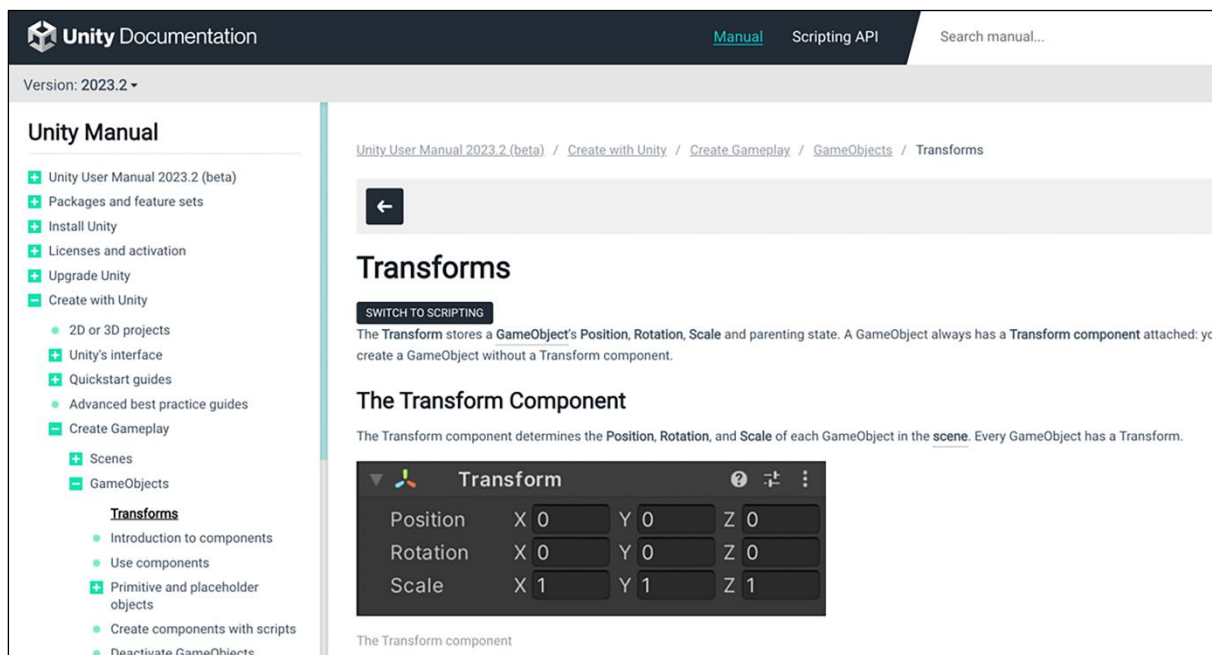
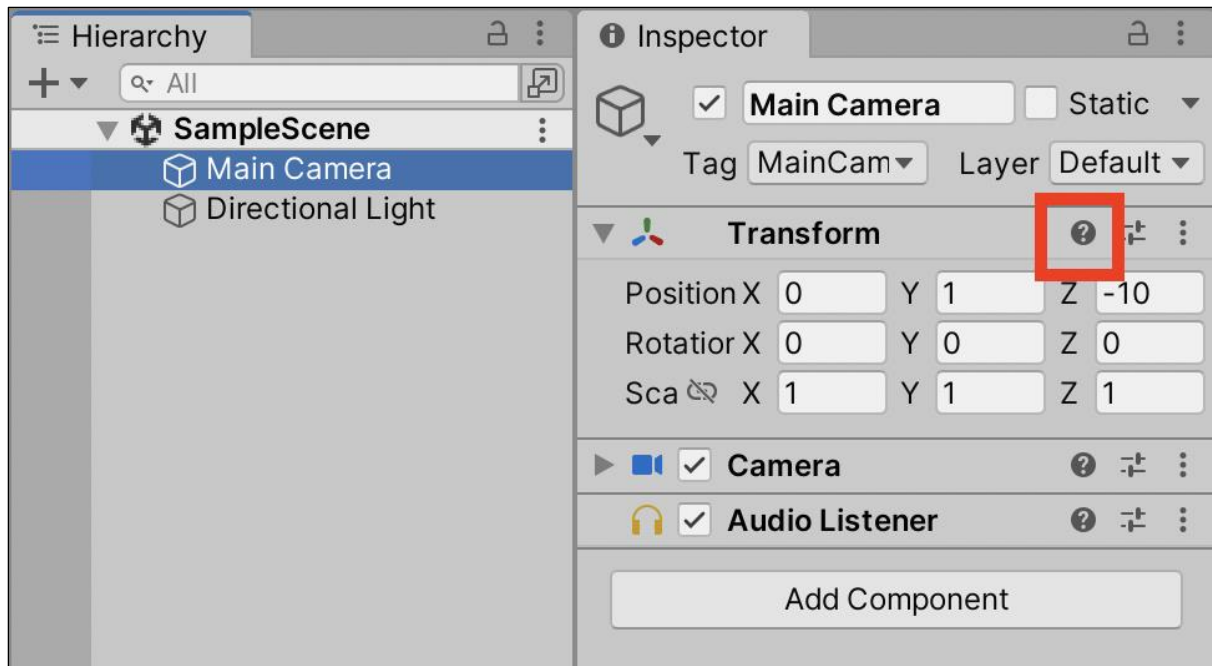














Transforms

SWITCH TO SCRIPTING

The **Transform** is used to store a **GameObject**'s position, rotation, scale and parenting state and is thus very important. A **GameObject** attached - it is not possible to remove a **Transform** or to create a **GameObject** without one.

The Transform Component

The **Transform** component determines the **Position**, **Rotation**, and **Scale** of each object in the **scene**. Every **GameObject** has a **Transform**

Transform			
Position	X 0	Y 0	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Transform

class in [UnityEngine](#) / Inherits from: [Component](#) / Implemented in: [UnityEngine.CoreModule](#)

SWITCH TO MANUAL


Description

Position, rotation and scale of an object.

Every object in a Scene has a Transform. It's used to store and manipulate the position, rotation and scale of the object. Every Transform can have a parent and apply position, rotation and scale hierarchically. This is the hierarchy seen in the Hierarchy pane. They also support enumerators so you can loop through

```
using UnityEngine;

public class Example : MonoBehaviour
{
    // Moves all transform children 10 units upwards!
    void Start()
    {
        foreach (Transform child in transform)
        {
            child.position += Vector3.up * 10.0f;
        }
    }
}
```



Docs Documentation Learn Certifications Q&A Code Samples Shows Events

Search

Sign in

.NET Languages Workloads APIs Resources

Download .NET

Filter by title

C# documentation

- Get started
- Fundamentals
- What's new in C#
- Tutorials
- C# concepts
- How-to C# articles
- The .NET Compiler Platform SDK (Roslyn APIs)
- C# programming guide
 - Overview
 - Programming concepts
 - Statements, expressions, and equality
 - Types
 - Classes, Structs, and Records
 - Interfaces
 - Delegates
 - Arrays
 - Strings
 - Indexers
 - Events
 - Generics

Download PDF

Docs / .NET / C# guide /

C# programming guide

Article • 03/31/2022 • 2 minutes to read • 16 contributors

Like

Comment

In this article

[Program sections](#)[Language Sections](#)[Platform Sections](#)[See also](#)

This section provides detailed information on key C# language features and features accessible to C# through .NET.

Most of this section assumes that you already know something about C# and general programming concepts. If you are a complete beginner with programming or with C#, you might want to visit the [Introduction to C# Tutorials](#) or [.NET In-Browser Tutorial](#), where no prior programming knowledge is required.

For information about specific keywords, operators, and preprocessor directives, see [C# Reference](#). For information about the C# Language Specification, see [C# Language Specification](#).

Program sections

[Inside a C# Program](#)

[Main\(\) and Command-Line Arguments](#)

Language Sections

[Statements](#) [Operators and expressions](#) [Expression-bodied members](#) [Equality](#) [Comparisons](#)

[Types](#)

Filter by title

C# documentation

- Get started
- Fundamentals
- What's new in C#
- Tutorials
- C# concepts
- How-to C# articles
- The .NET Compiler Platform SDK (Roslyn APIs)
- C# programming guide
 - Overview
 - Programming concepts
 - Statements, expressions, and equality
 - Types
 - Classes, Structs, and Records
 - Interfaces
 - Delegates
 - Arrays
 - Strings
 - Programming with strings**
 - How to determine whether a string represents a numeric value

Docs / .NET / C# guide / Programming guide /

Strings and string literals

Article • 06/21/2022 • 16 minutes to read • 20 contributors

Like

Comment

A string is an object of type `String` whose value is text. Internally, the text is stored as a sequential read-only collection of `Char` objects. There's no null-terminating character at the end of a C# string; therefore a C# string can contain any number of embedded null characters (`'\0'`). The `Length` property of a string represents the number of `Char` objects it contains, not the number of Unicode characters. To access the individual Unicode code points in a string, use the `StringInfo` object.

string vs. System.String

In C#, the `string` keyword is an alias for `String`. Therefore, `String` and `string` are equivalent, regardless it's recommended to use the provided alias `string` as it works even without `using System;`. The `String` class provides many methods for safely creating, manipulating, and comparing strings. In addition, the C# language overloads some operators to simplify common string operations. For more information about the keyword, see [string](#). For more information about the type and its methods, see [String](#).

Declaring and initializing strings

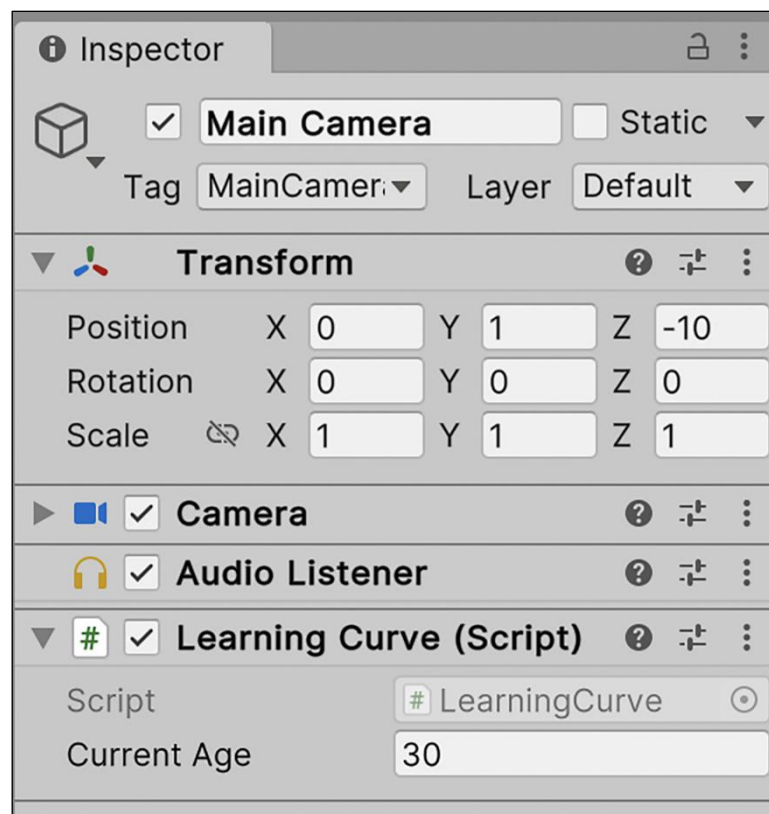
You can declare and initialize strings in various ways, as shown in the following example:

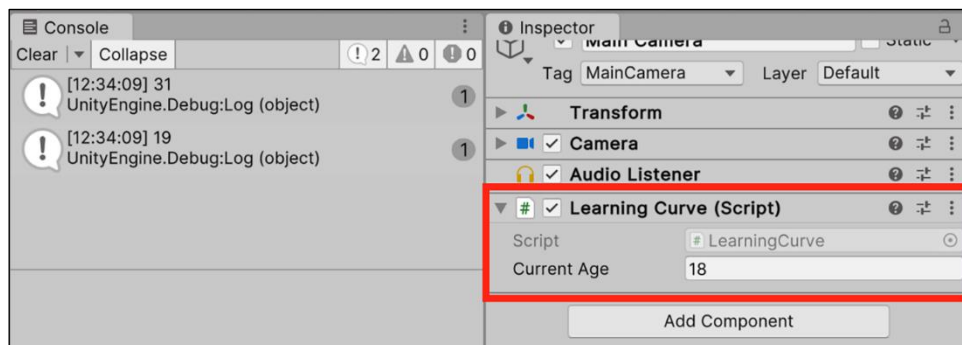
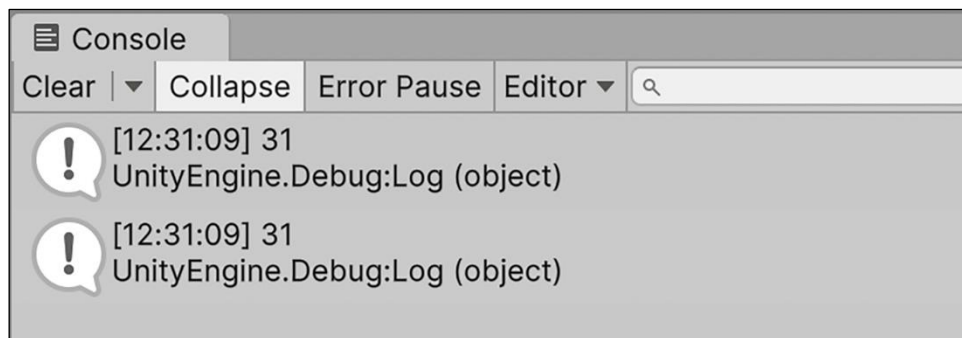
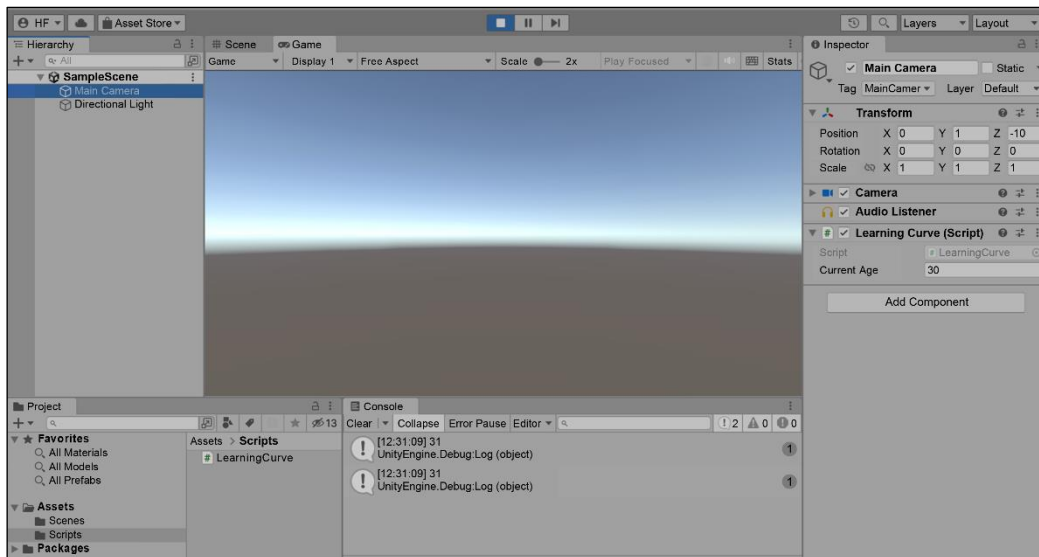
Chapter 2: The Building Blocks of Programming



```
LearningCurve.cs
LearningCurve ▶ Start()

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class LearningCurve : MonoBehaviour
6  {
7      public int CurrentAge = 30;
8
9      // Start is called before the first frame update
10     void Start()
11     {
12         Debug.Log(30 + 1);
13         Debug.Log(CurrentAge + 1);
14     }
15
```





```
LearningCurve.cs
LearningCurve ▶ ComputeAge()

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class LearningCurve : MonoBehaviour
6  {
7      public int CurrentAge = 30;
8      public int AddedAge = 1;
9
10     // Start is called before the first frame update
11     void Start()
12     {
13         ComputeAge();
14     }
15
16     void ComputeAge()
17     {
18         Debug.Log(CurrentAge + AddedAge);
19     }
20
```

← Calling the method

← The method

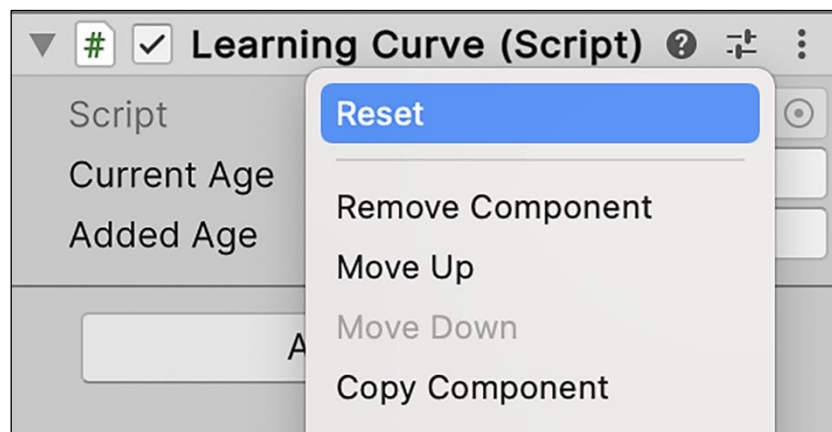
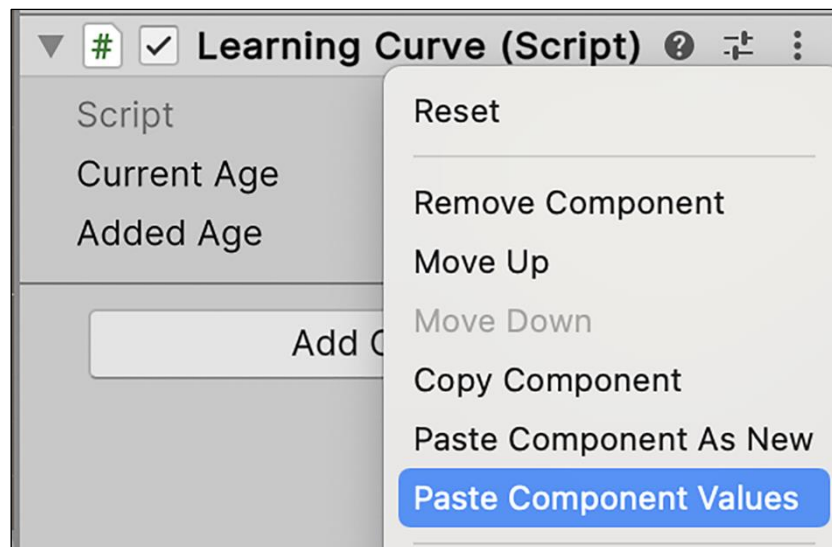
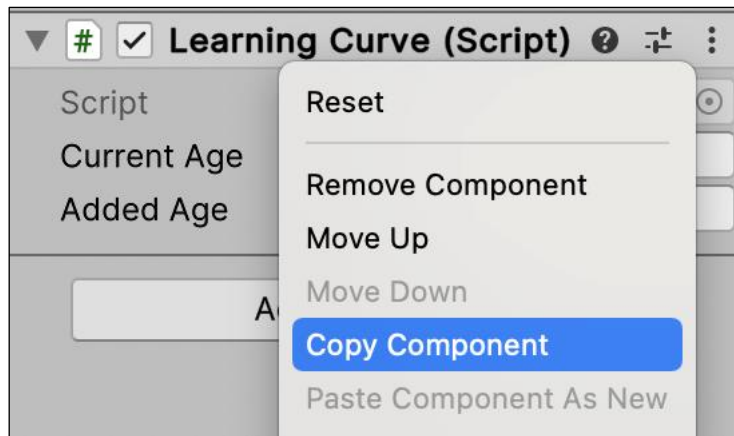
```
Console
Clear | Collapse | Error Pause | Editor | 🔍

[12:42:11] 19
UnityEngine.Debug:Log (object)
```

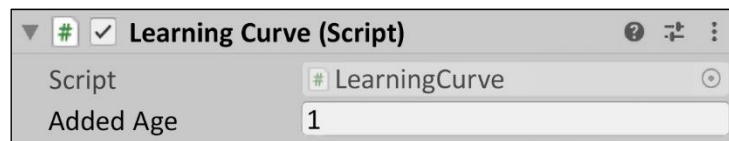
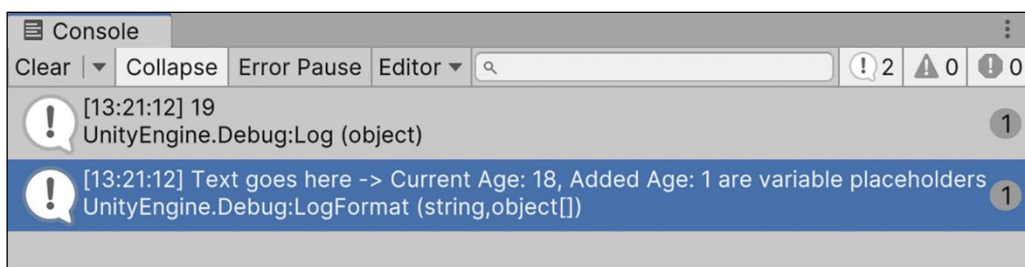
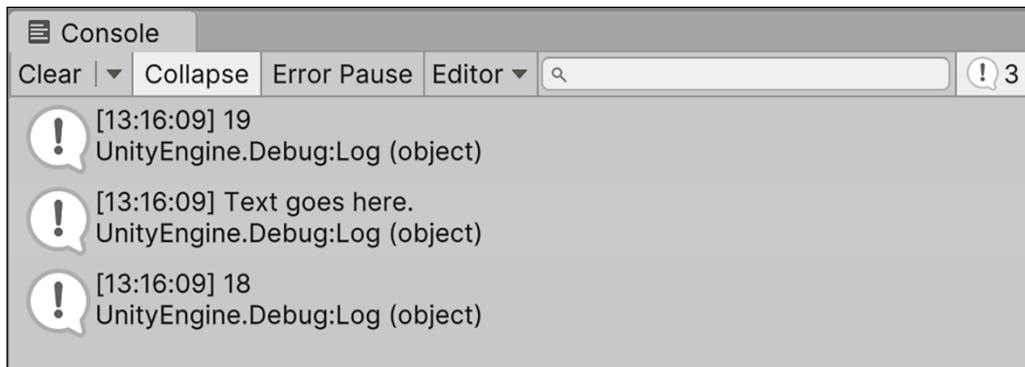
```
16     /// <summary>
17     /// Computes a modified age by adding two variables together
18     /// </summary>
19     void ComputeAge()
20     {
21         Debug.Log(CurrentAge + AddedAge);
22     }
23
```

```
// Start is called before the first frame update
void Start()
{
    ComputeAge();
}

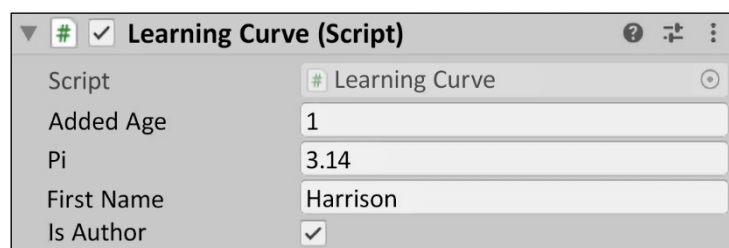
/// <summary>
/// Computes a modified age by adding two variables together
/// </summary>
void ComputeAge()
{
    Debug.Log(CurrentAge + AddedAge);
}
```

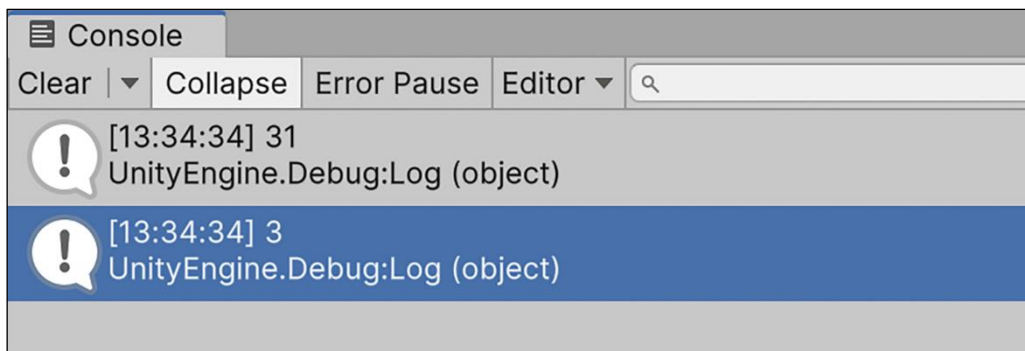
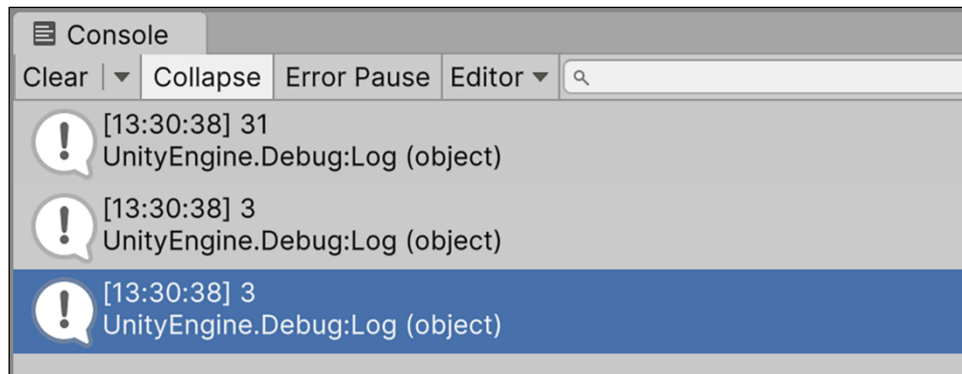
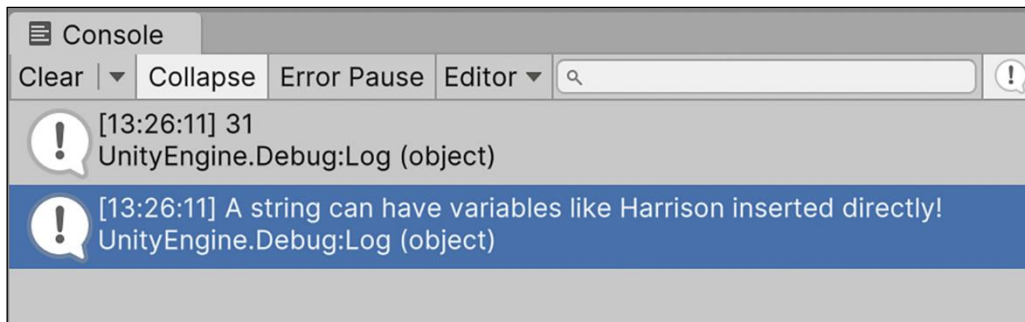


Chapter 3: Diving into Variables, Types, and Methods



Type	Contents of the variable
int	A simple integer, such as the number 3
float	A number with a decimal, such as the number 3.14
string	Characters in double quotes, such as, "Watch me go now"
bool	A Boolean, either true or false

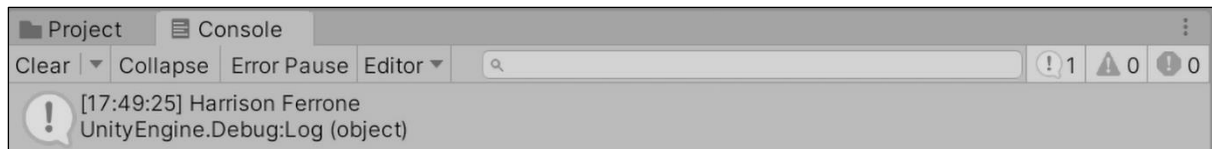




```
5 public class LearningCurve : MonoBehaviour
6 {
7     public string CharacterClass = "Ranger";
8
9     // Start is called before the first frame update
10    void Start()
11    {
12        int CharacterHealth = 100;
13        Debug.Log(CharacterClass + " - HP: " + CharacterHealth);
14    }
15
16    void CreateCharacter()
17    {
18        string CharacterName = "Aragorn";
19        Debug.Log(CharacterName + " - " + CharacterClass);
20    }
21 }
```

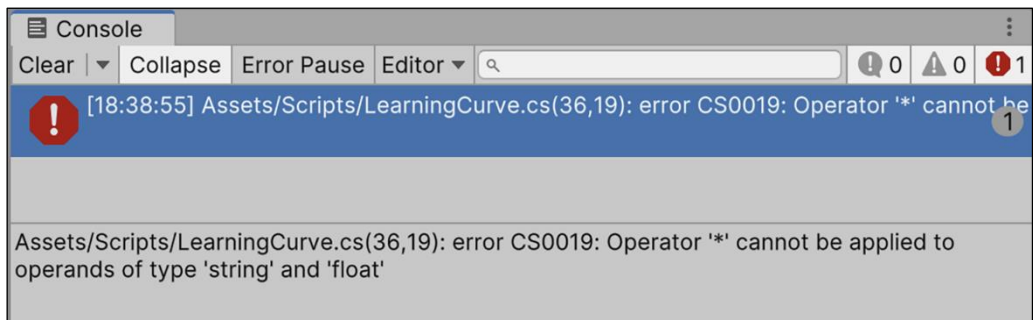
Annotations in the code:

- Class scope**: Points to the `public string CharacterClass = "Ranger";` line.
- Local scope 1**: Points to the `int CharacterHealth = 100;` line.
- Local scope 2**: Points to the `string CharacterName = "Aragorn";` line.

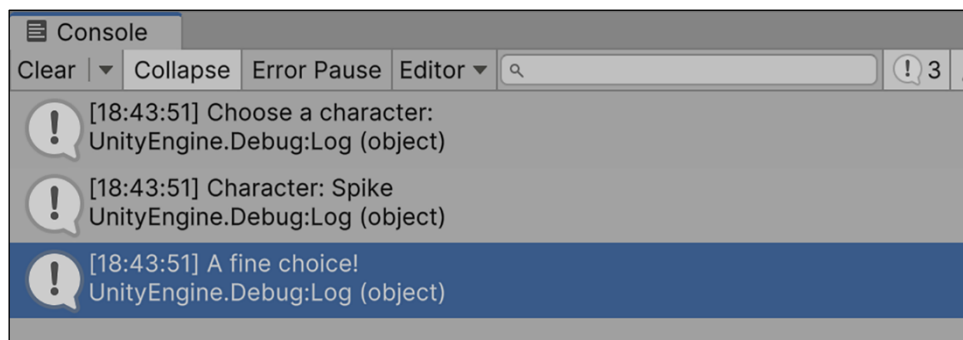


```
Debug.Log(FirstName * Pi);
```

(field) string LearningCurve.FirstName
CS0019: Operator '*' cannot be applied to operands of type 'string' and 'float'



```
13 // Use this for initialization
14 void Start ()
15 {
16     Debug.Log("Choose" a character.");
17     GenerateCharacter();
18     Debug.Log("A fine choice.");
19 }
20
21 public void GenerateCharacter()
22 {
23     Debug.Log("Character: Spike");
24 }
```



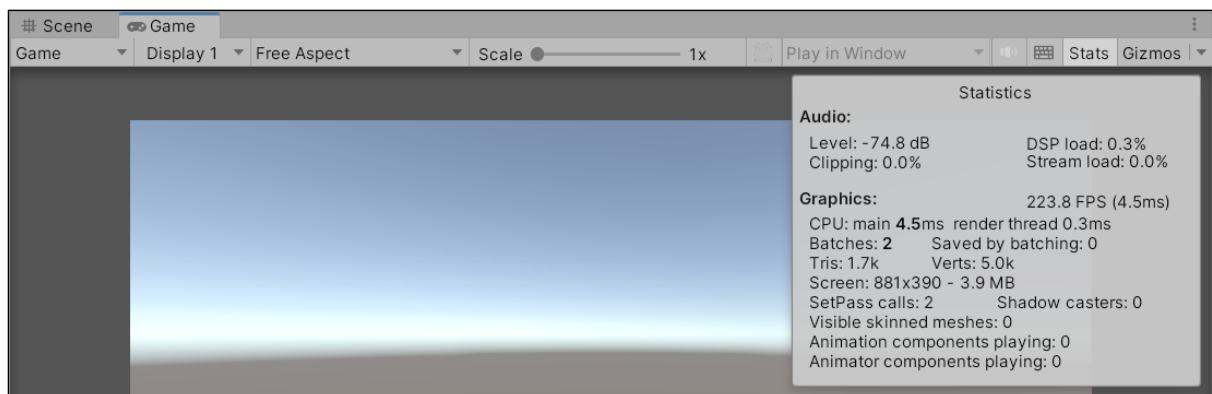
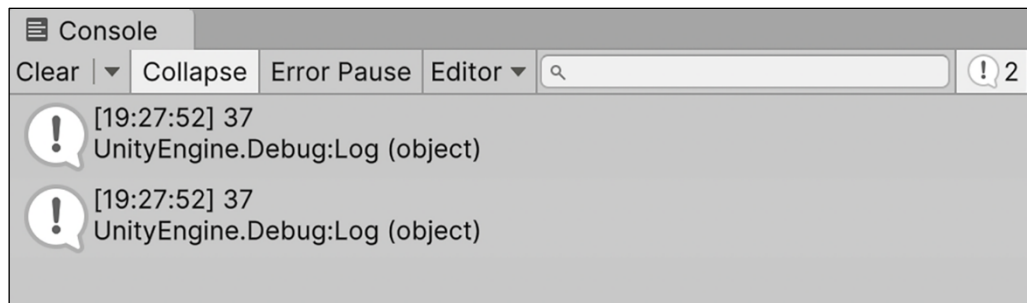
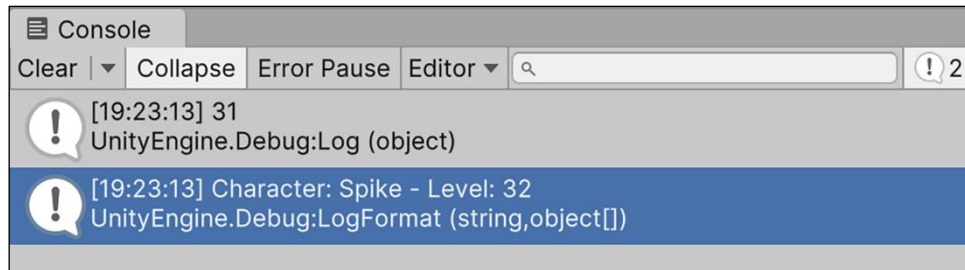
```

13 // Use this for initialization
14 void Start ()
15 {
16     int characterLevel = 32;
17     GenerateCharacter("Spike", characterLevel);
18 }
19
20 public void GenerateCharacter(string name, int level)
21 {
22     Debug.LogFormat("Character: {0} - Level: {1}", name, level);
23 }

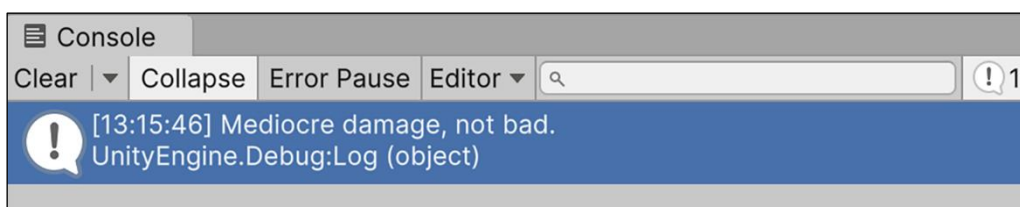
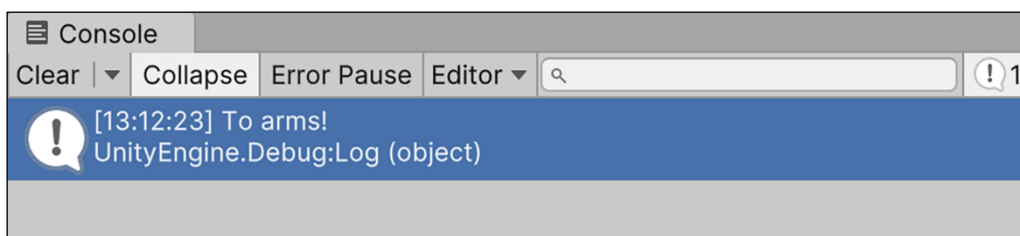
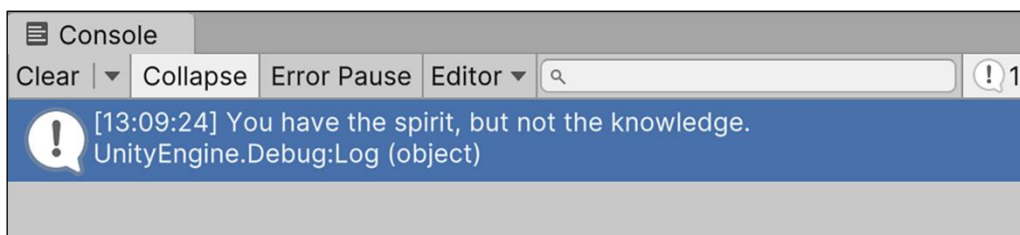
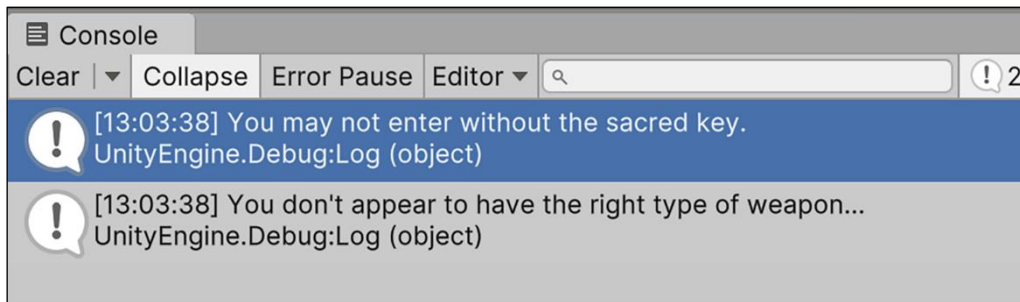
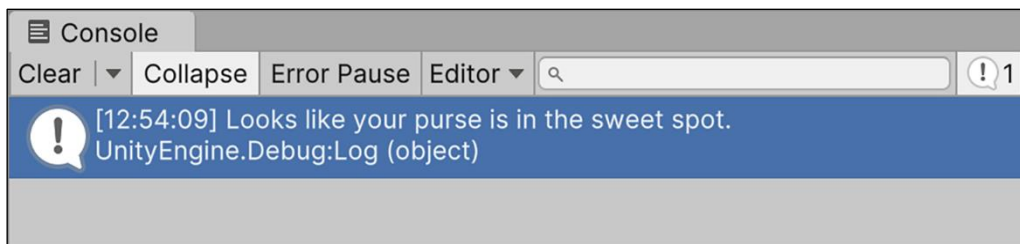
```

Arguments

Parameters



Chapter 4: Control Flow and Collection Types

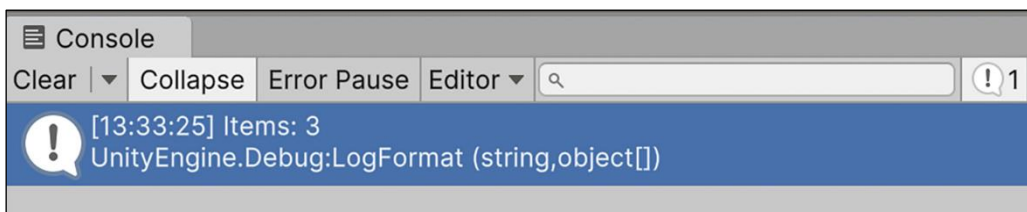
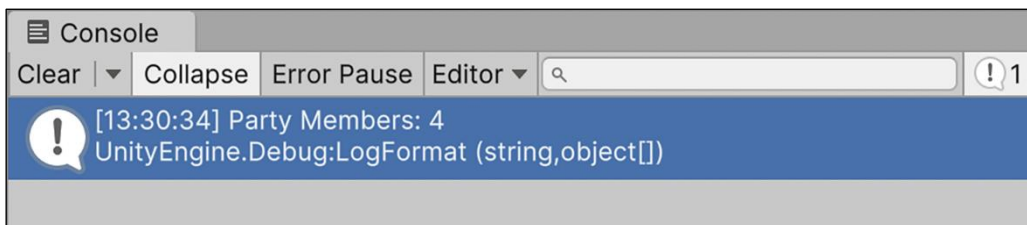
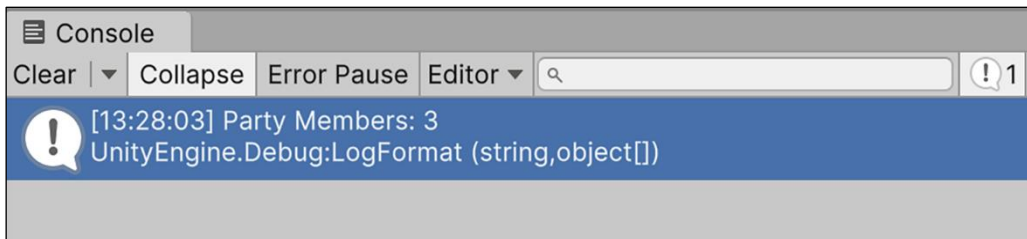
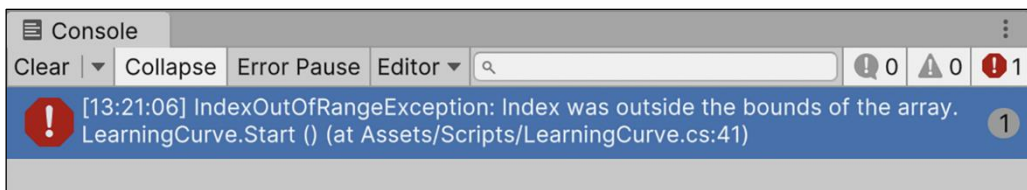


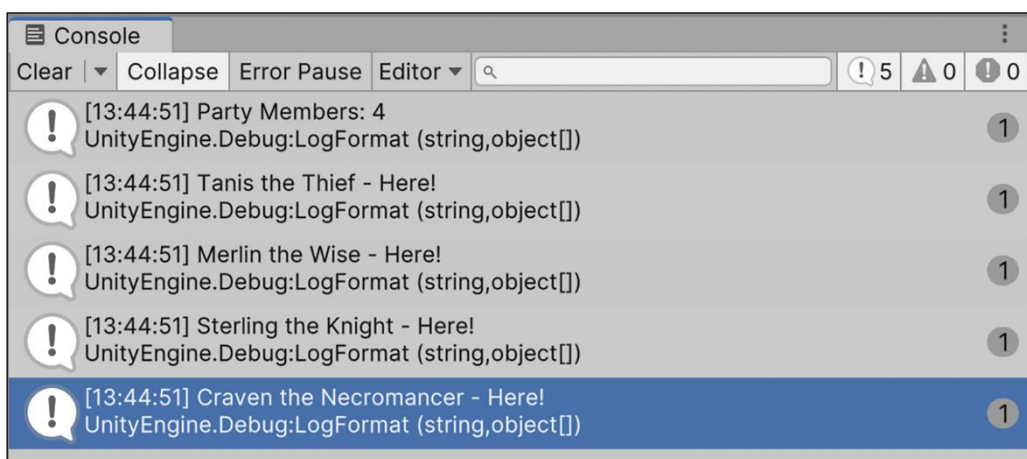
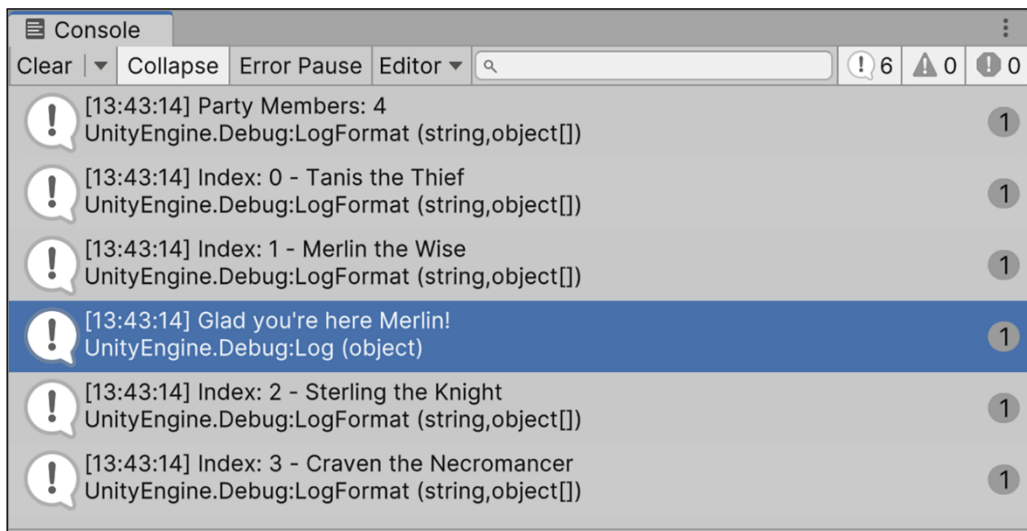
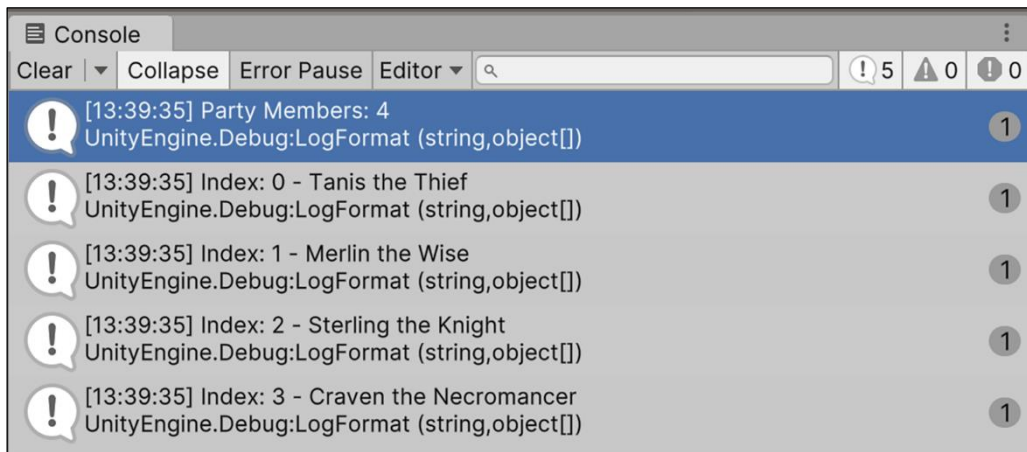
Index

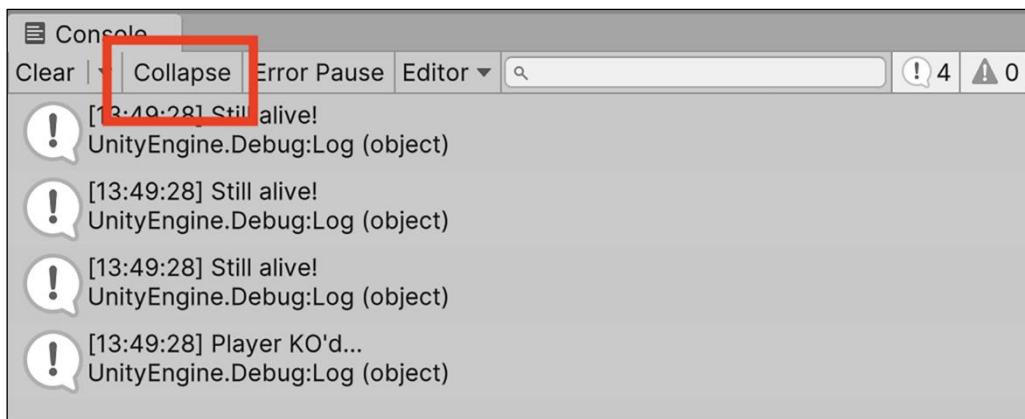
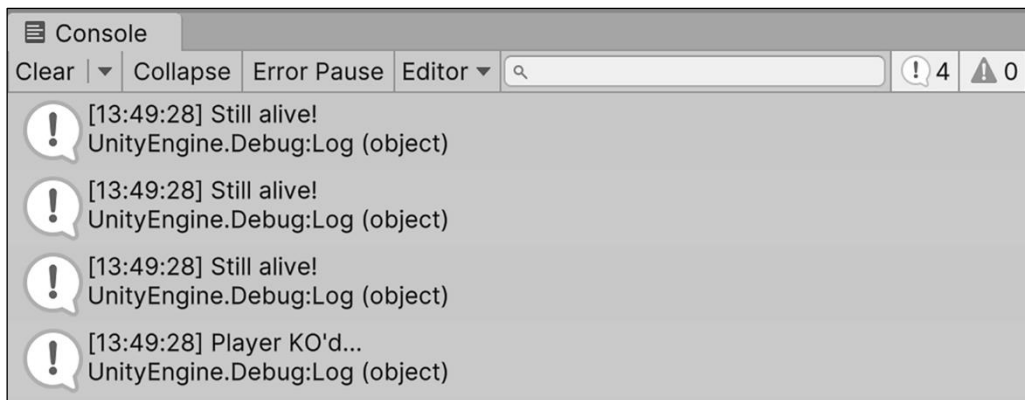
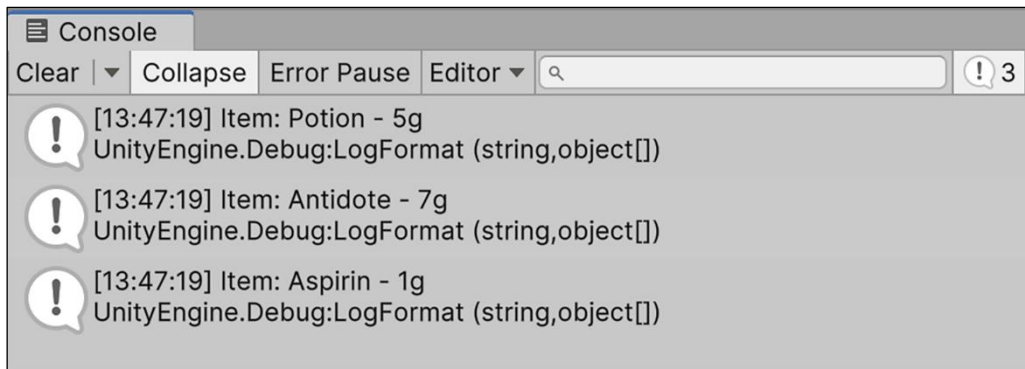
0 1 2

```
int[] topPlayerScores = [452, 713, 984];
```

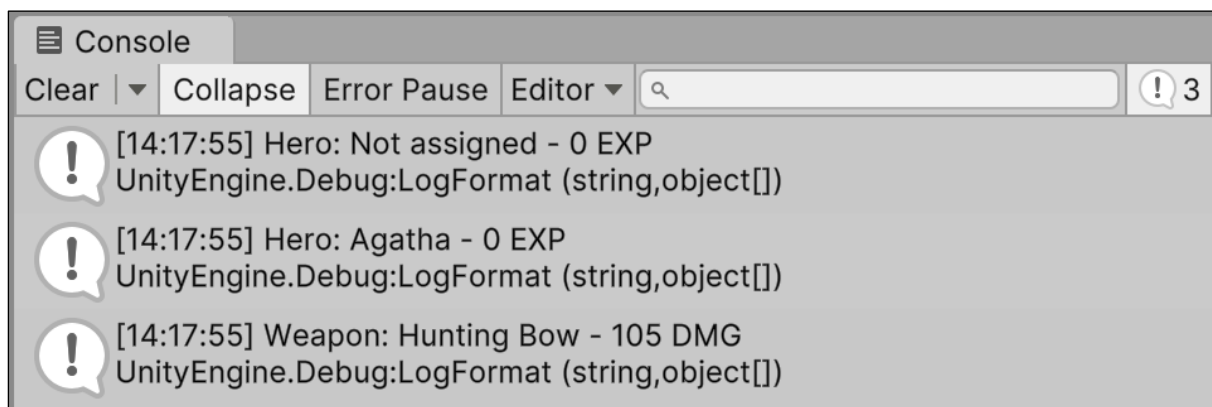
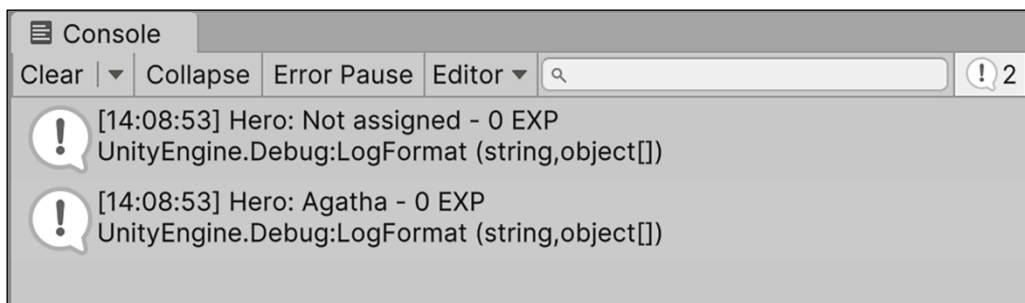
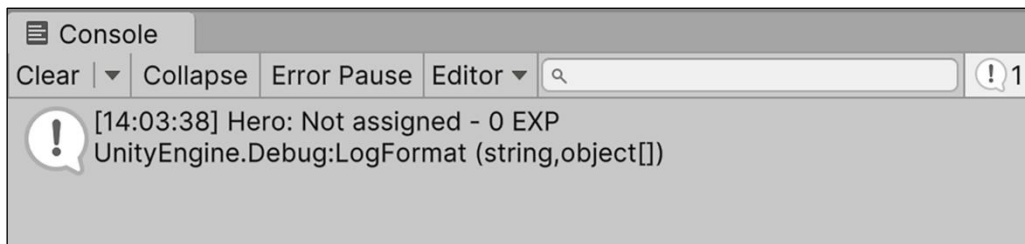
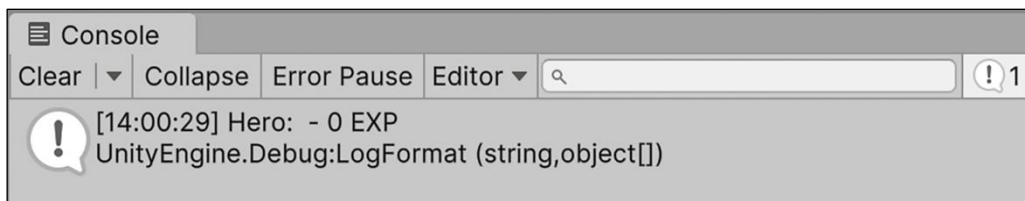
```
int[,] Coordinates = new int[3, 2]
{
    Columns
    0   1
    {5, 4}, Index 0 (or row 0)
    {1, 7}, Index 1 (or row 1)
    {9, 3} Index 2 (or row 2)
};
```

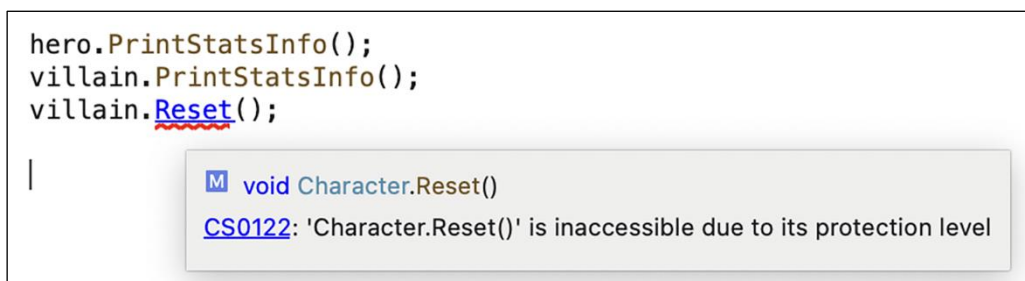
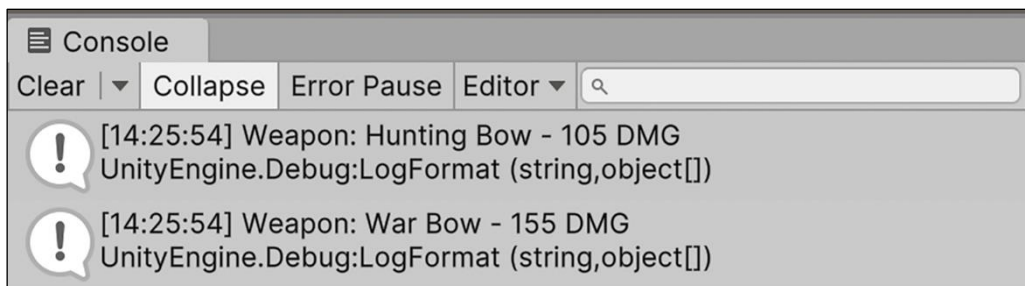
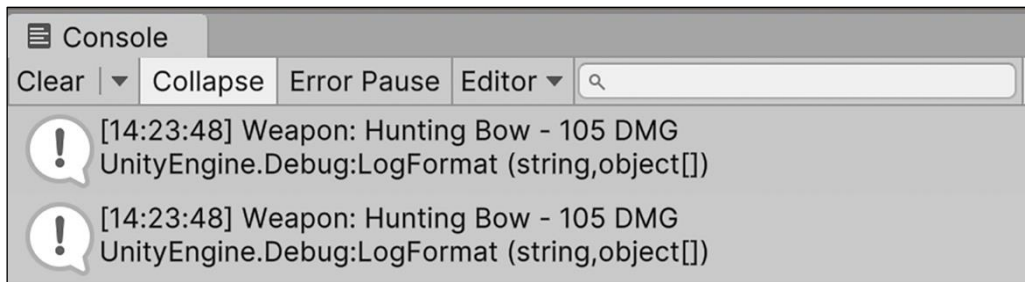
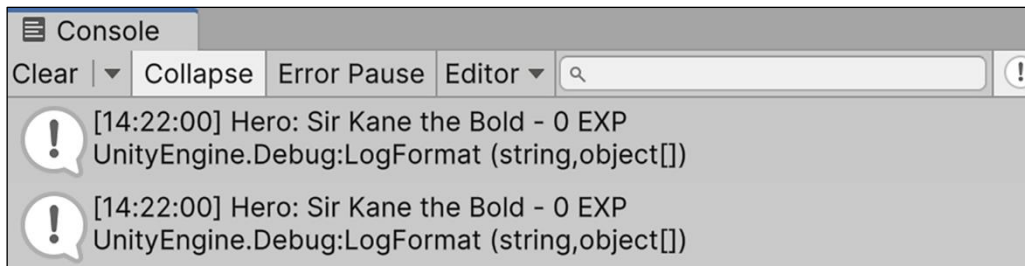
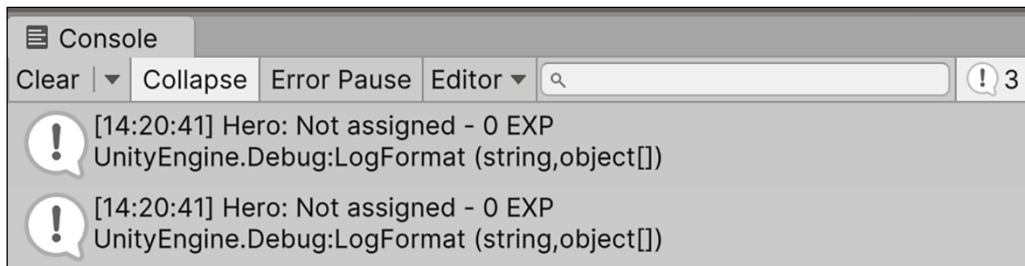


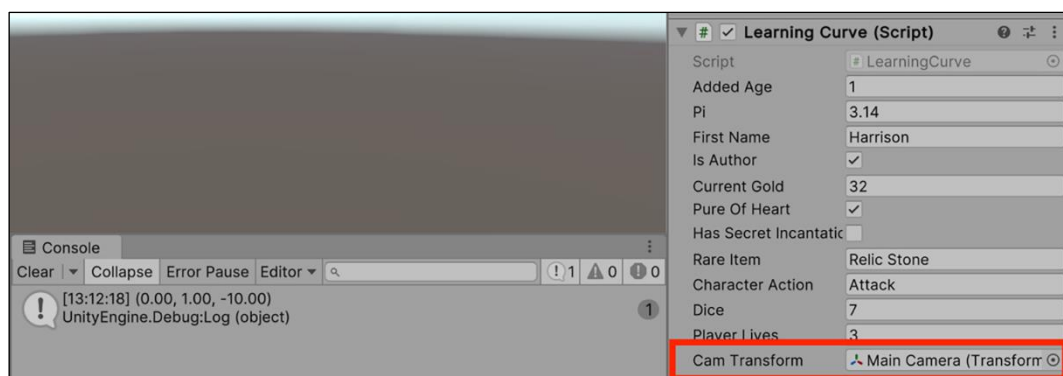
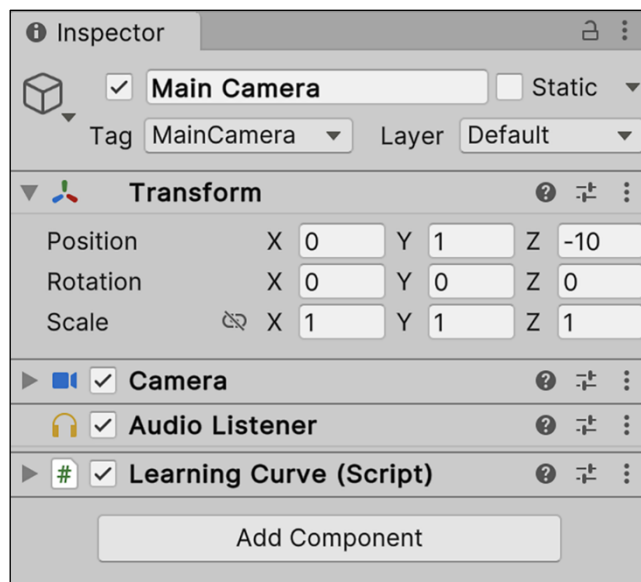
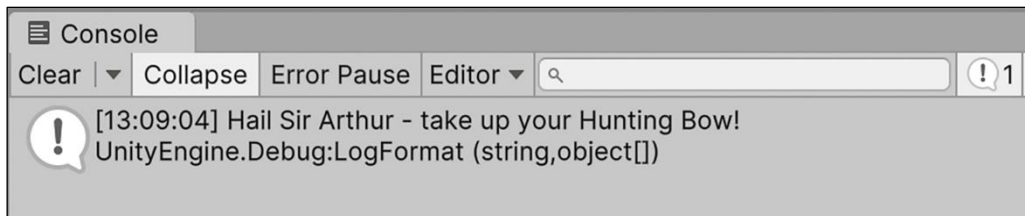
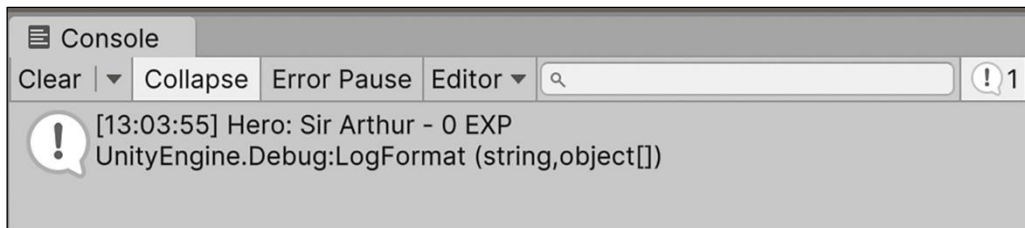




Chapter 5: Working with Classes, Structs, and OOP







Inspector

Directional Light

Static

Tag

Untagged

Layer

Default

Dice	7
Player Lives	3
Cam Transform	<div><div></div>None (Transform)<div></div></div>
Direction Light	<div><div></div>Directional Light<div></div></div>
Light Transform	<div><div></div>None (Transform)<div></div></div>

Chapter 6: Getting Your Hands Dirty with Unity

Concept

Game prototype focused on stealthily avoiding enemies and collecting health items - with a little FPS on the side.

Gameplay

Main mechanic centers around using line-of-sight to stay one step ahead of patrolling enemies and collecting required items.

Combat will consist of shooting projectiles at enemies, which will automatically trigger an attack response.

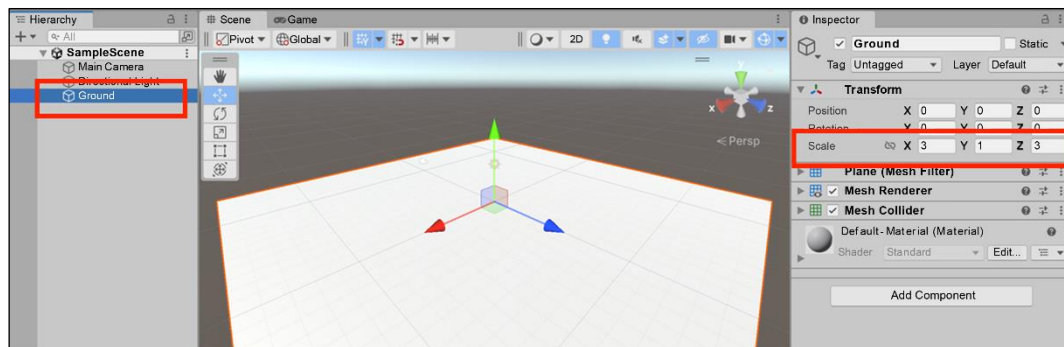
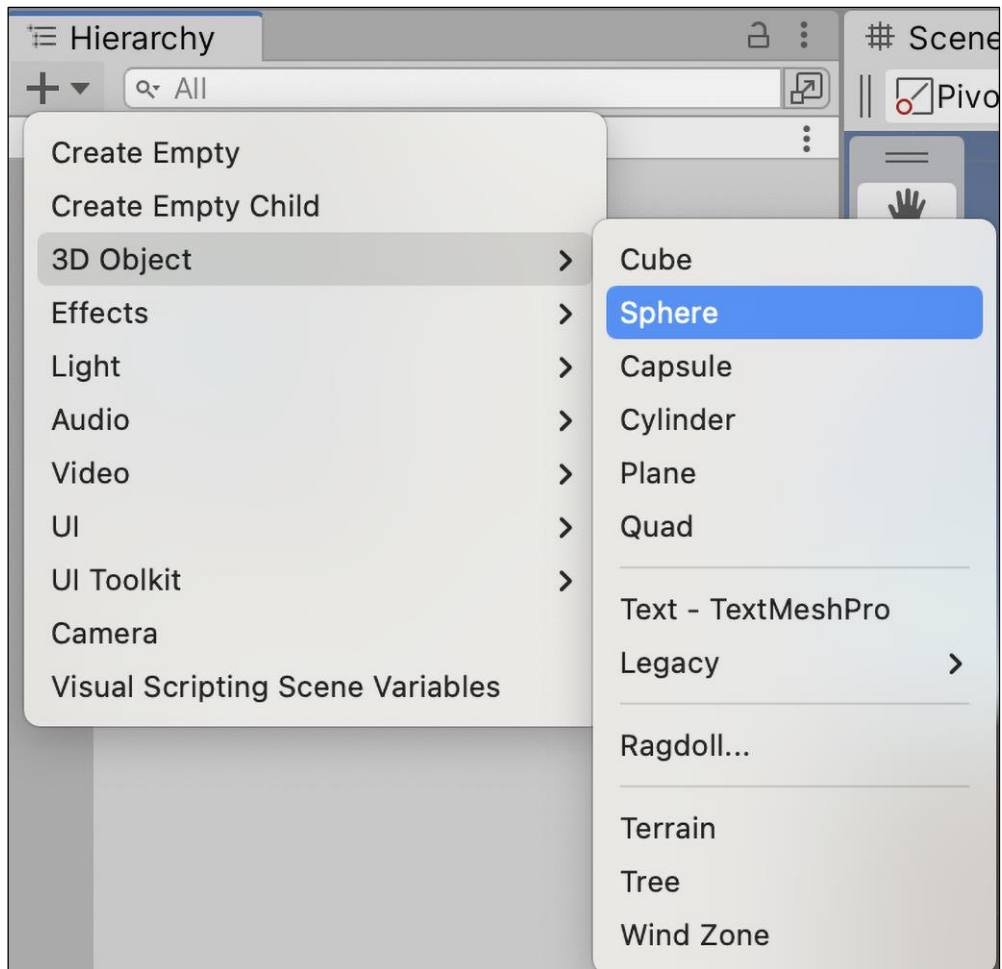
Interface

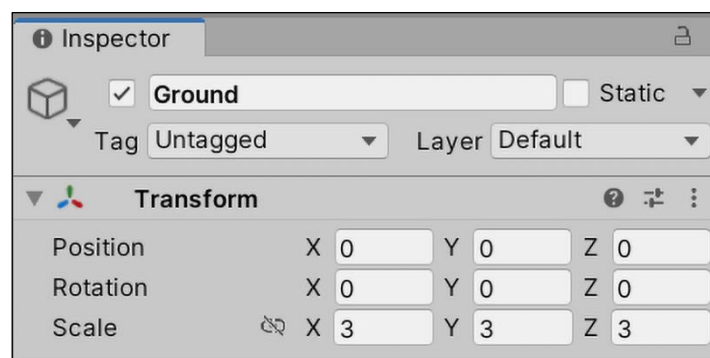
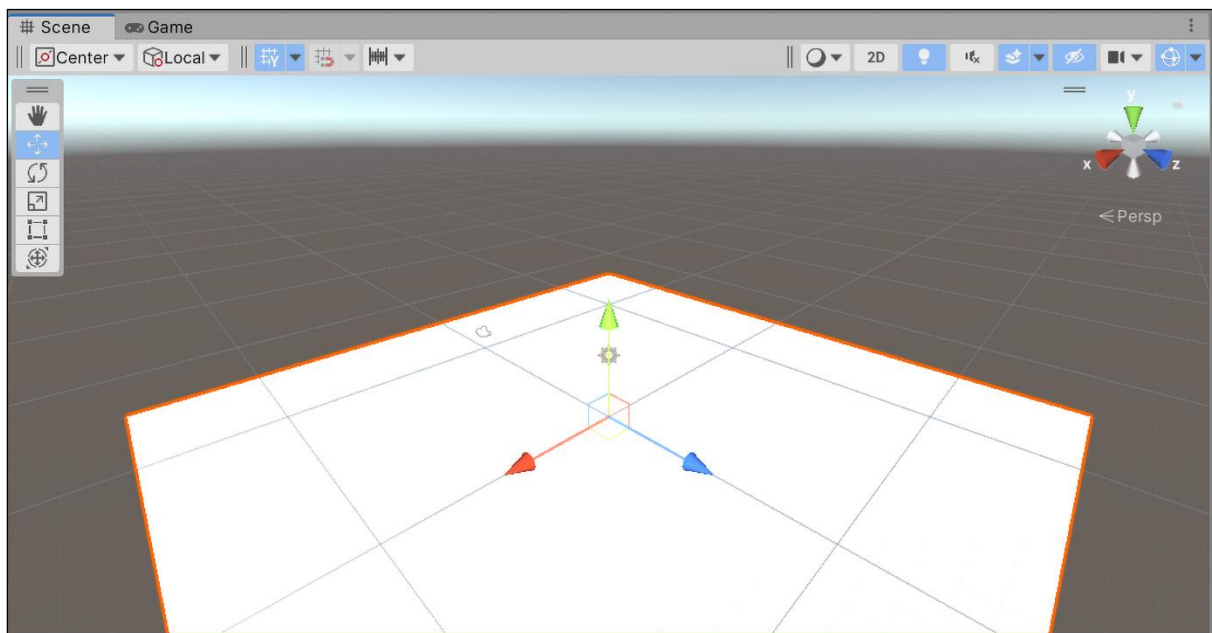
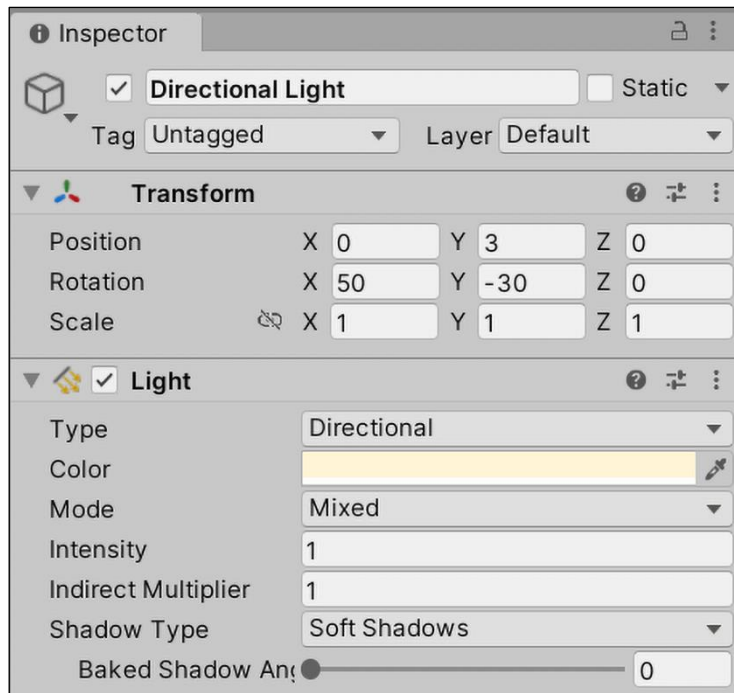
Control scheme for movement will be the WASD or arrow keys using the mouse for camera control. Shooting mechanic will use the Space bar, and item collection will work off of object collisions.

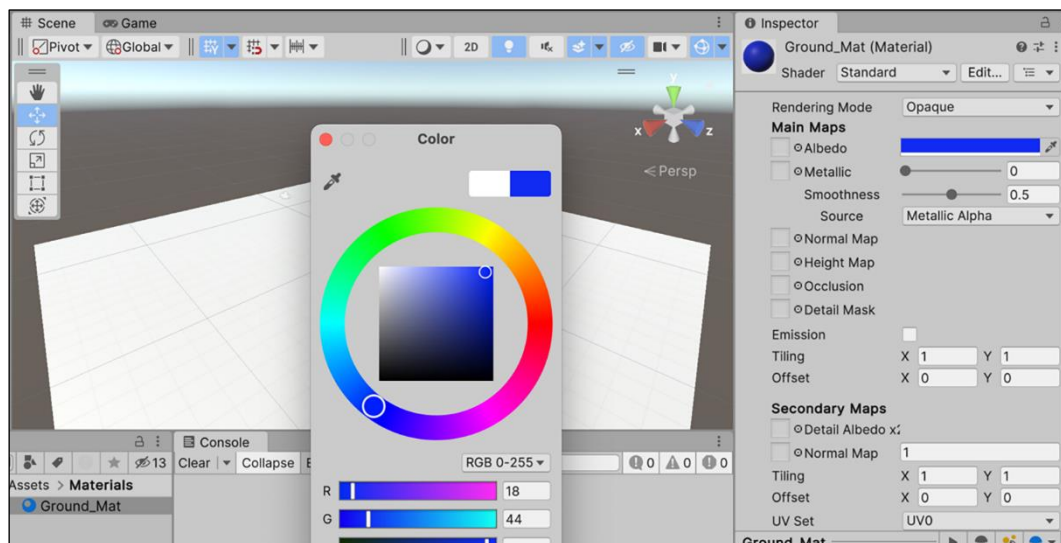
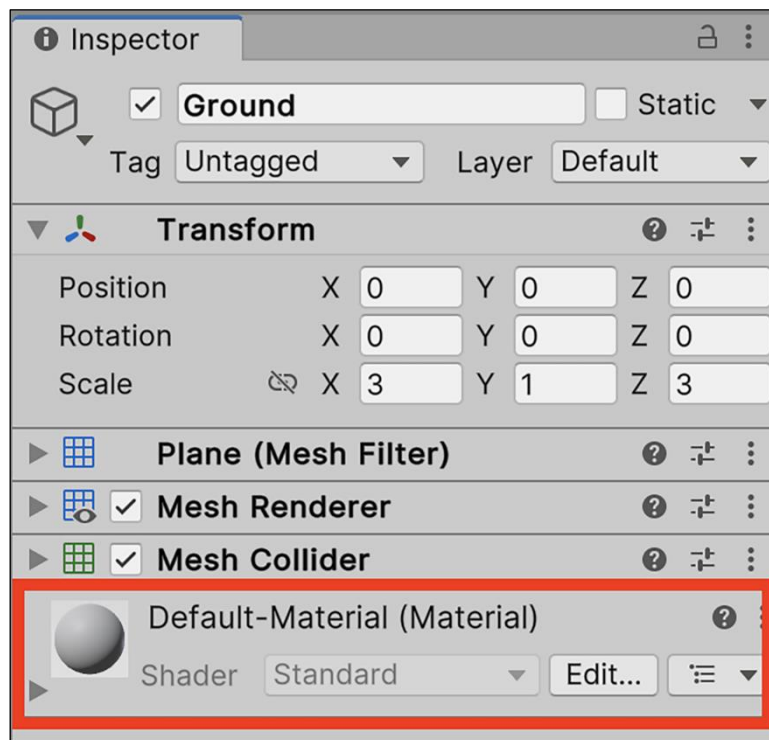
Simple HUD will show items collected and remaining ammo, as well as a standard health bar.

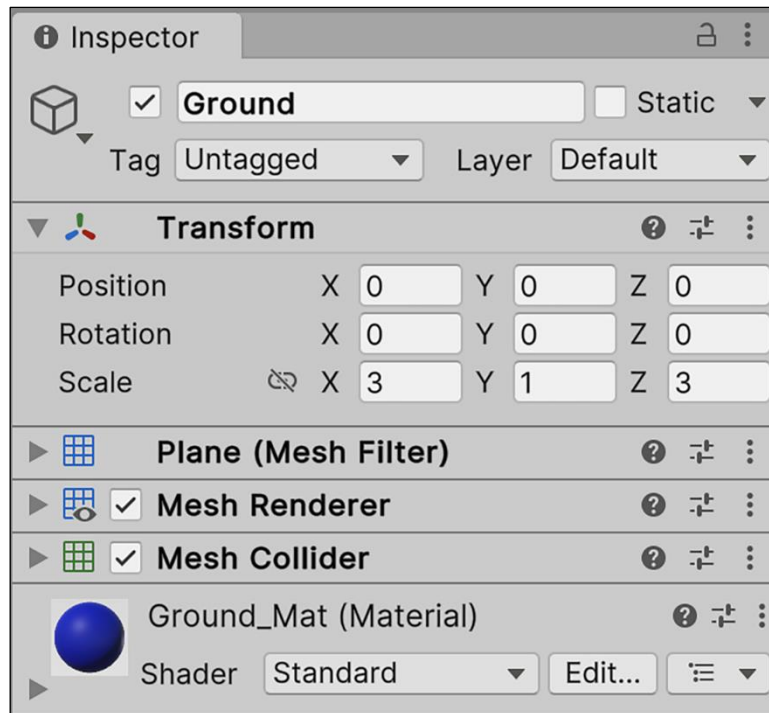
Art Style

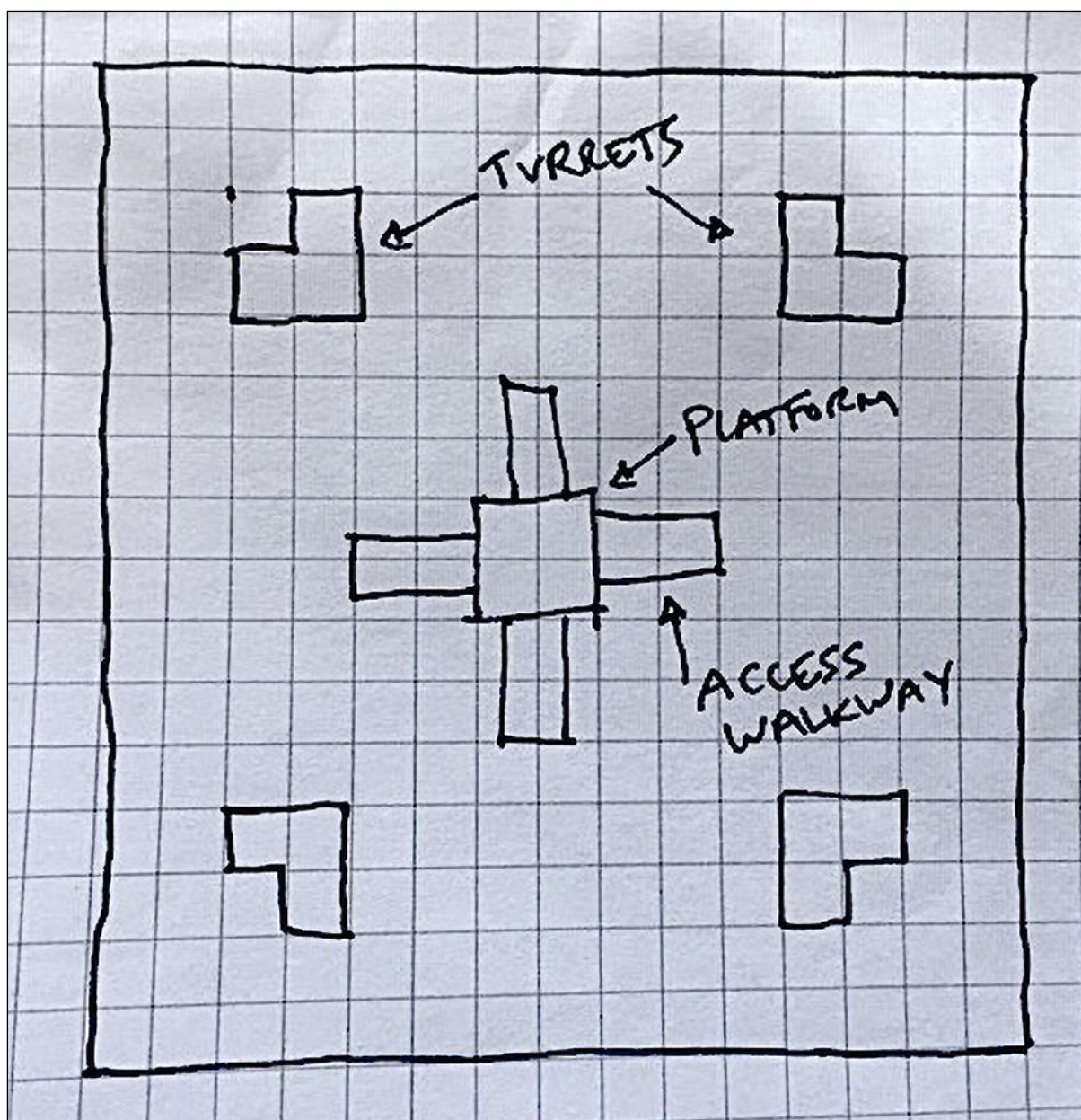
Level and character art style will be all primitive GameObjects for fast and efficient, no-frills development. These can be swapped out at a later date with 3D models or terrain environments if needed.

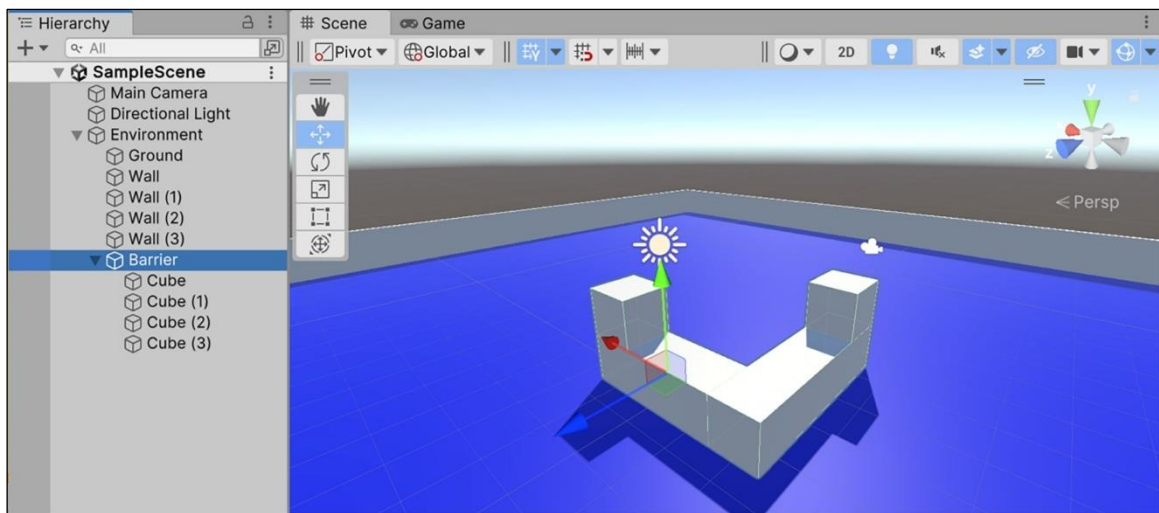
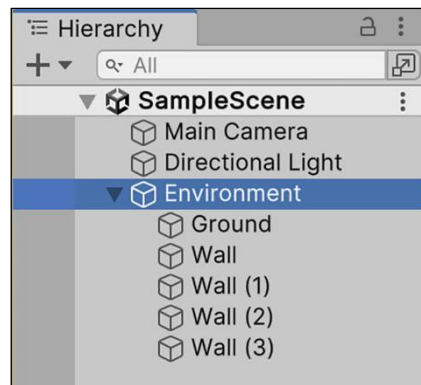
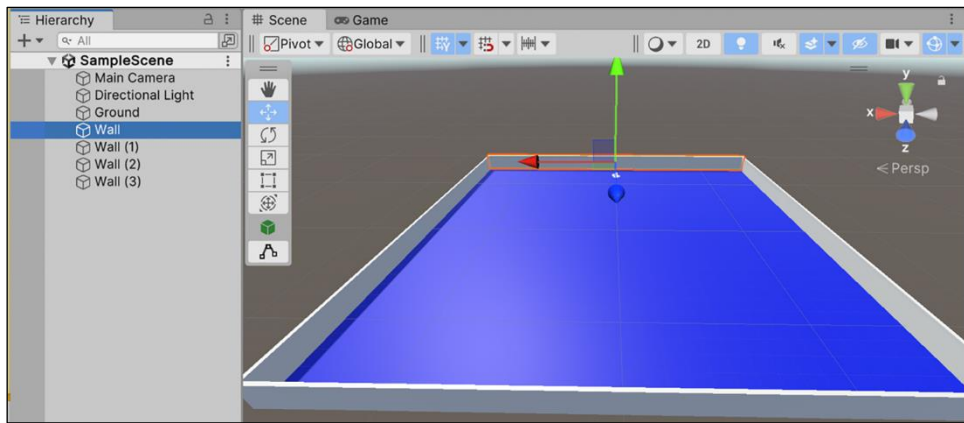


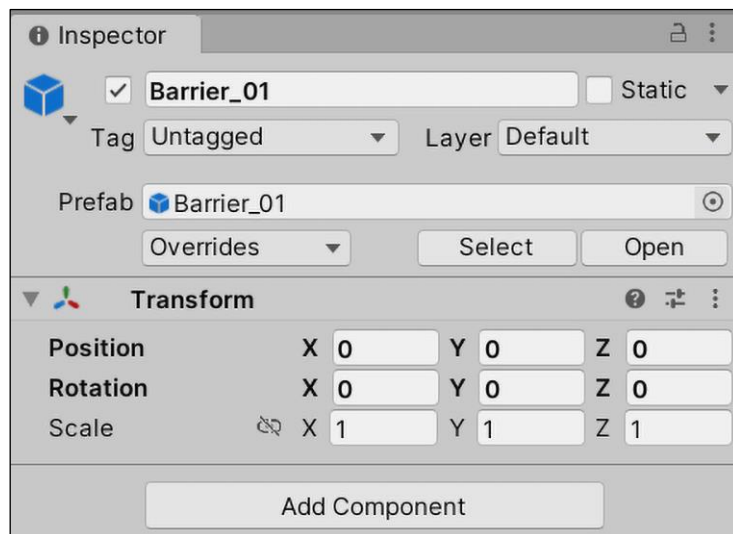
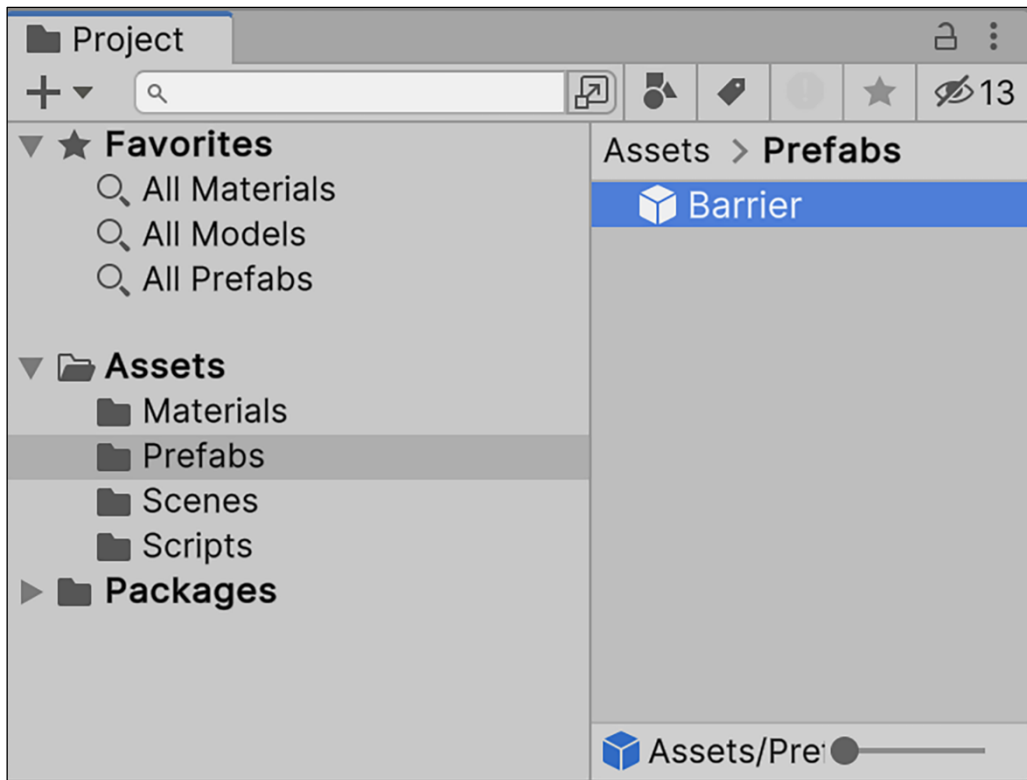


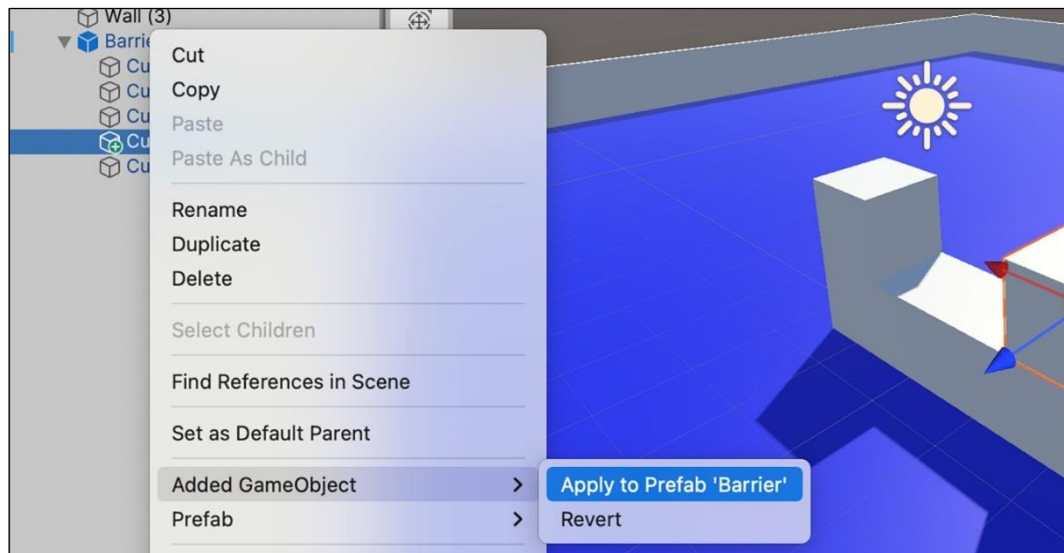
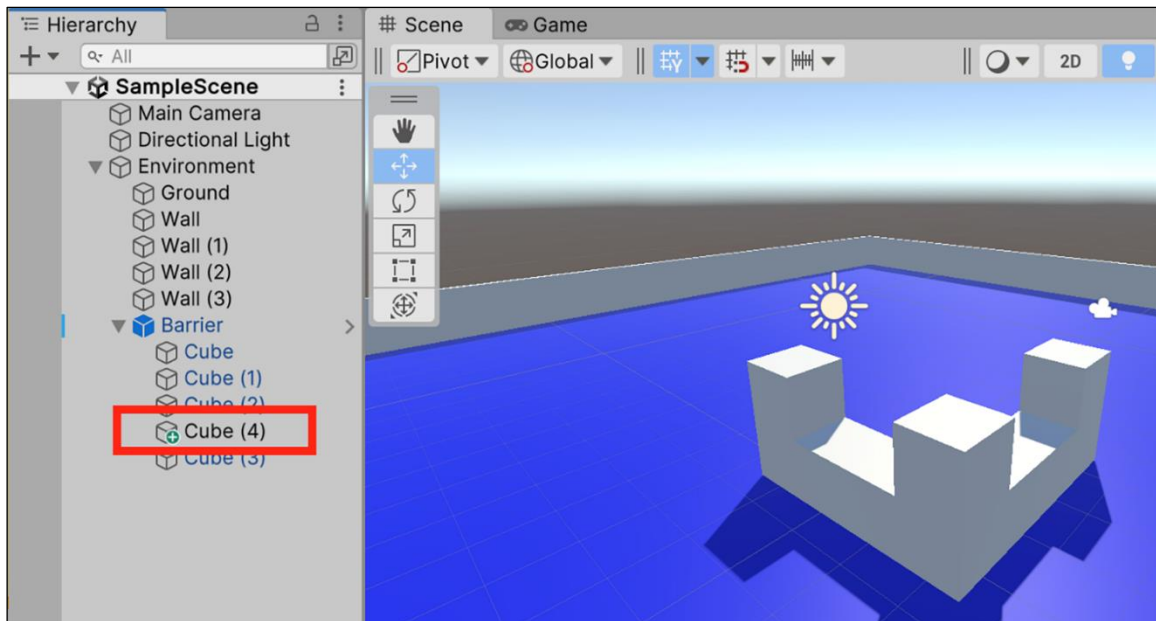


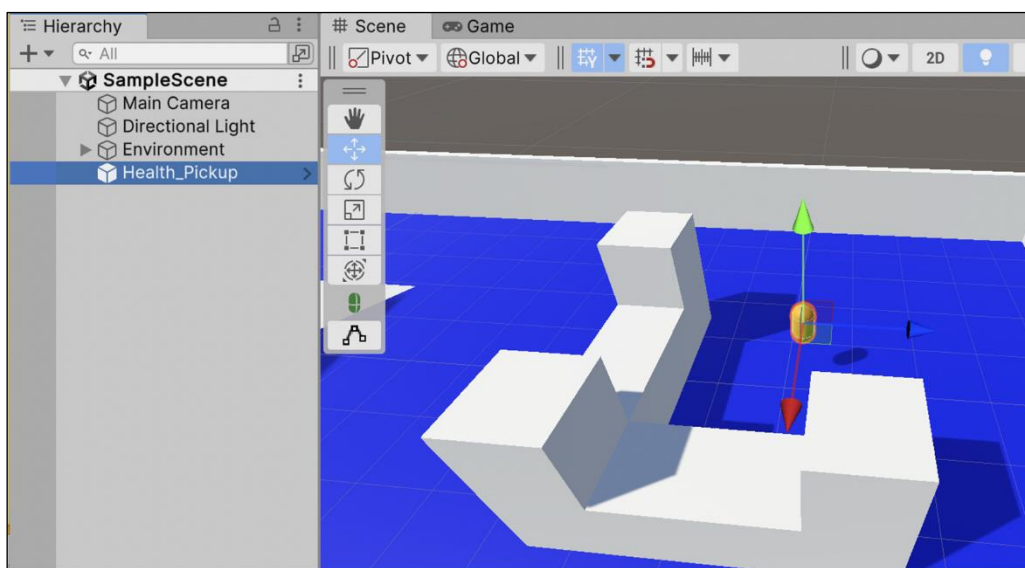
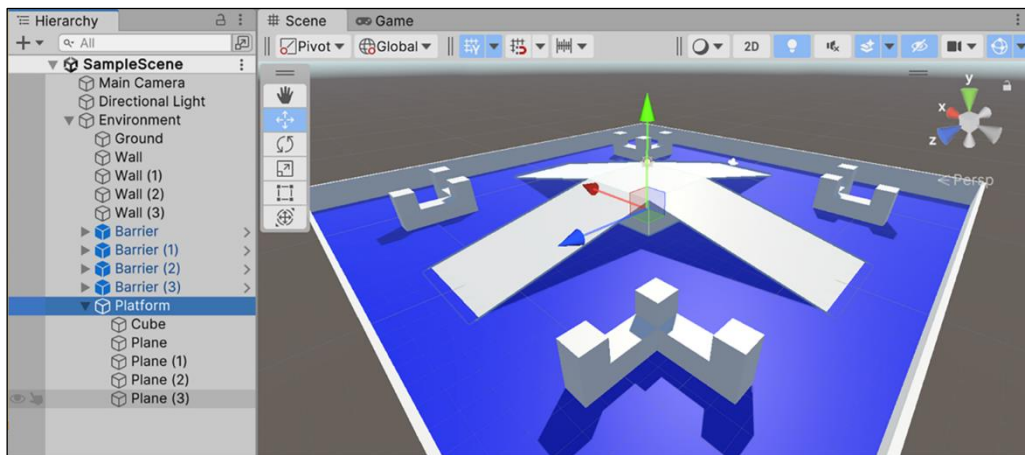
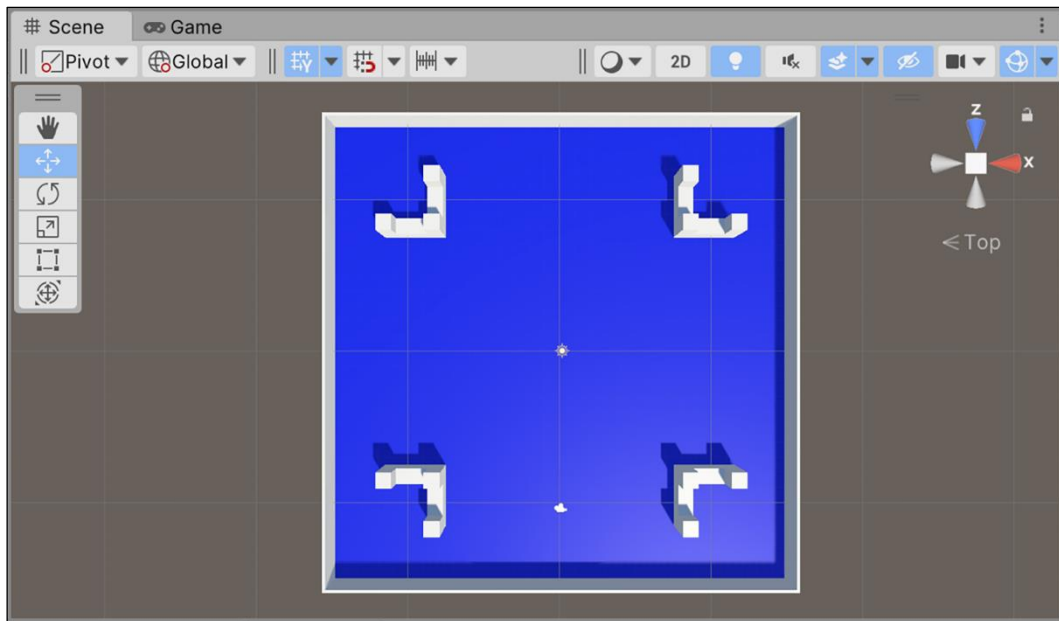


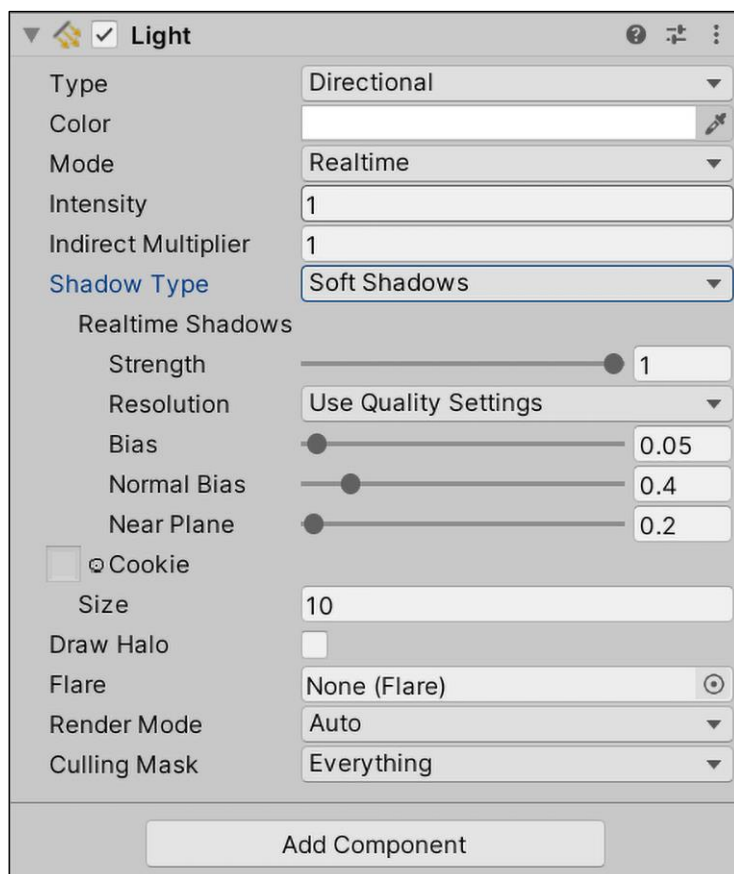
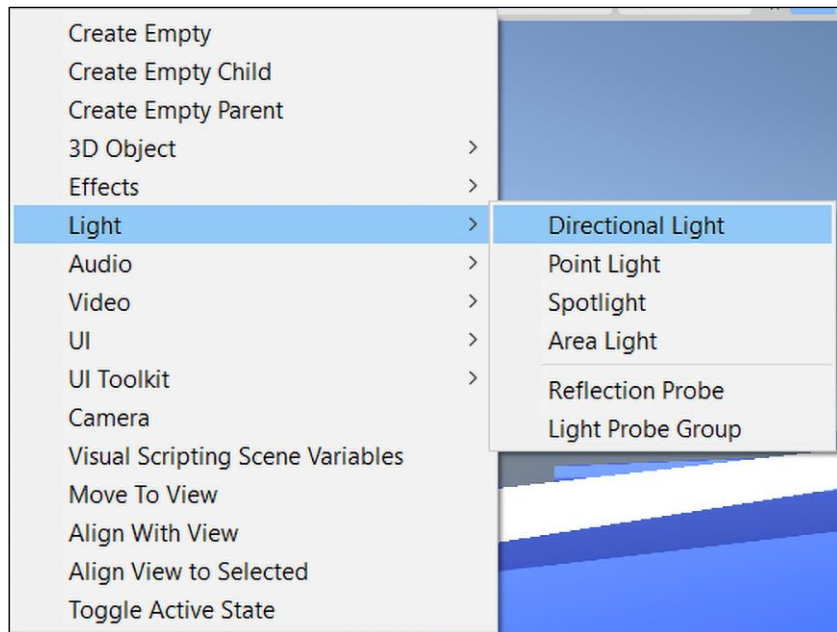


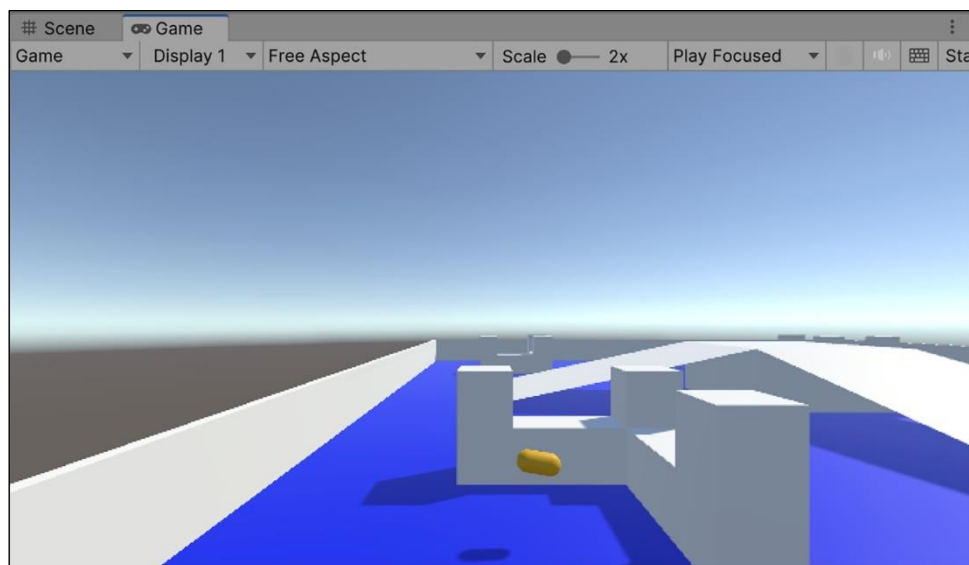
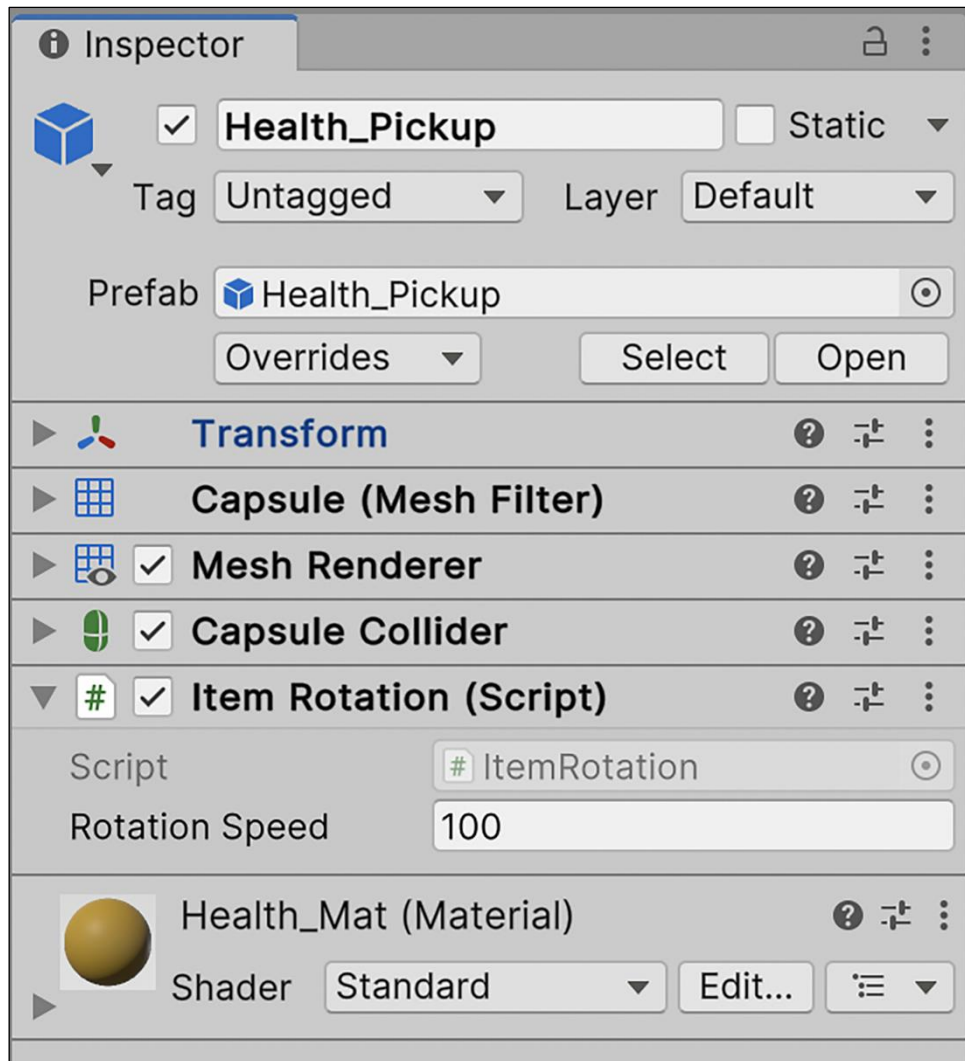


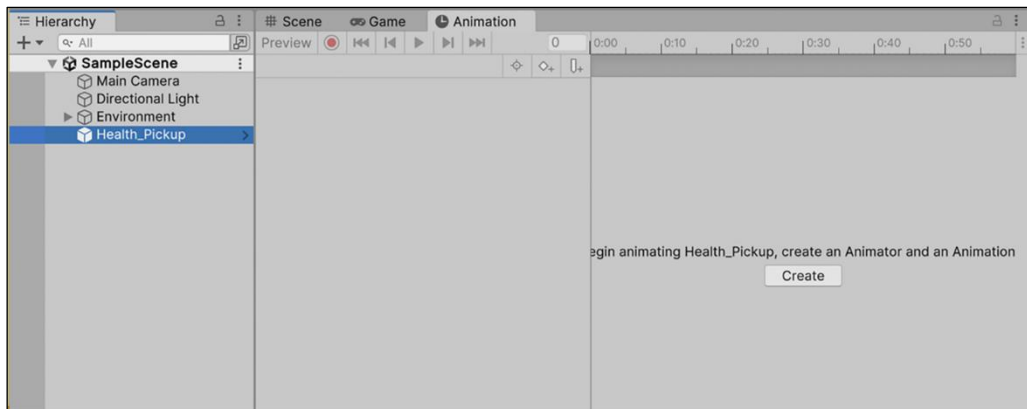
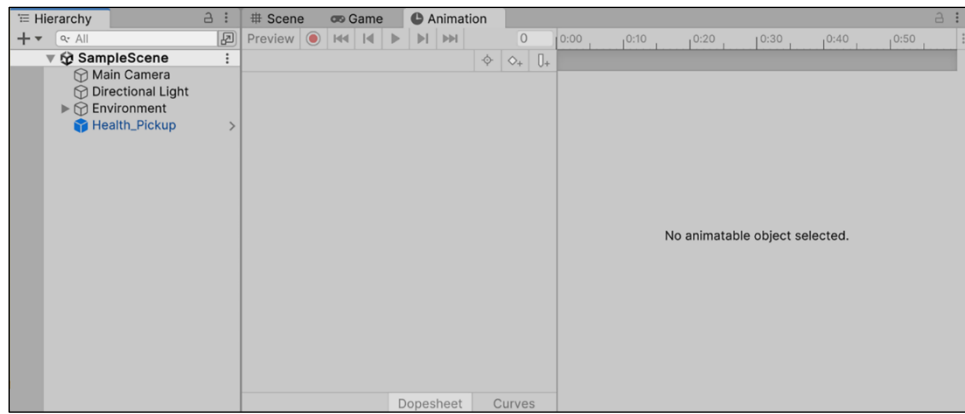




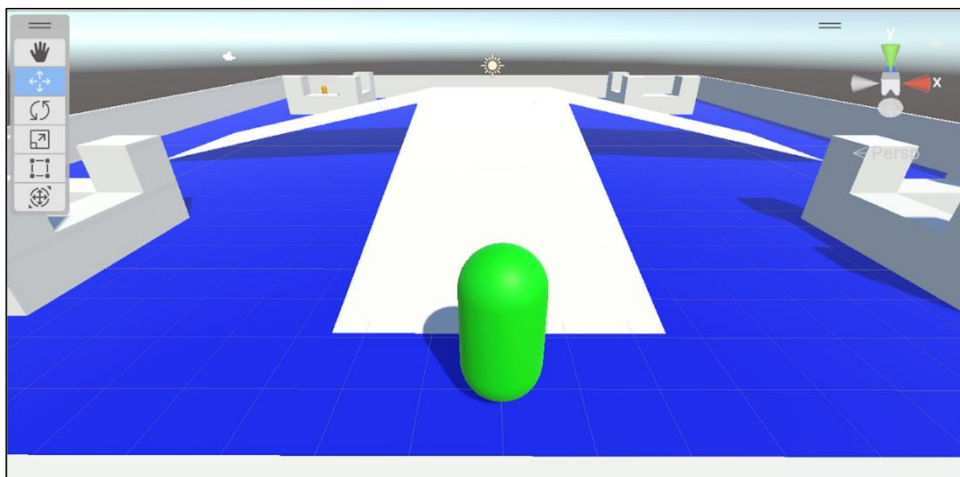
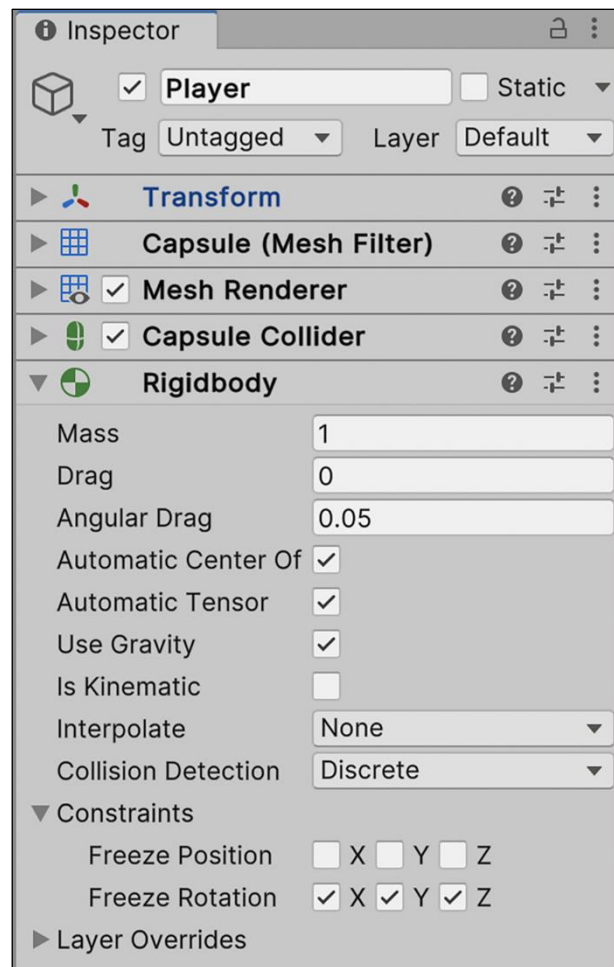


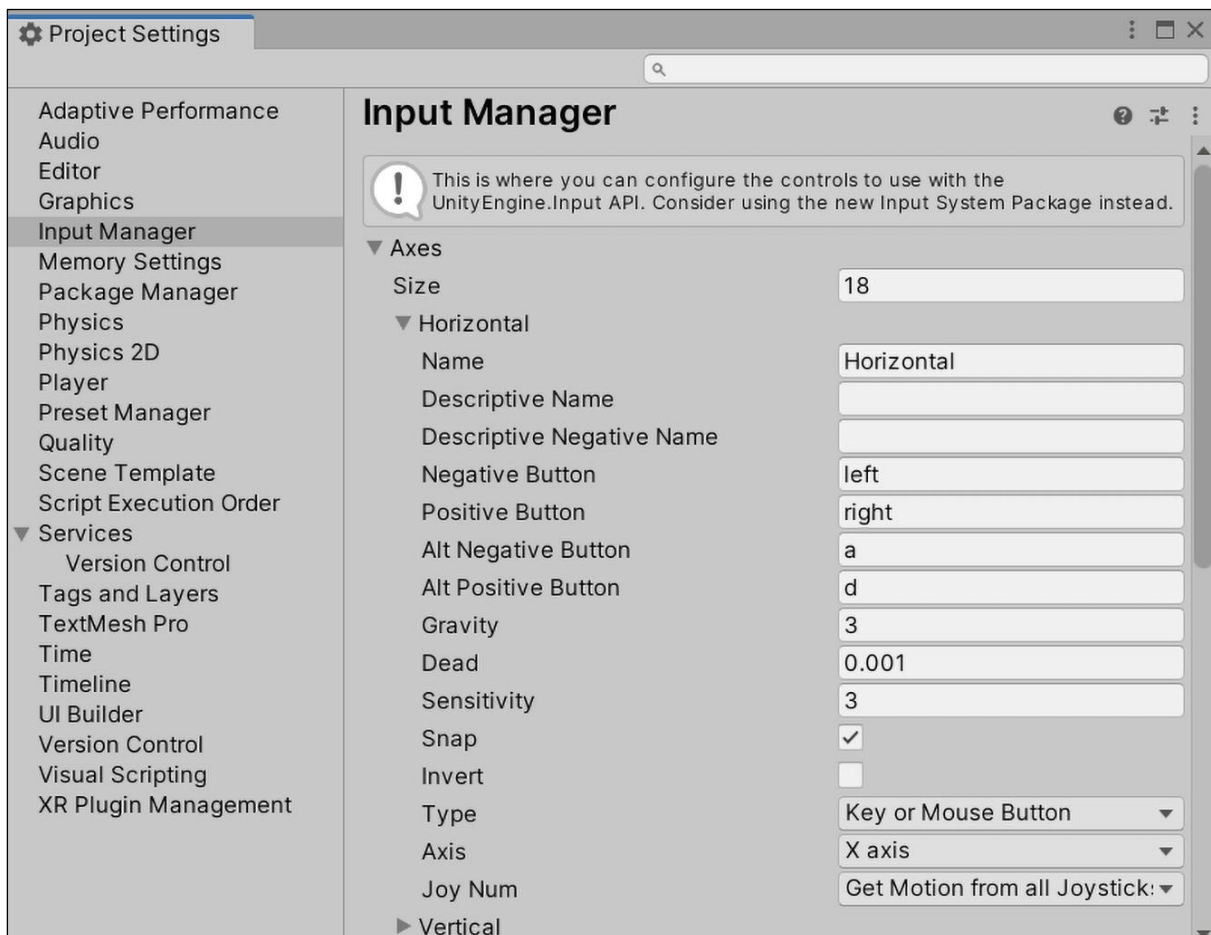
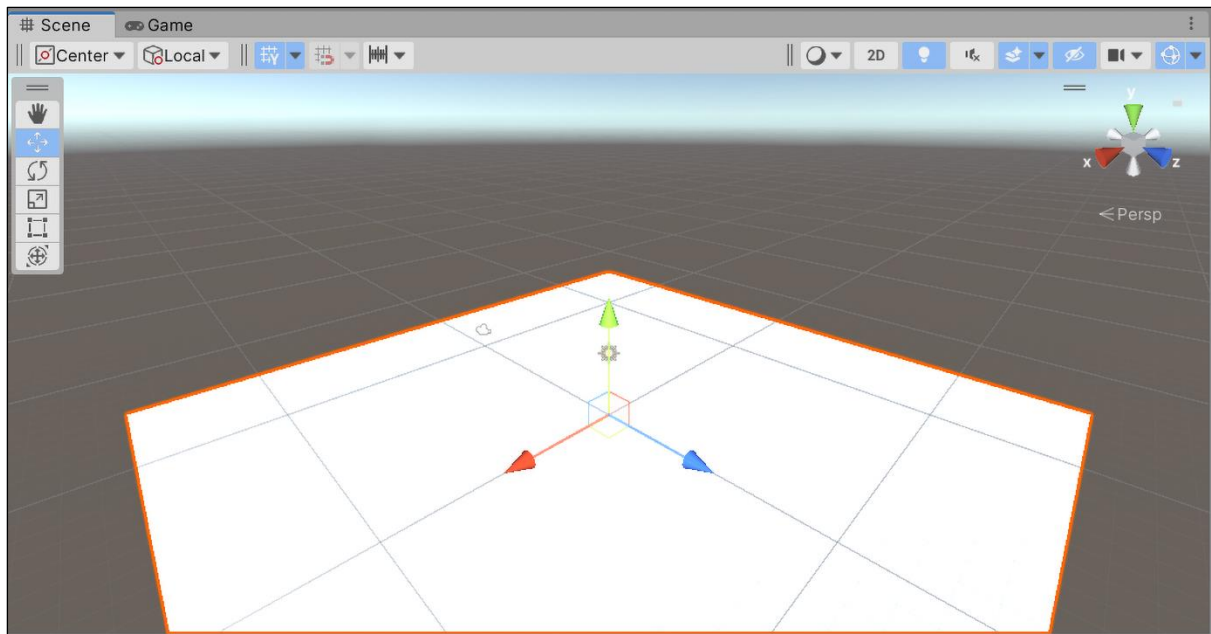


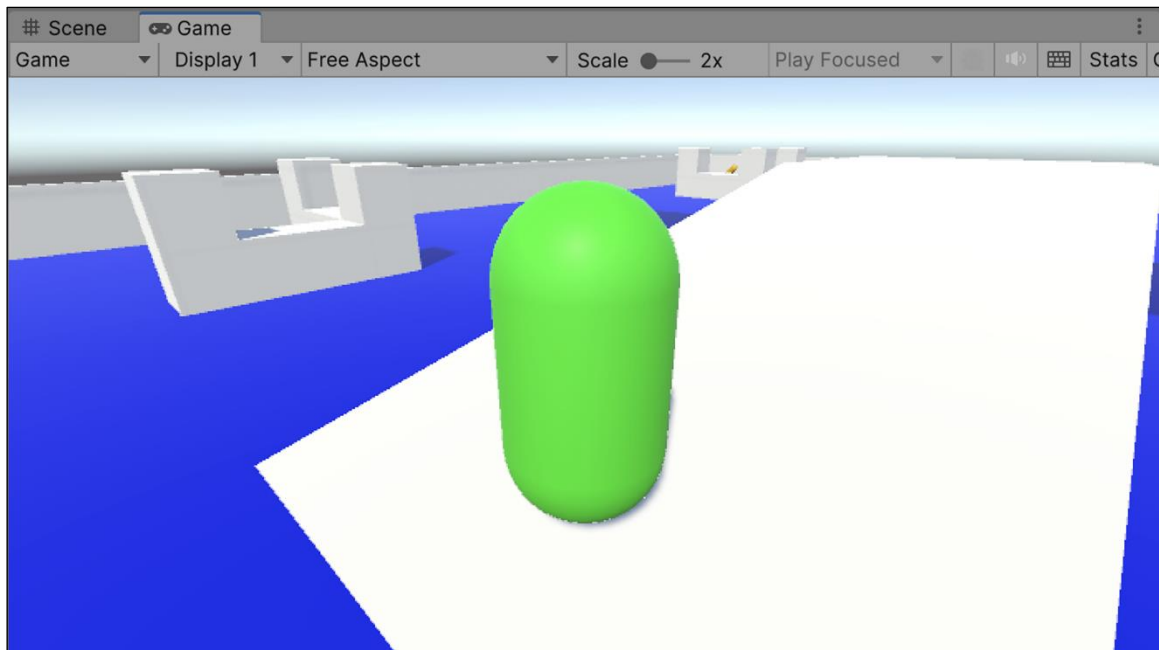




Chapter 7: Movement, Camera Controls, and Collisions







Rigidbody

Mass: 1

Drag: 0

Angular Drag: 0.05

Automatic Center Of: ☒

Automatic Tensor: ☒

Use Gravity: ☒

Is Kinematic: ☐

Interpolate: None

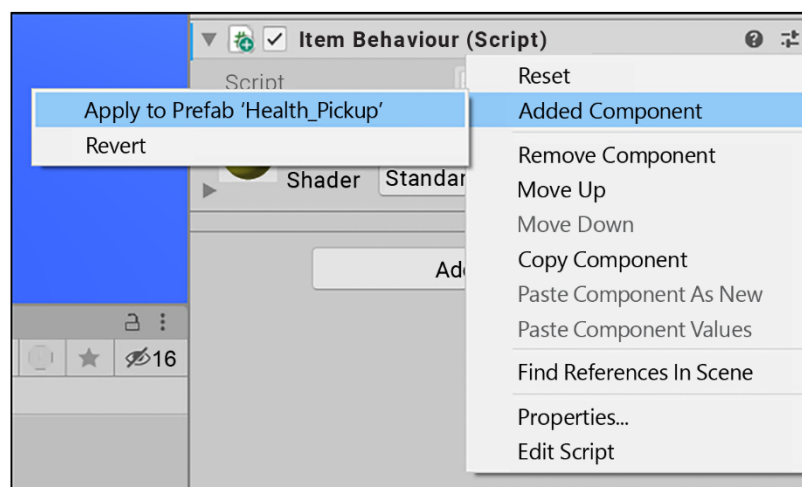
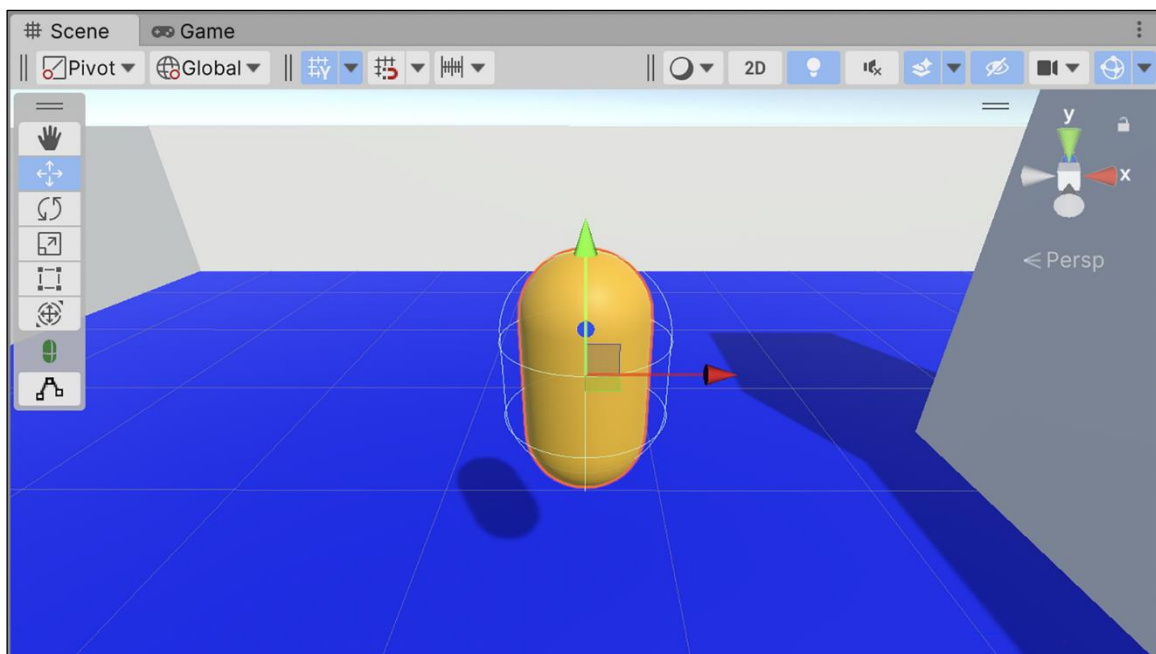
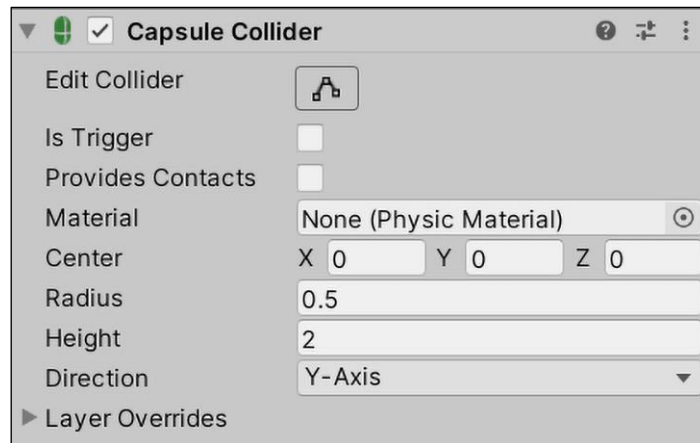
Collision Detection: Discrete

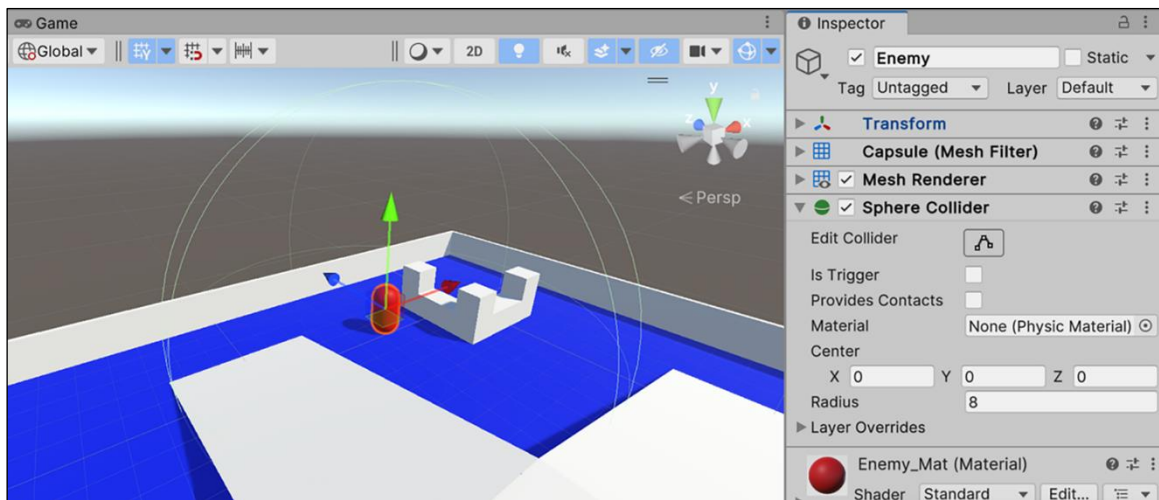
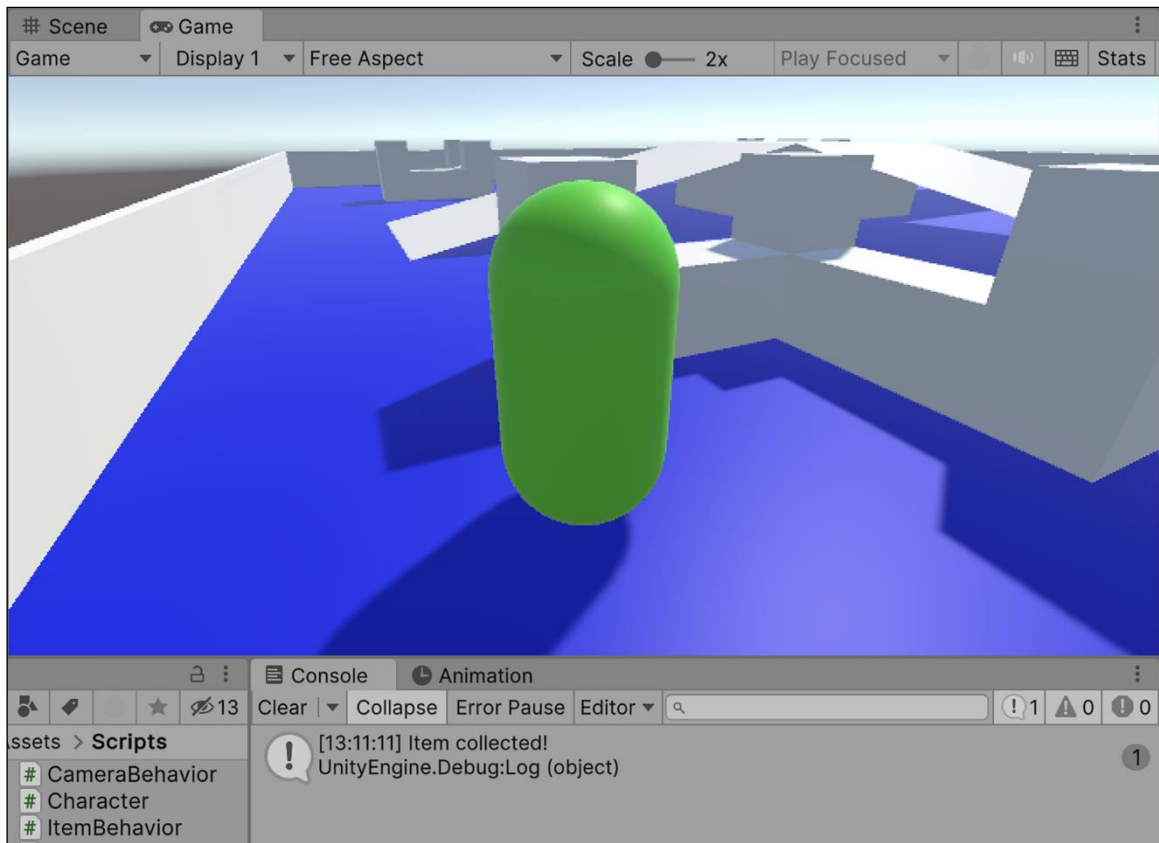
▼ Constraints

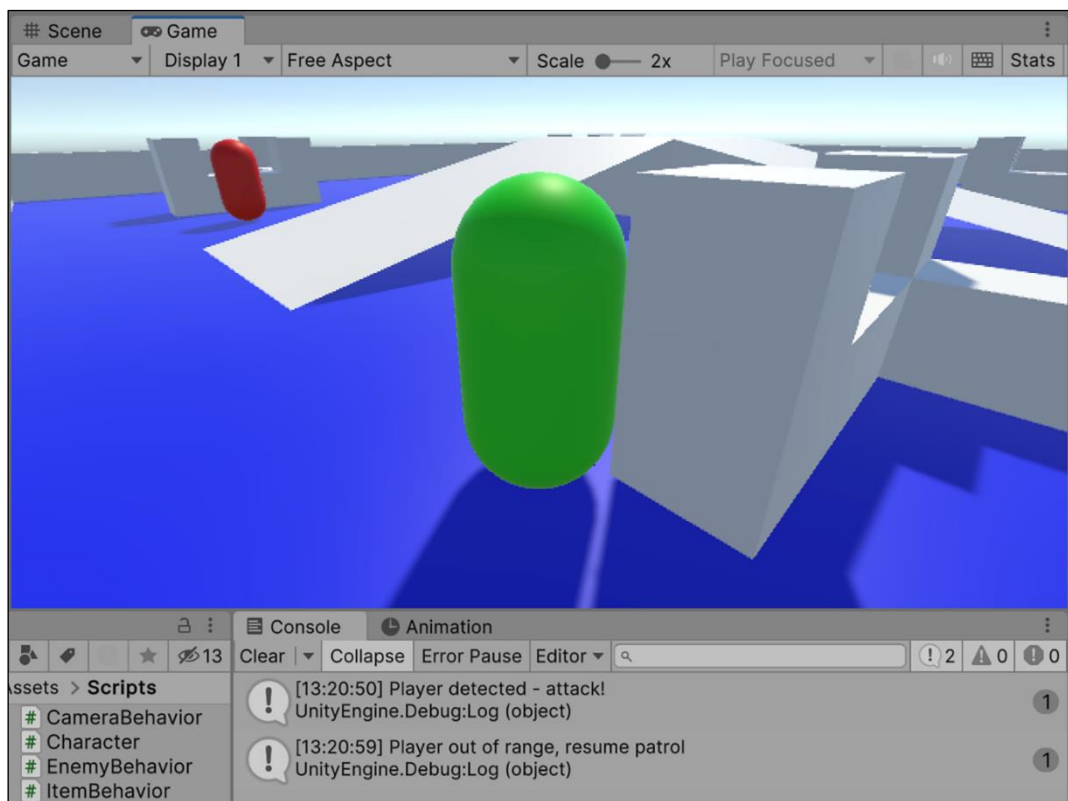
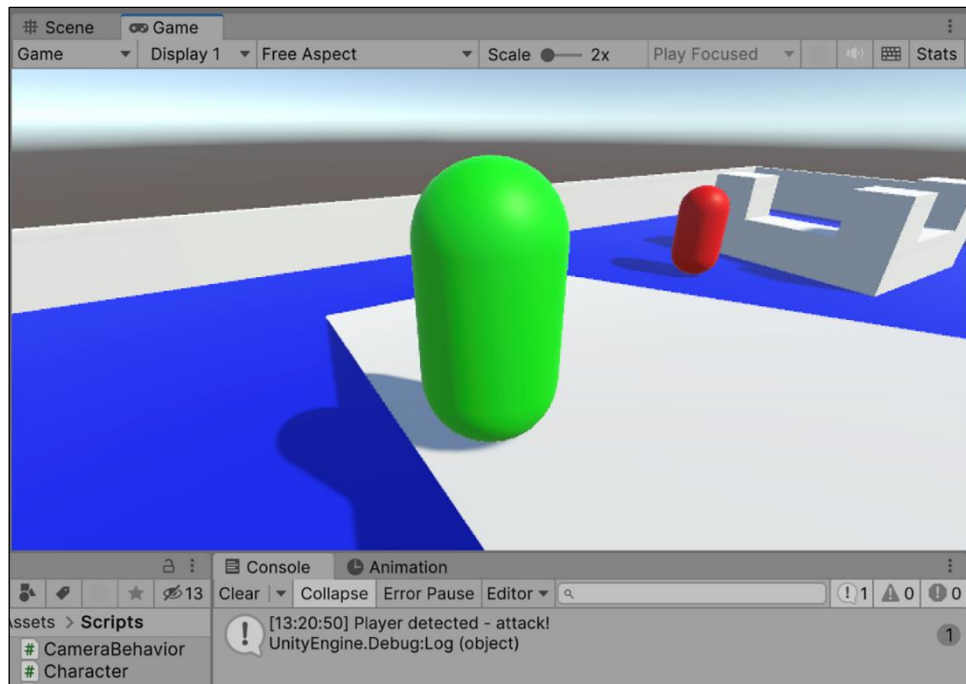
Freeze Position: ☐ X ☐ Y ☐ Z

Freeze Rotation: ☒ X ☒ Y ☒ Z

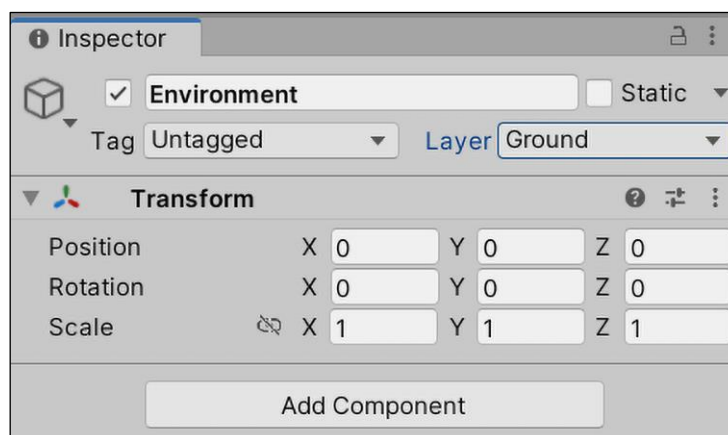
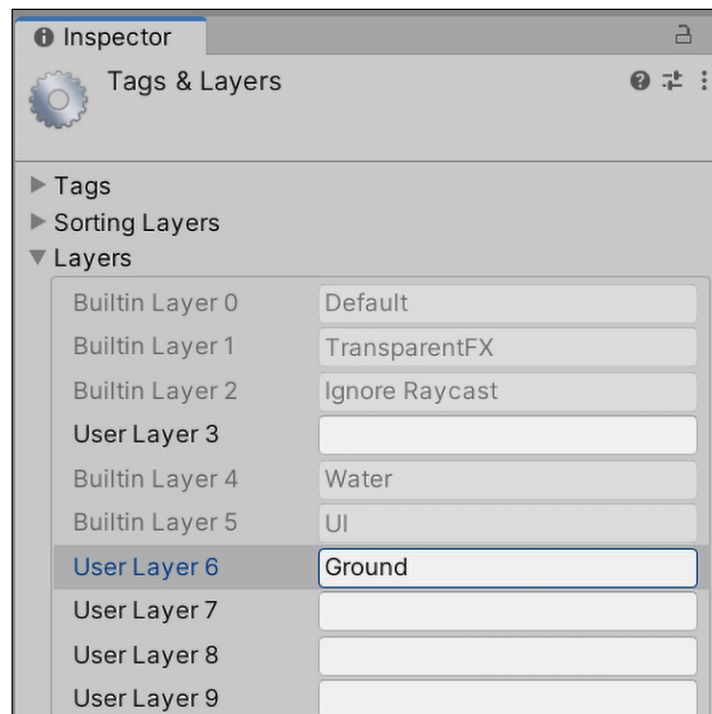
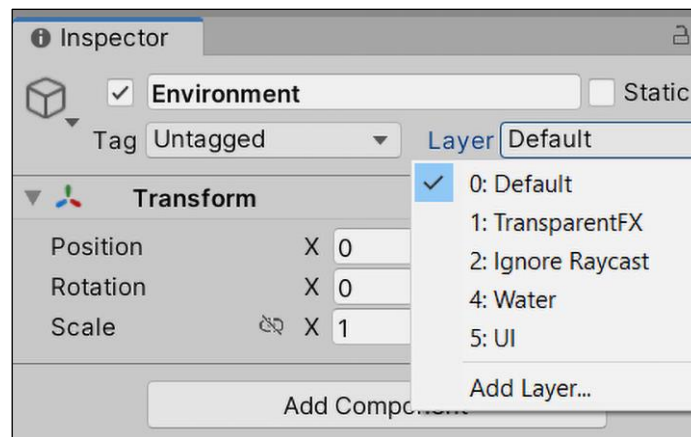
► Layer Overrides

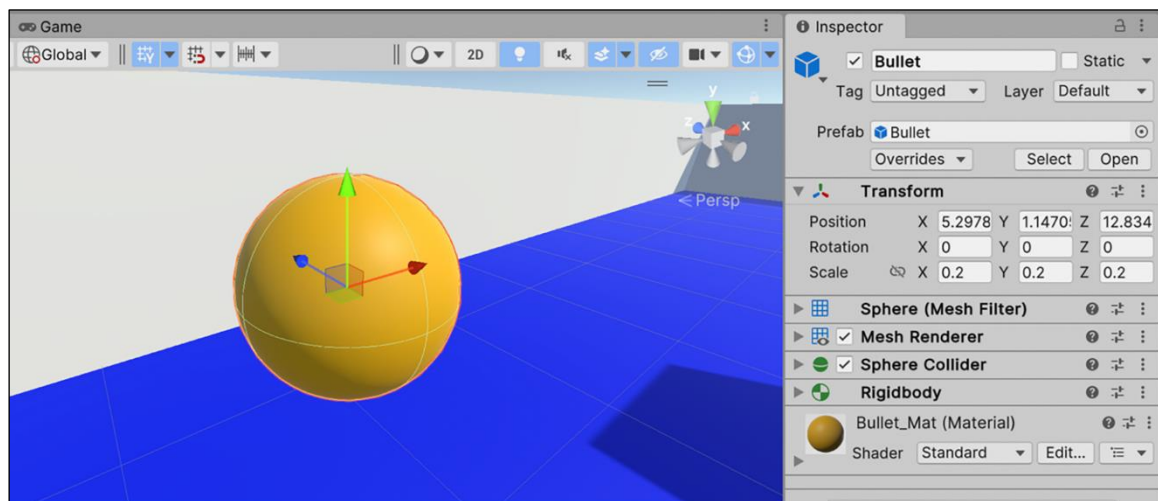
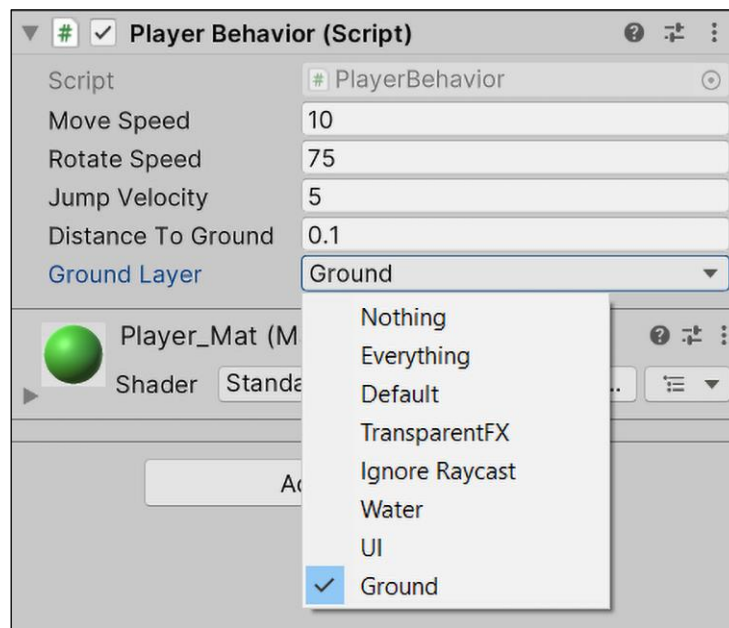


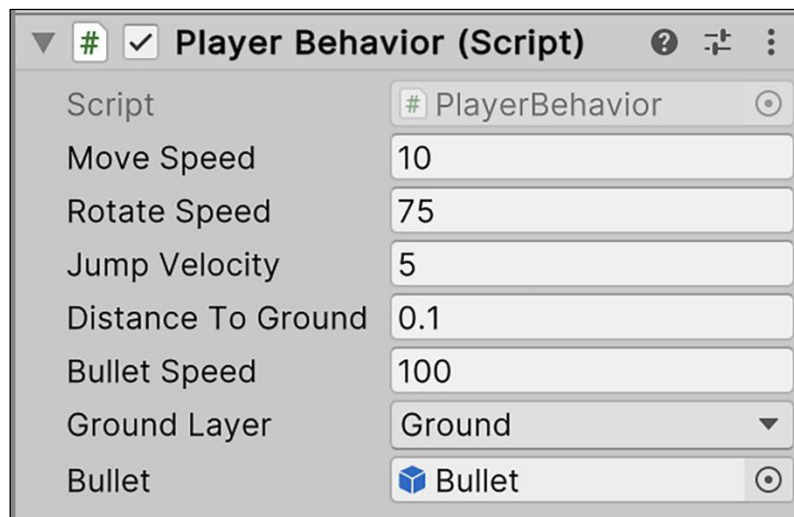
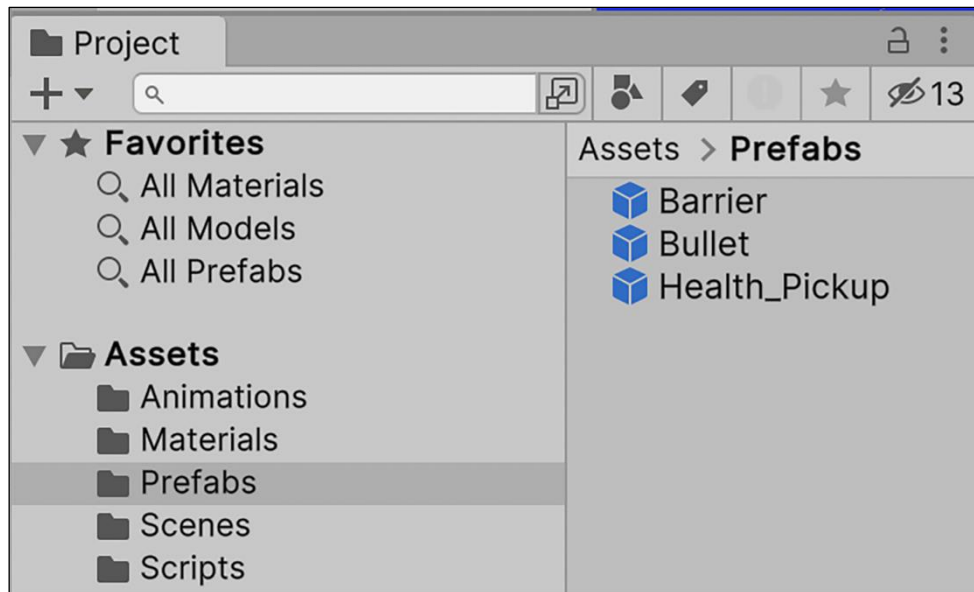


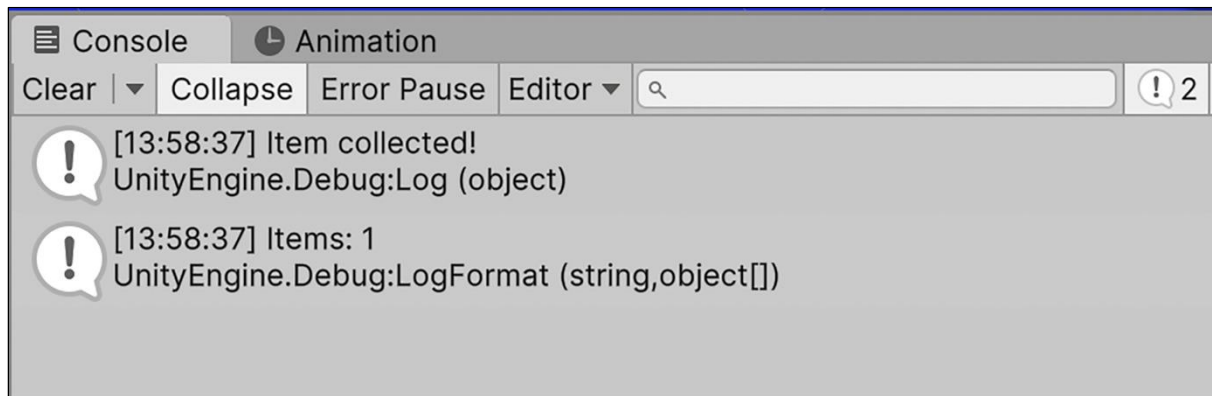
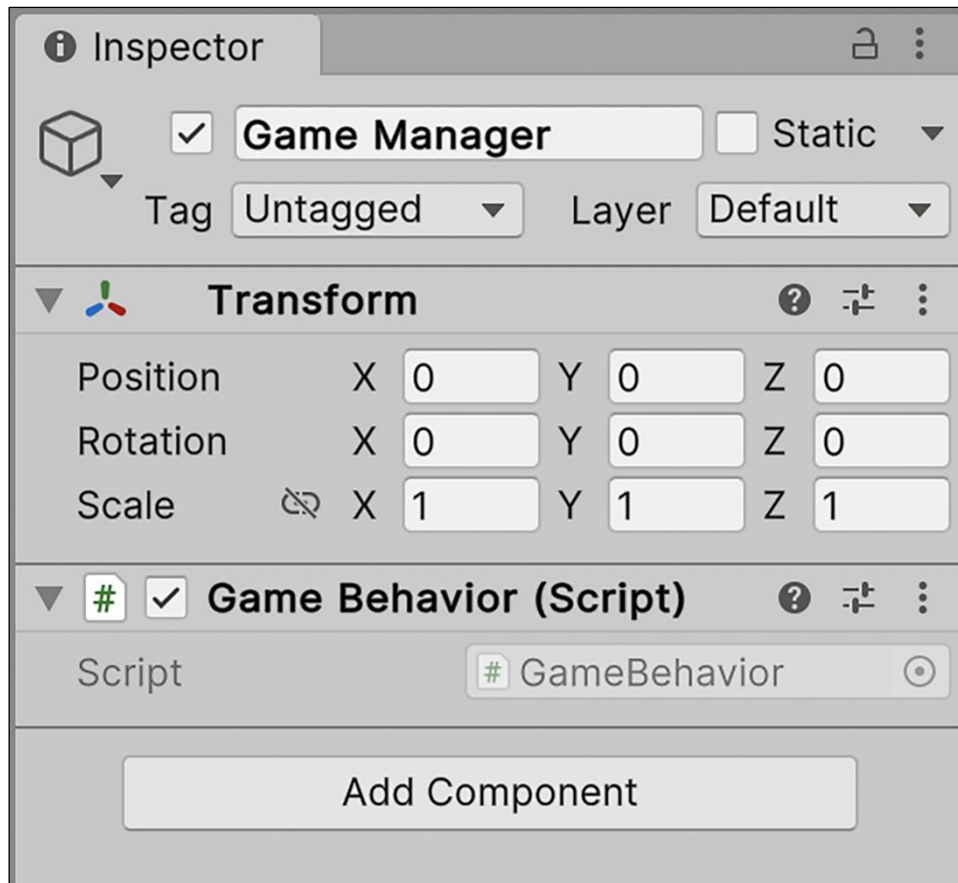


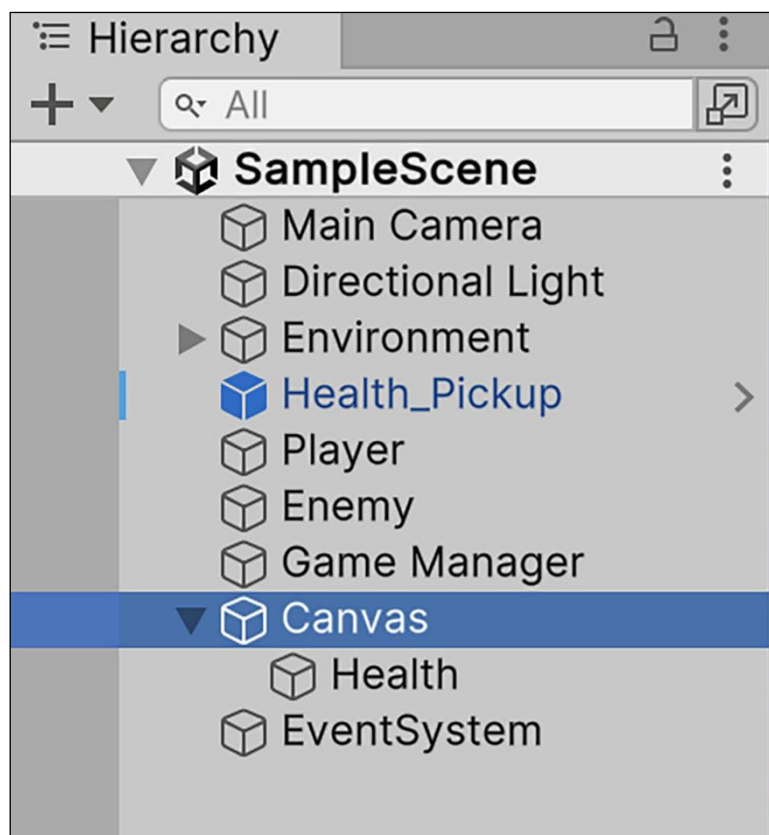
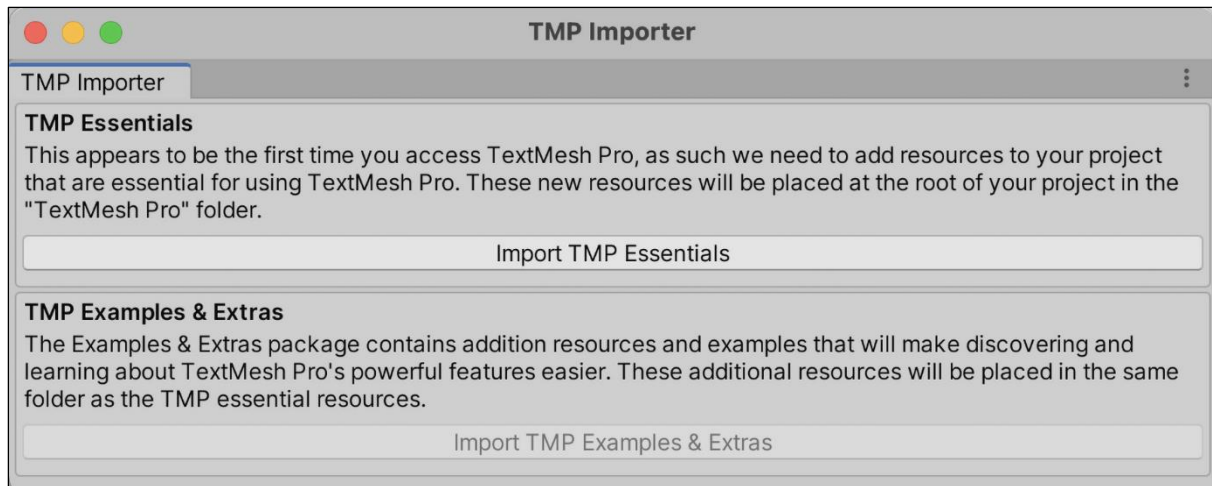
Chapter 8: Scripting Game Mechanics

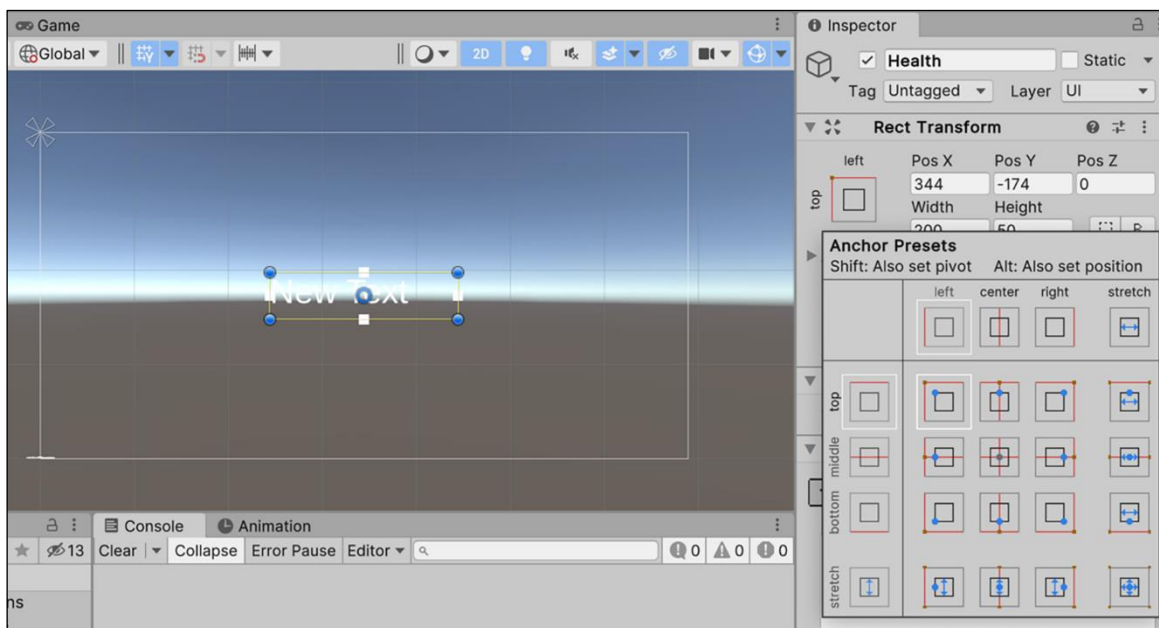
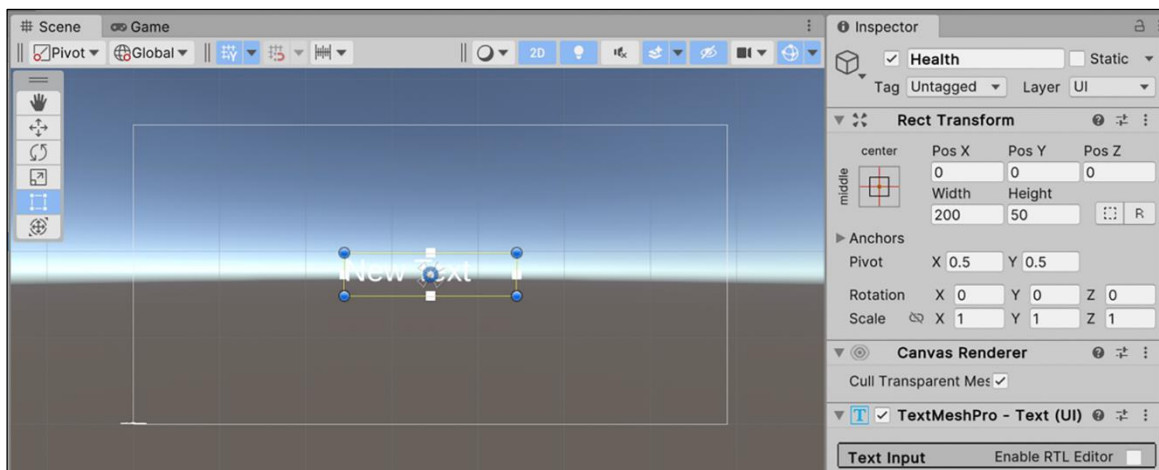
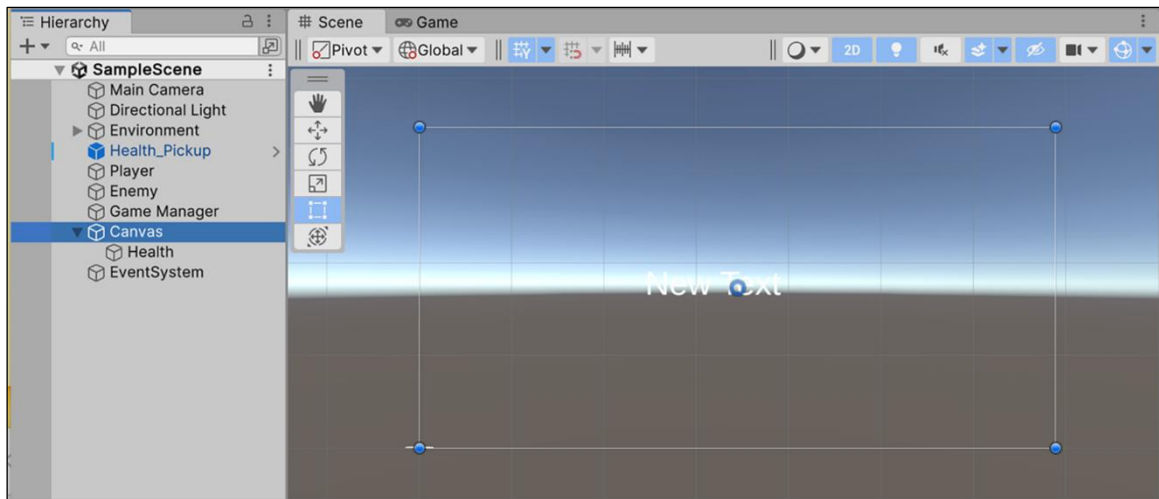


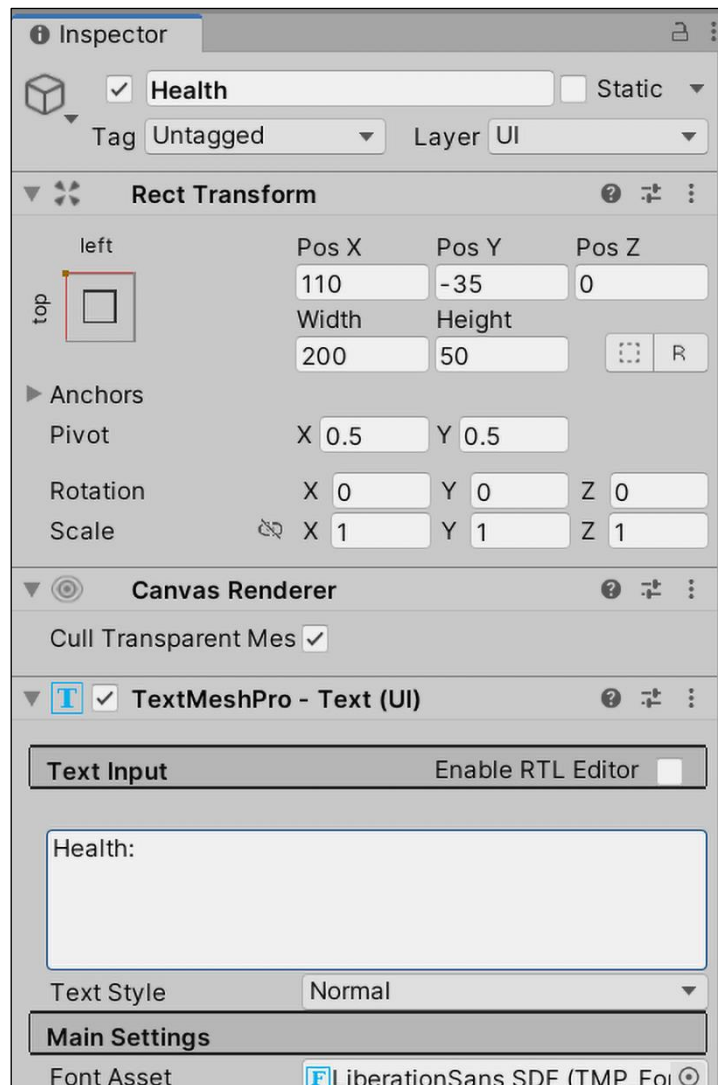
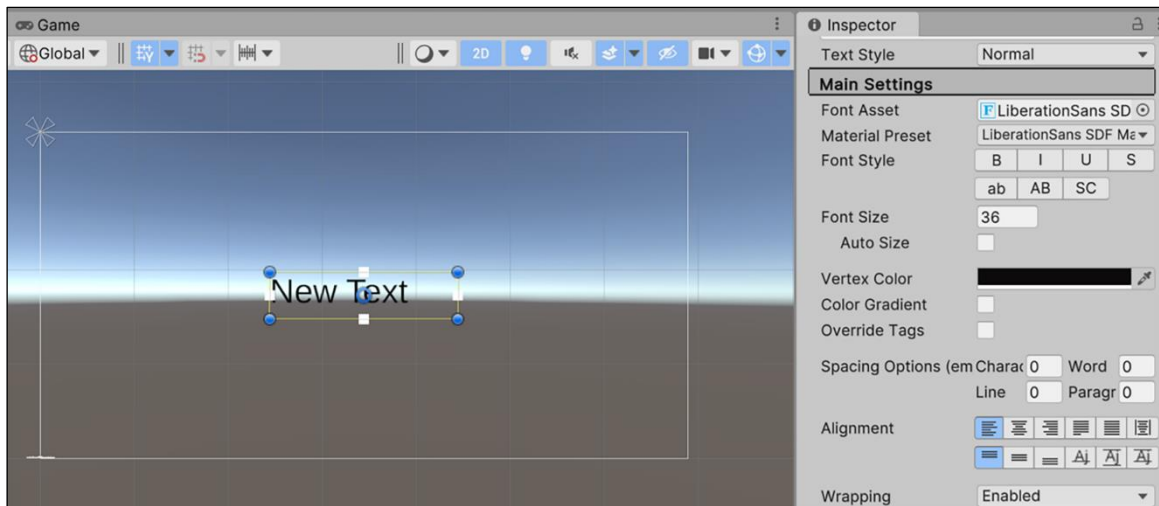


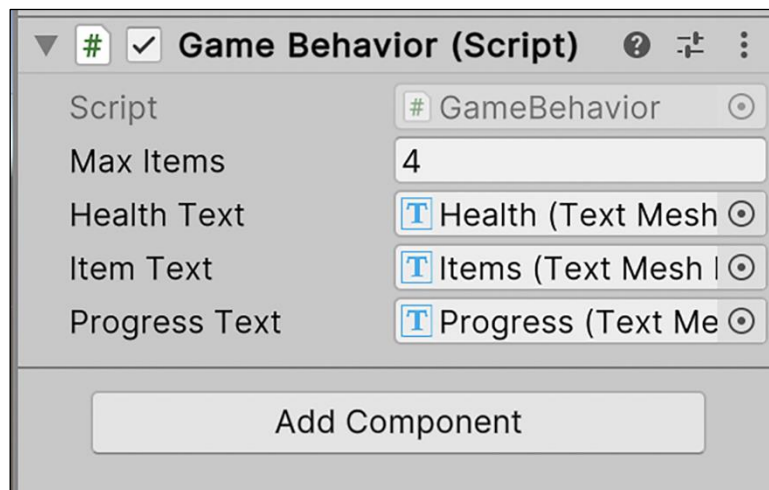
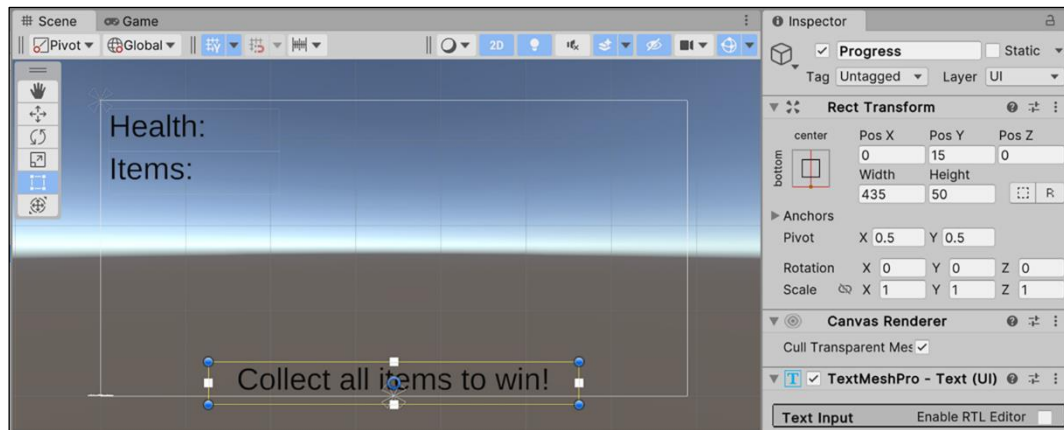
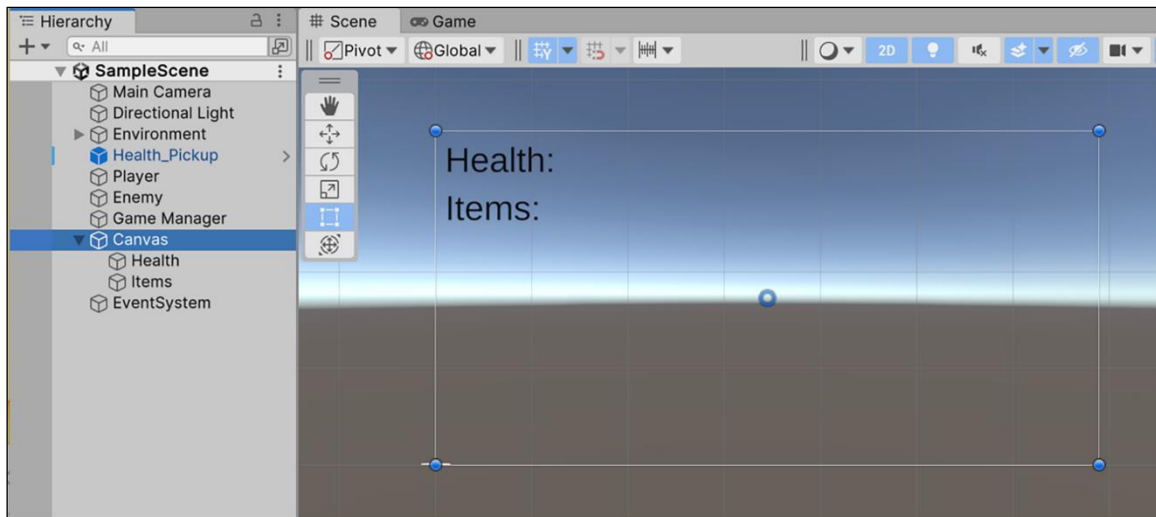


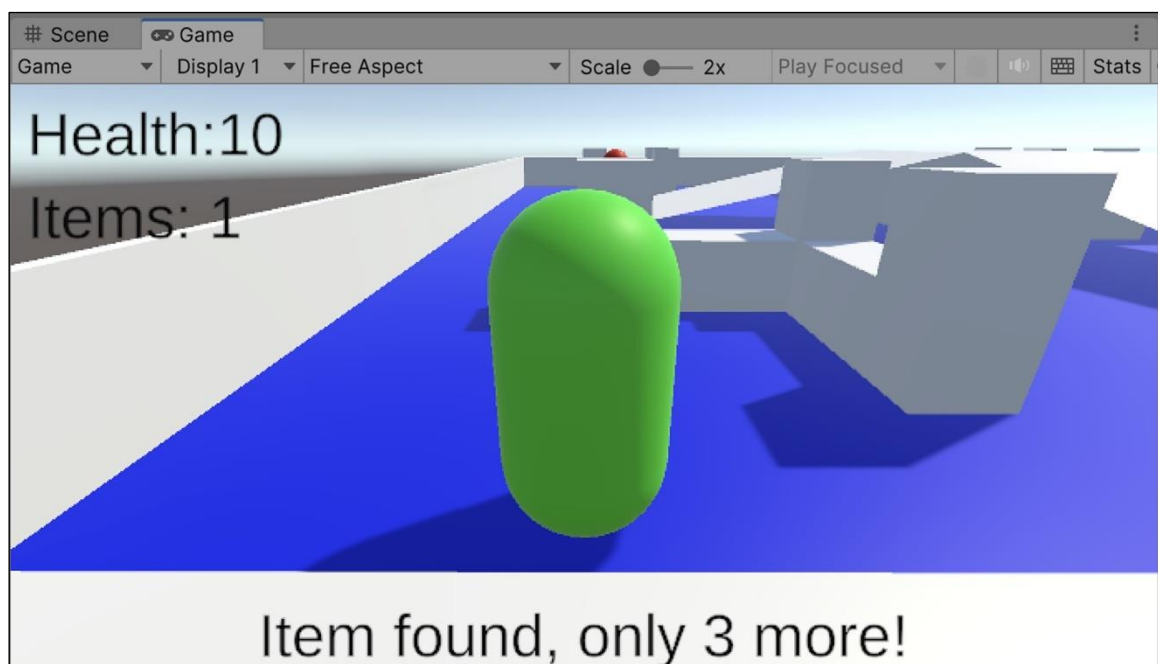
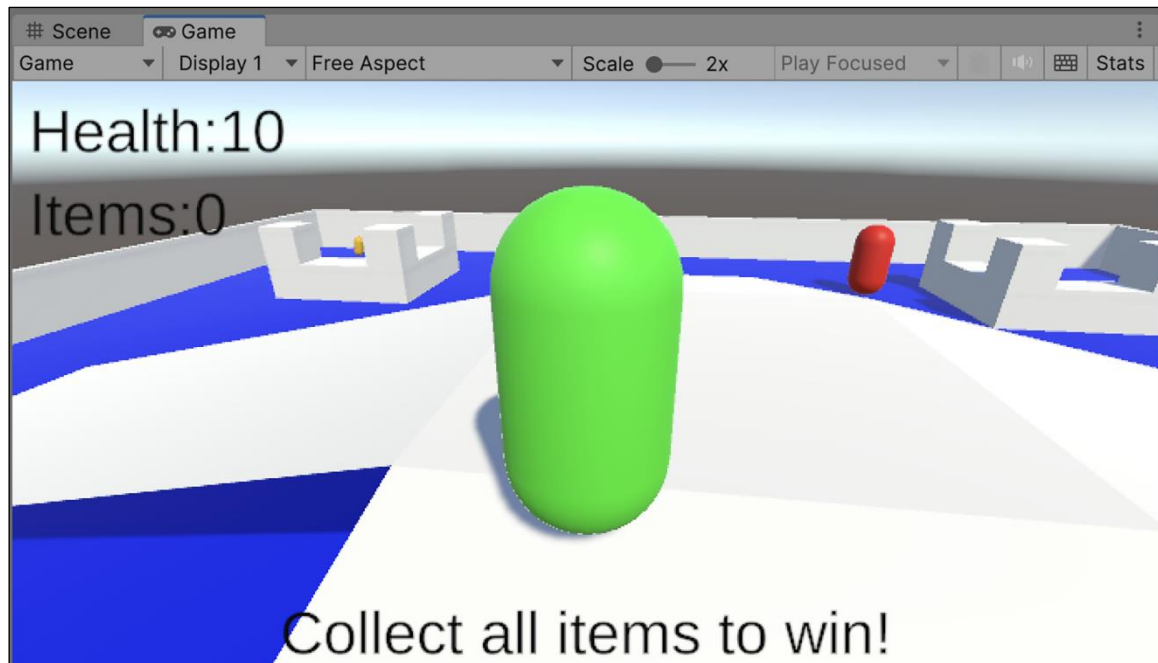


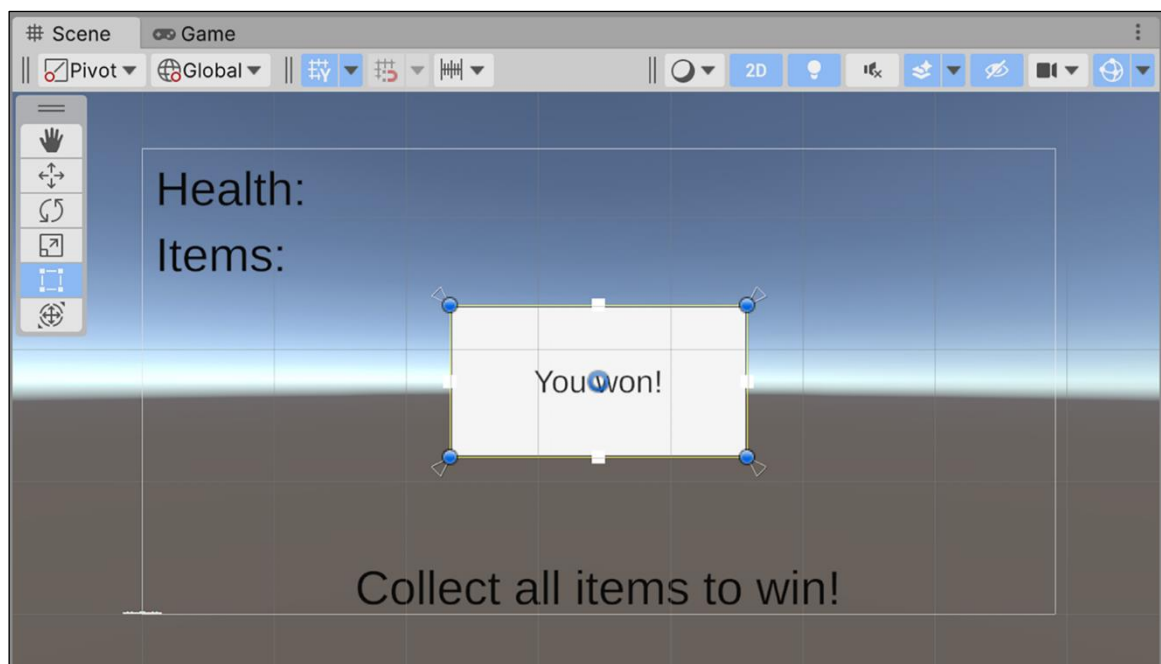
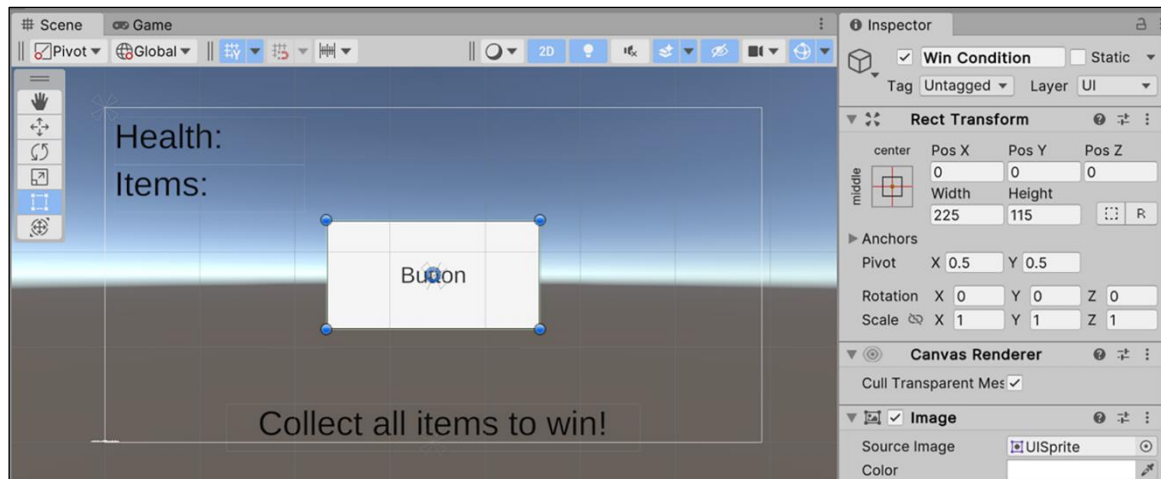


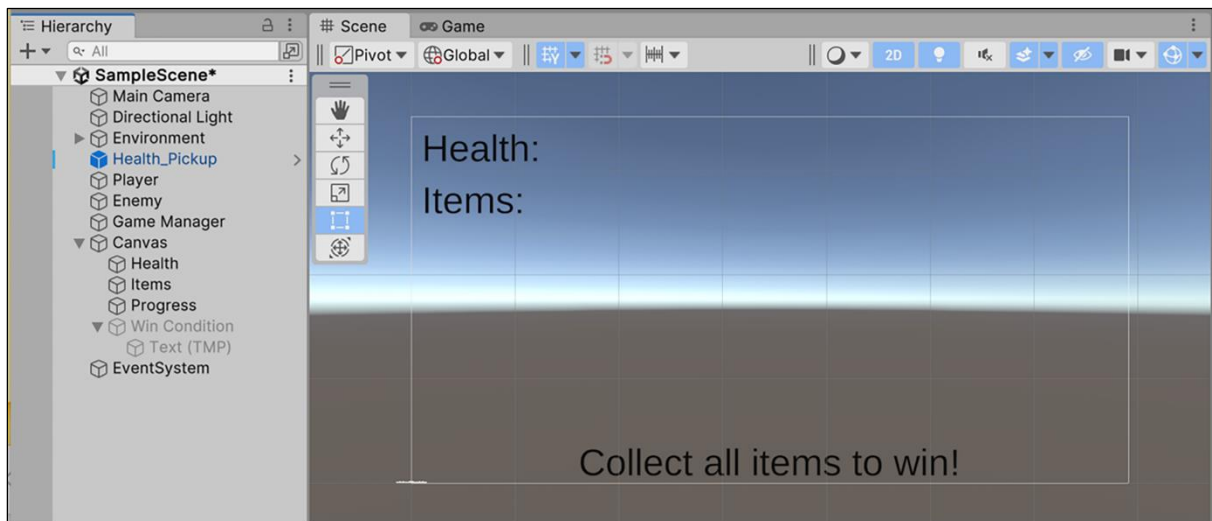
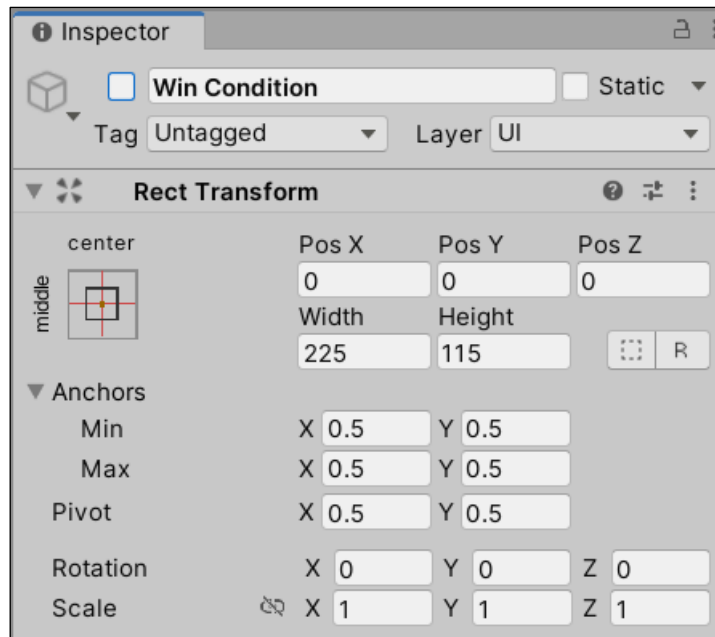


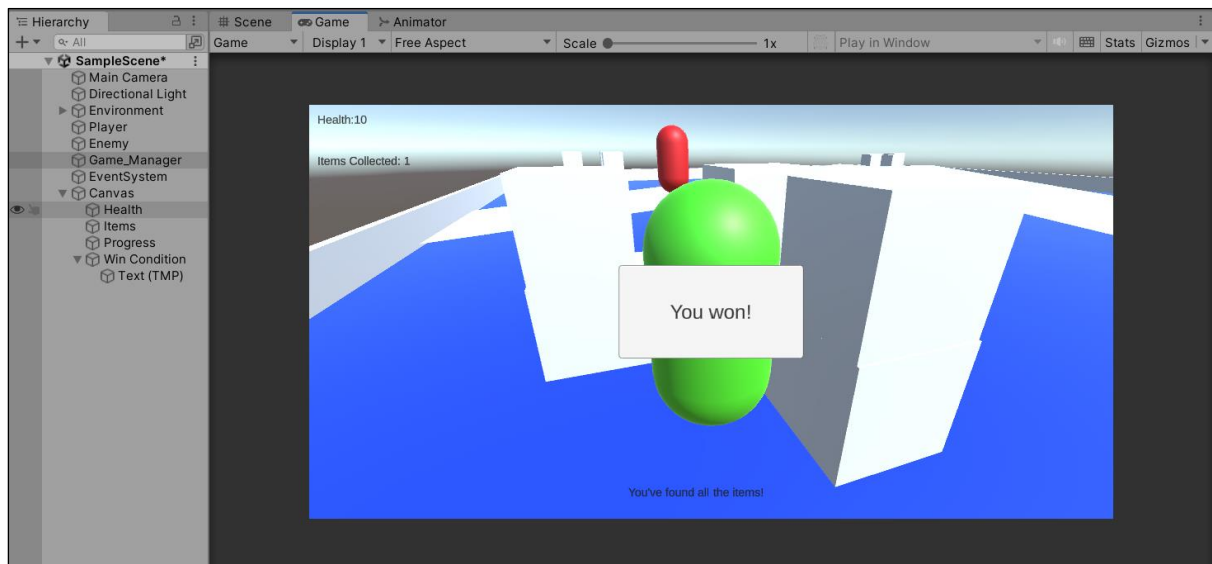
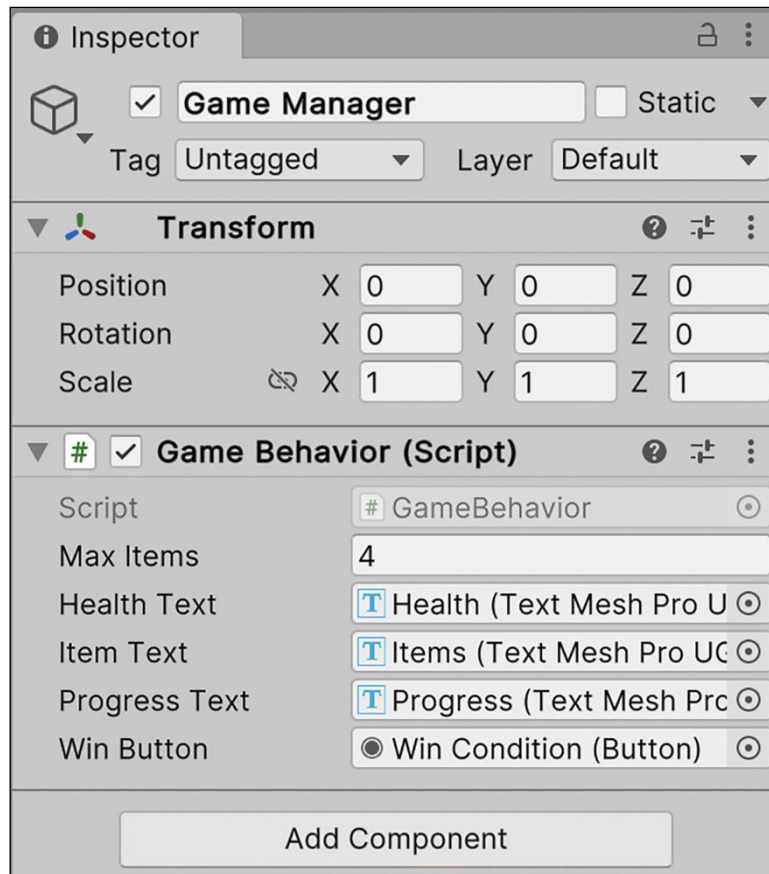


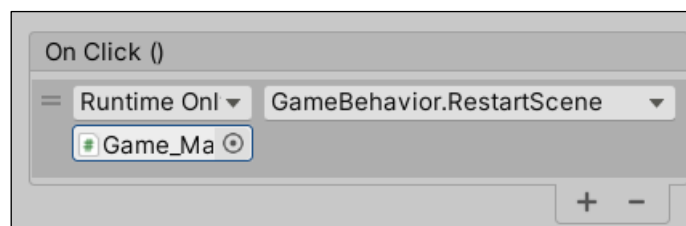
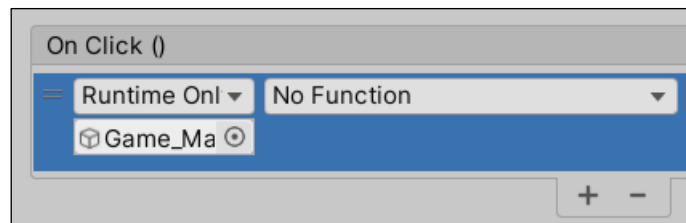
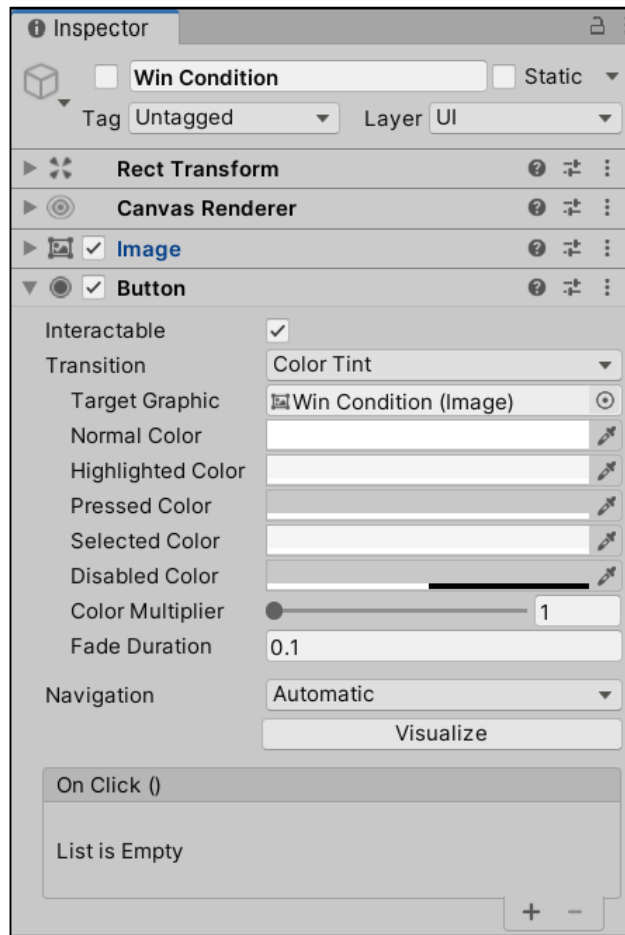












Lighting

Scene

Environment

Realtime Lightmaps

Baked Lightmaps

Progressive Update

Importance Sampling

Direct Samples

Indirect Samples

Environment Samples

Light Probe Samples

Max Bounces

Filtering

Lightmap Resolution

Lightmap Padding

Max Lightmap Size

Lightmap Compression

Ambient Occlusion

Directional Mode

Albedo Boost

Indirect Intensity

Lightmap Parameters

40

texels per unit

2

texels

1024

High Quality

Directional

1

1

Default-Medium

View

Workflow Settings

Light Probe Visualization

Auto Generate

Generate Lighting

0 Non-Directional Lightmaps

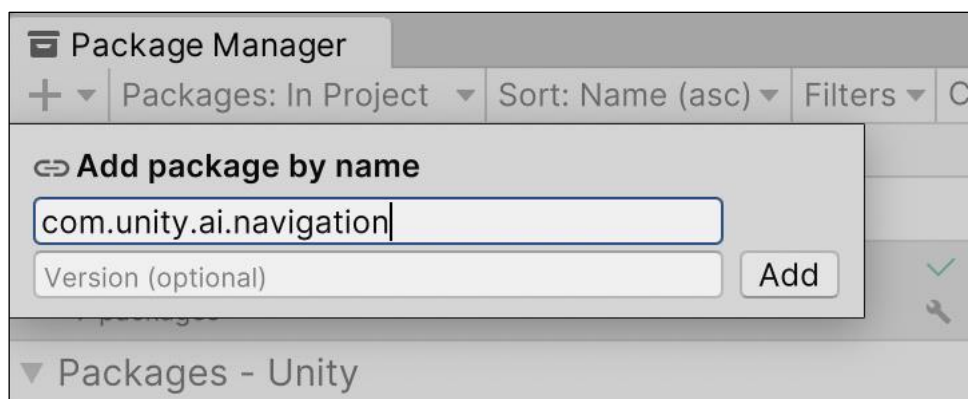
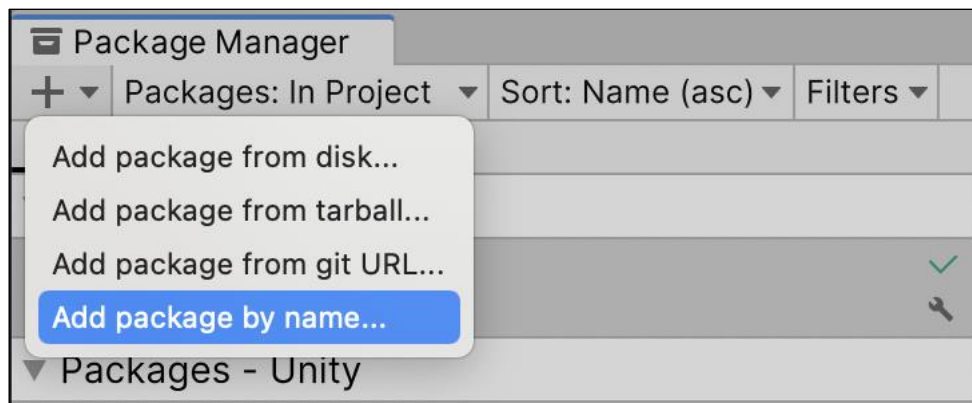
0 B

No Lightmaps

Occupied Texels: 0.0

Total Bake Time: 0:00:00

Chapter 9: Basic AI and Enemy Behavior



AI Navigation

Remove

1.1.4 · June 15, 2023 Release

From **Unity Registry** by Unity Technologies Inc.

com.unity.ai.navigation

[Documentation](#) | [Changelog](#) | [Licenses](#)

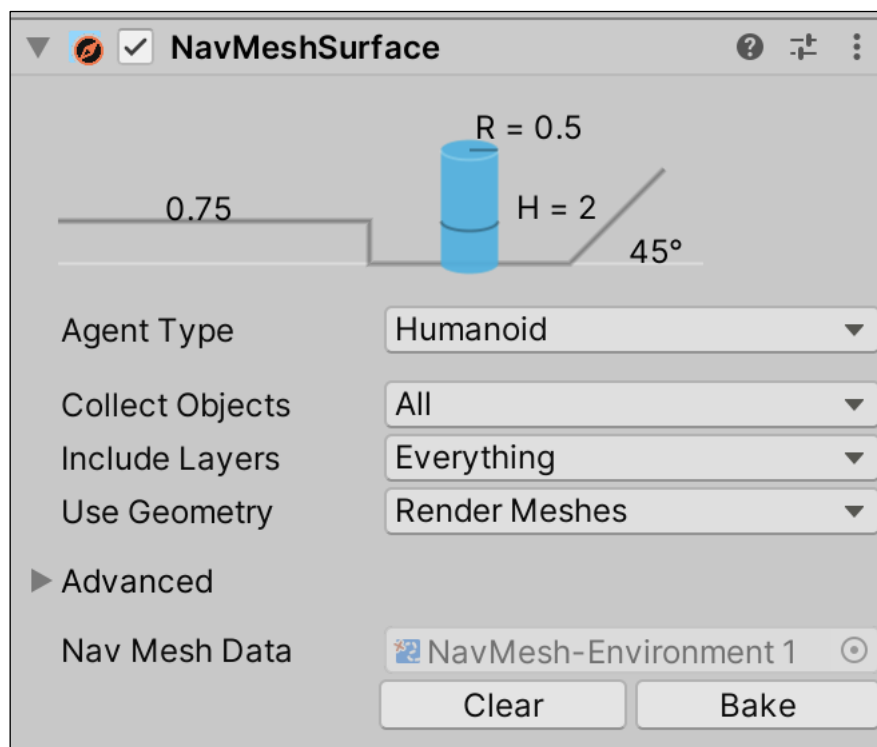
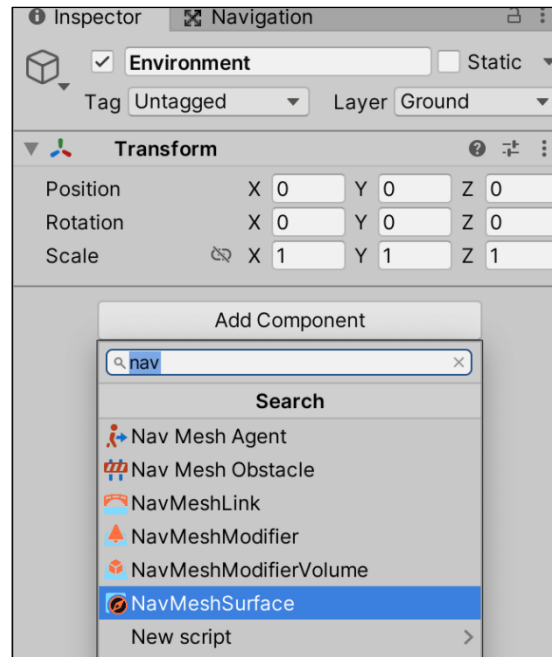
Description

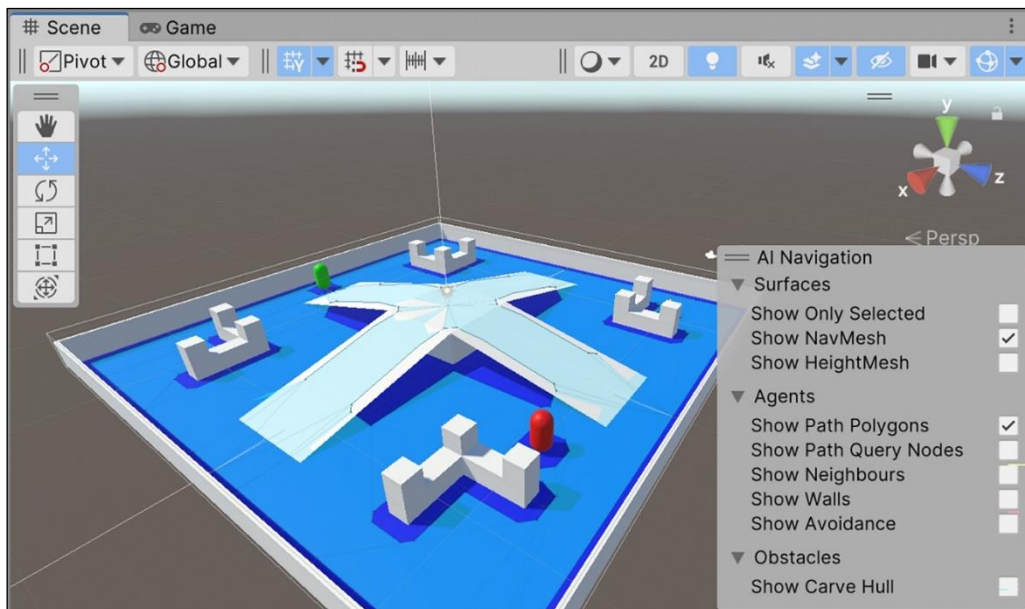
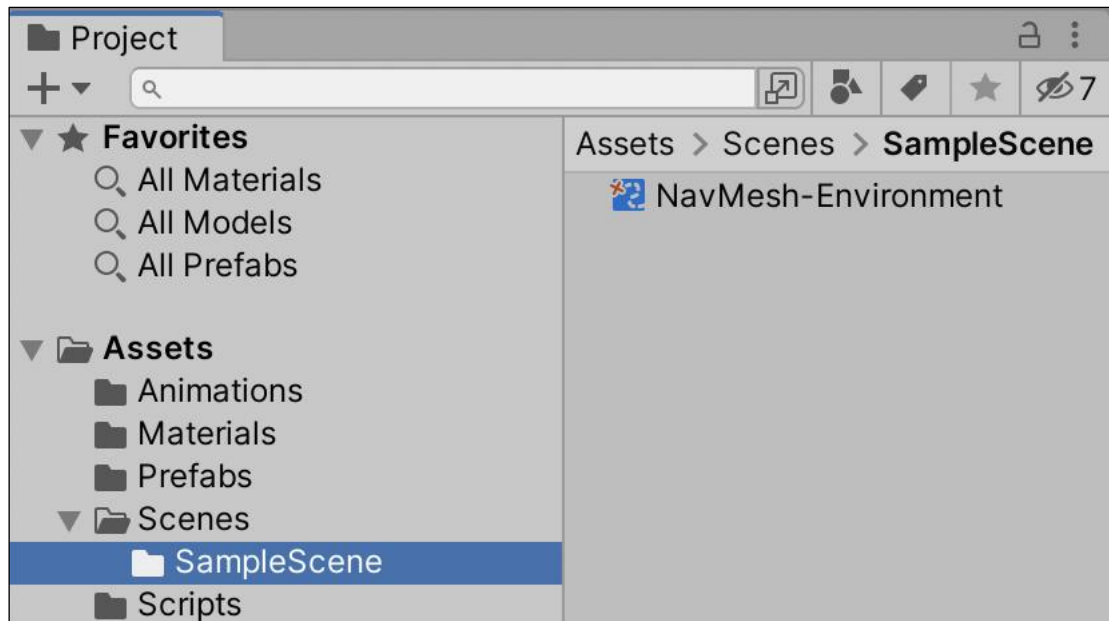
Version History

Dependencies

Samples

High-level NavMesh components for building and using NavMeshes at runtime and at edit time.





Nav Mesh Agent

?

Agent Type

Humanoid

Base Offset

1

Steering

Speed

3.5

Angular Speed

120

Acceleration

8

Stopping Distance

0

Auto Braking

☒

Obstacle Avoidance

Radius

0.5

Height

2

Quality

High Quality

Priority

50

Path Finding

Auto Traverse Off Mesh

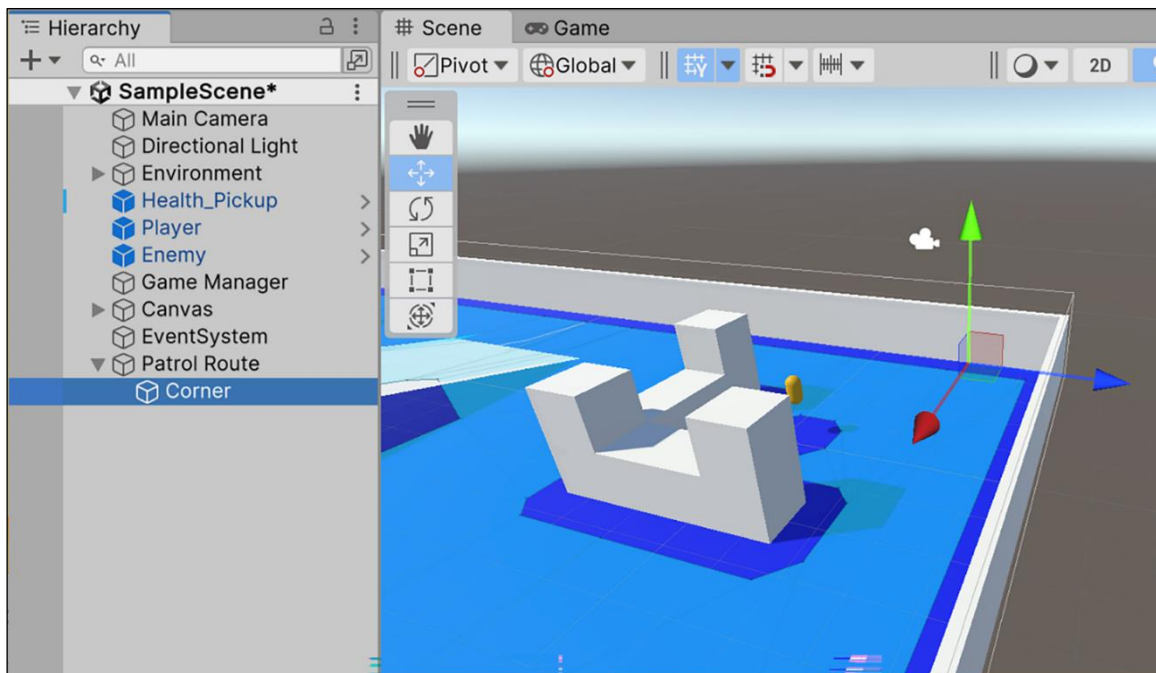
☒

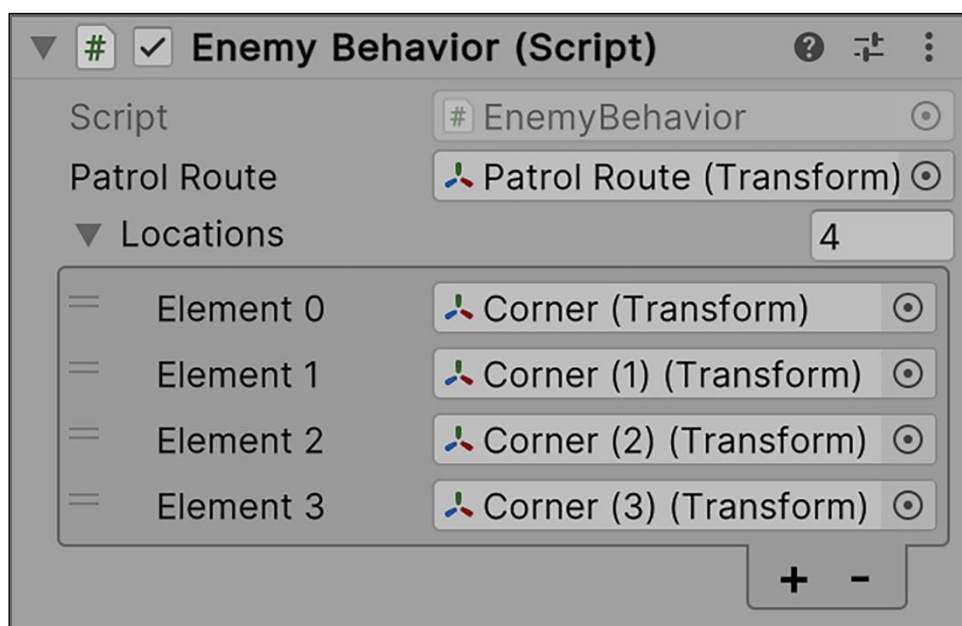
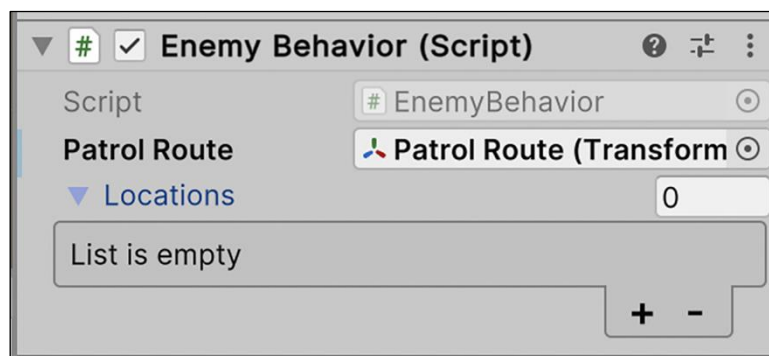
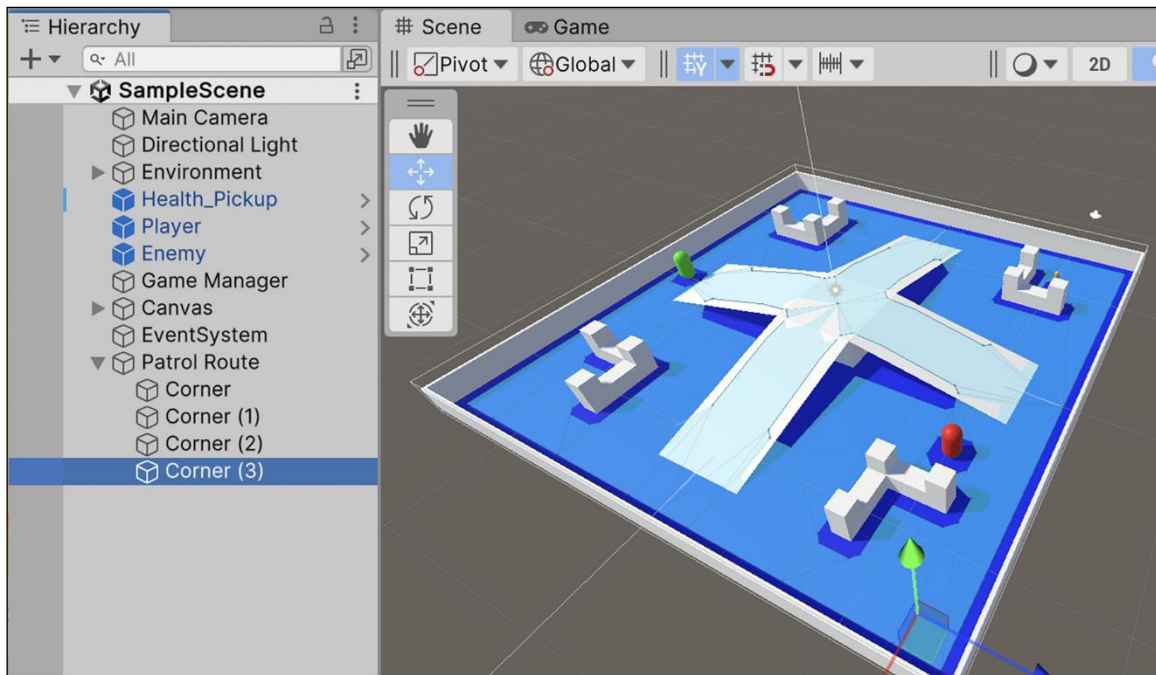
Auto Repath

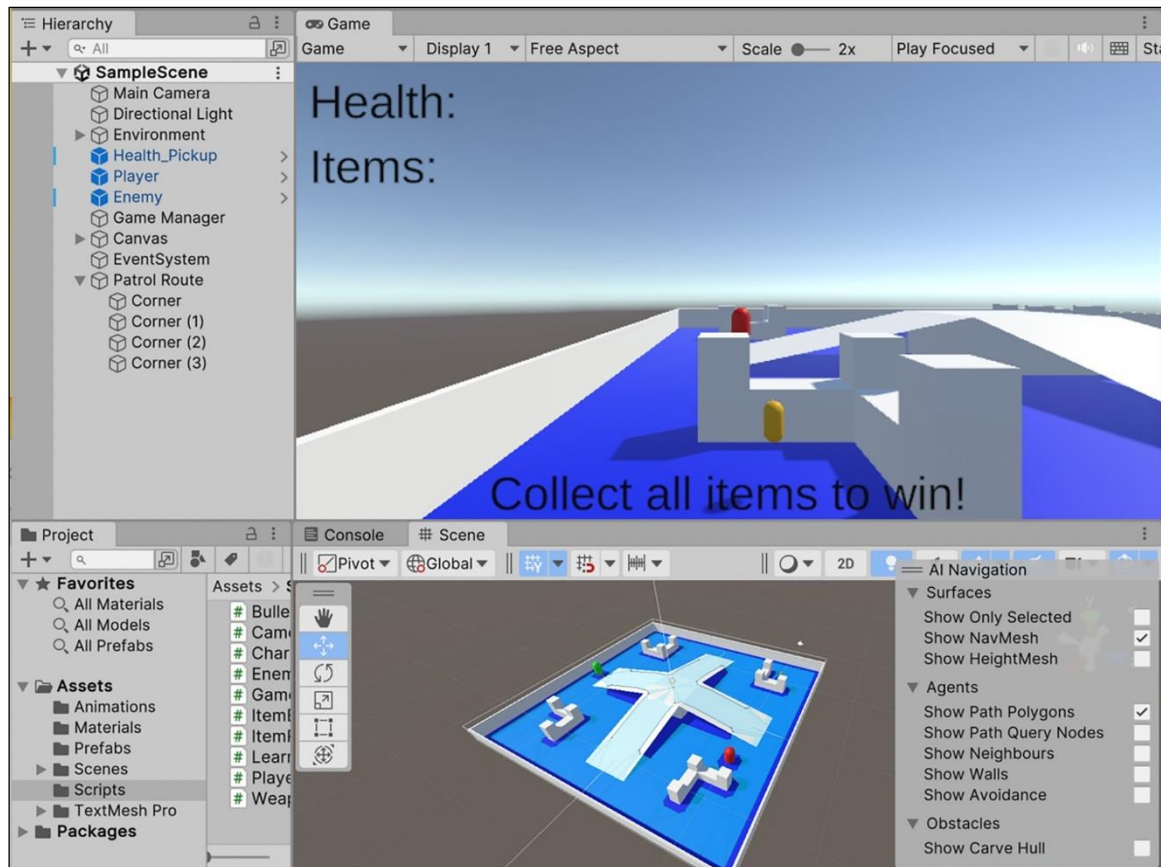
☒

Area Mask

Everything







#

✓

Game Behavior (Script)

?

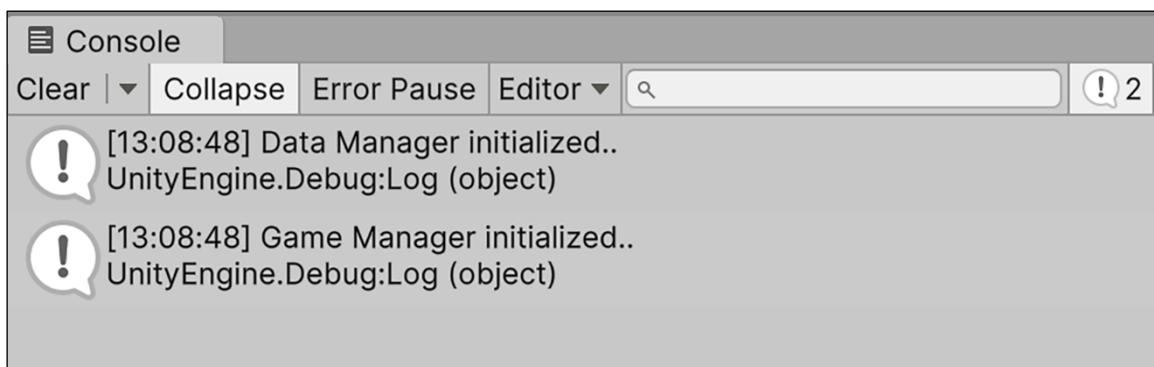
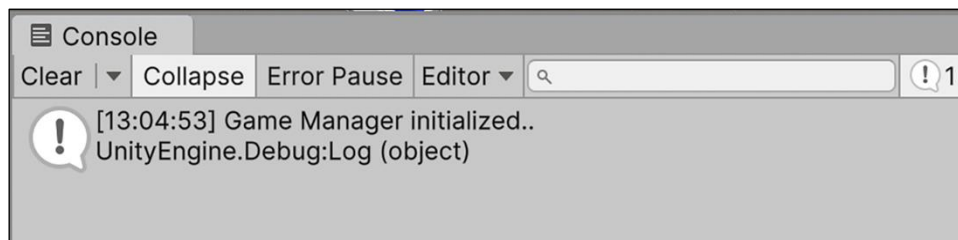
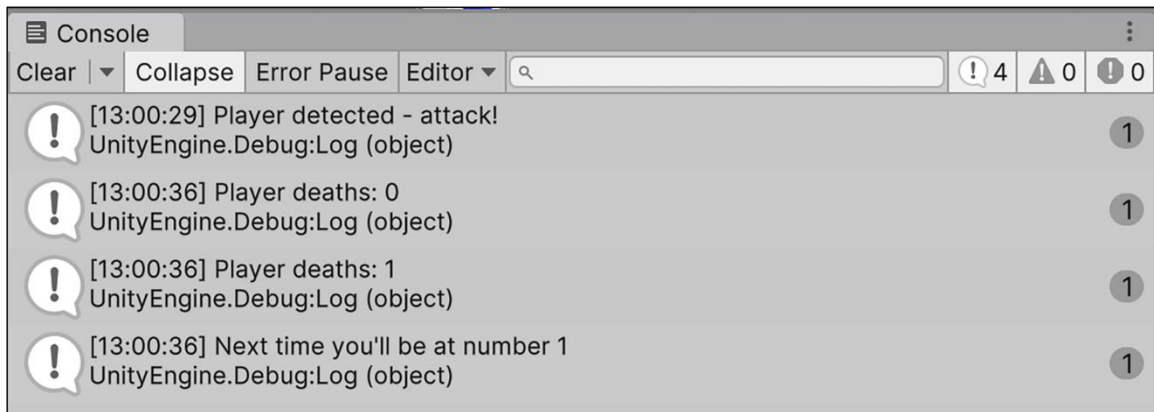
≡

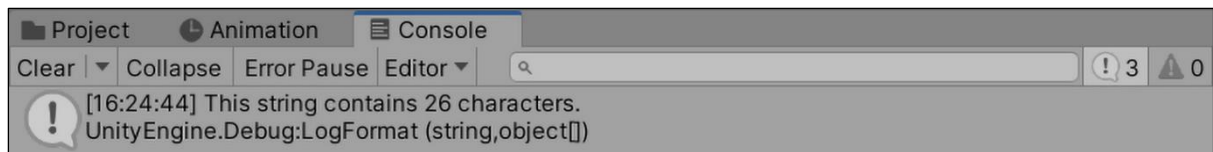
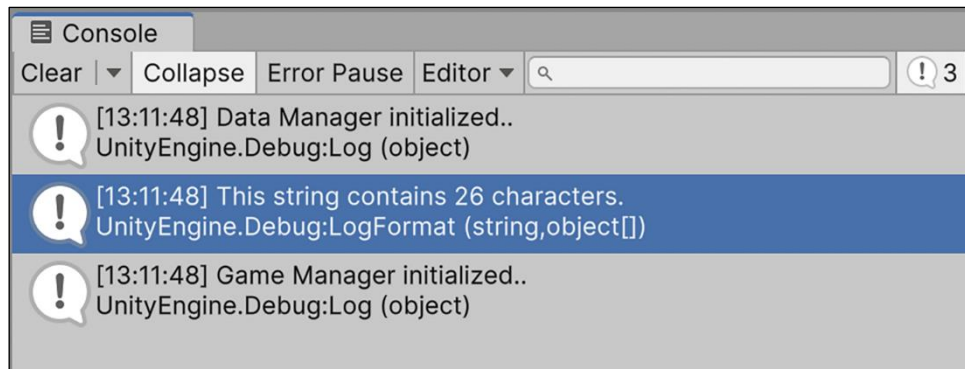
⋮

Script	# GameBehavior
Max Items	1
Health Text	T Health (Text Mesh Pro U
Item Text	T Items (Text Mesh Pro UC
Progress Text	T Progress (Text Mesh Prc
Win Button	Win Condition (Button)
Loss Button	Loss Condition (Button)

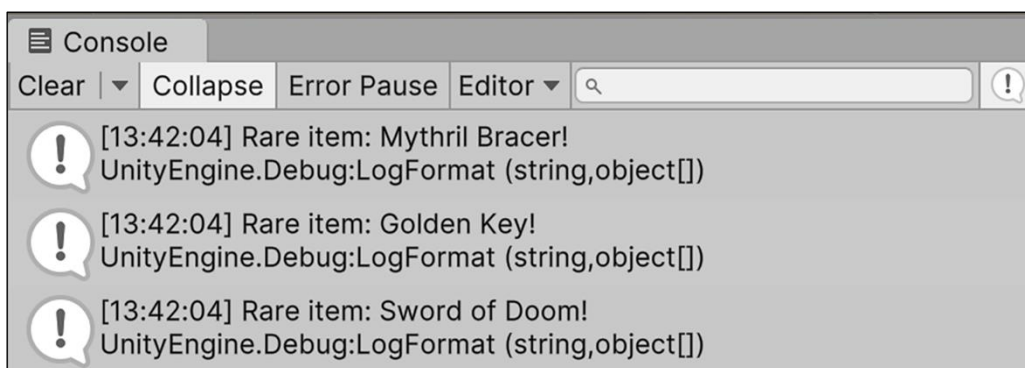
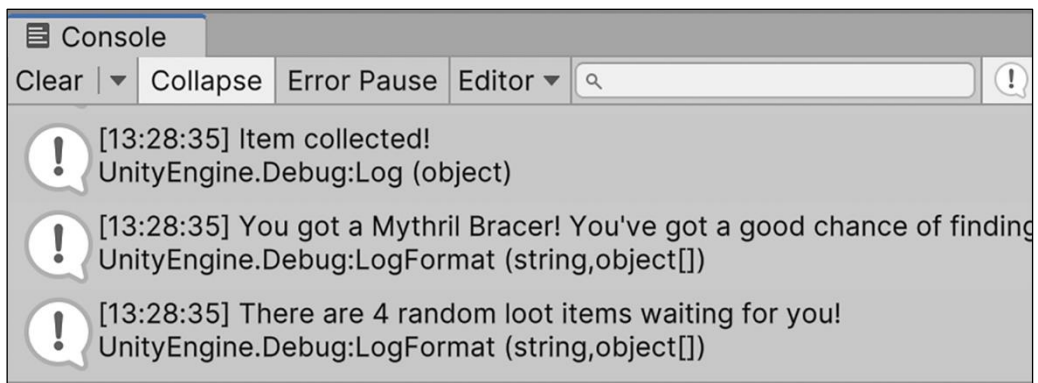
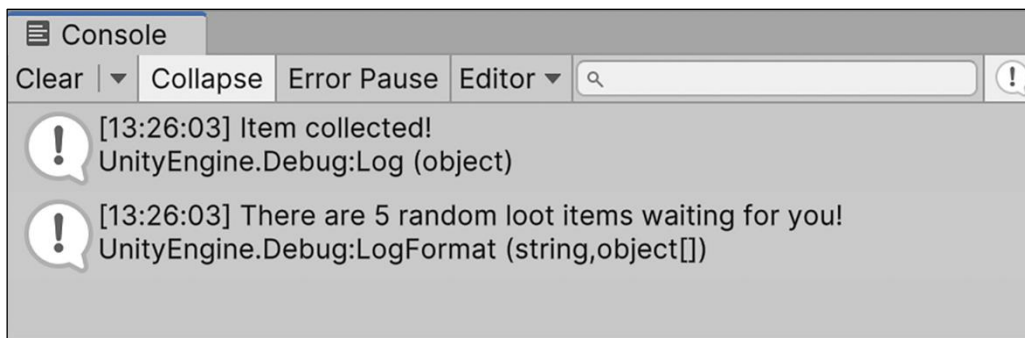
Chapter 10: Revisiting Types, Methods, and Classes

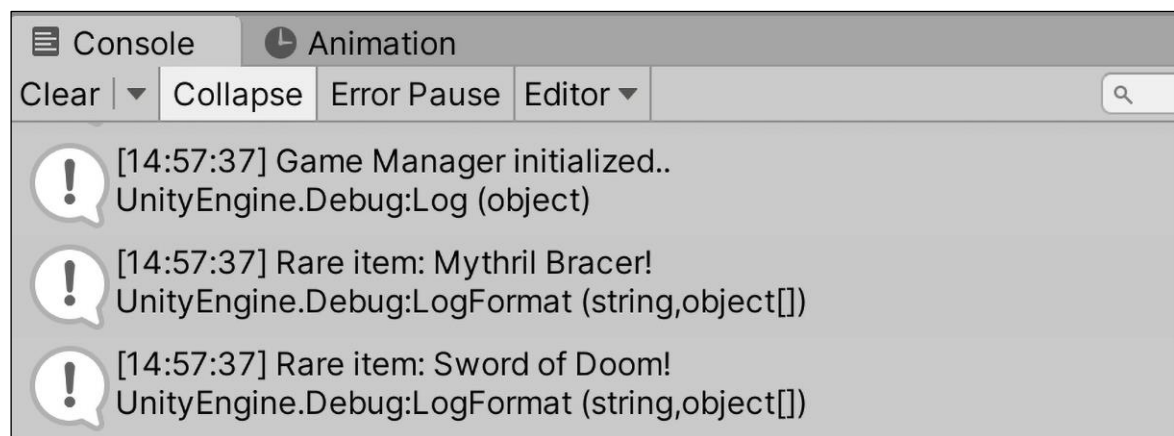
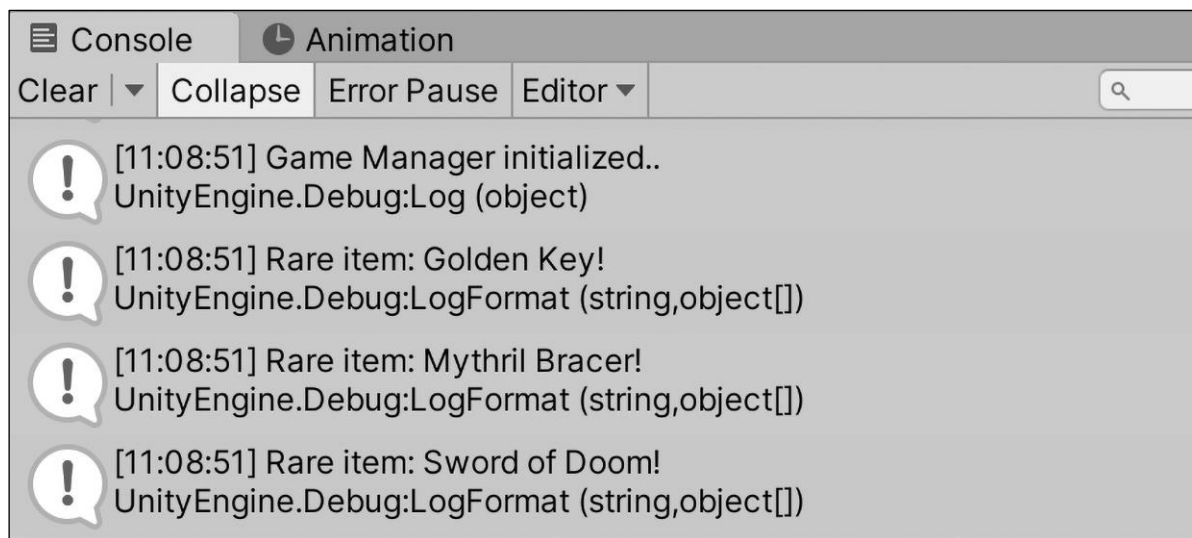
```
65  
66     public bool RestartLevel(int sceneIndex)  
67     {  
68         Utilities.RestartLevel()  
69     }  
70 }
```



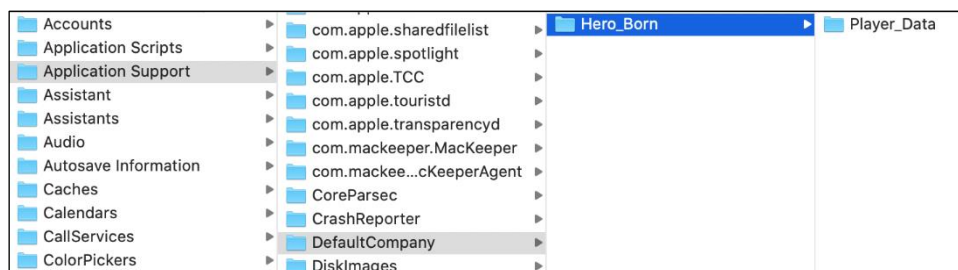
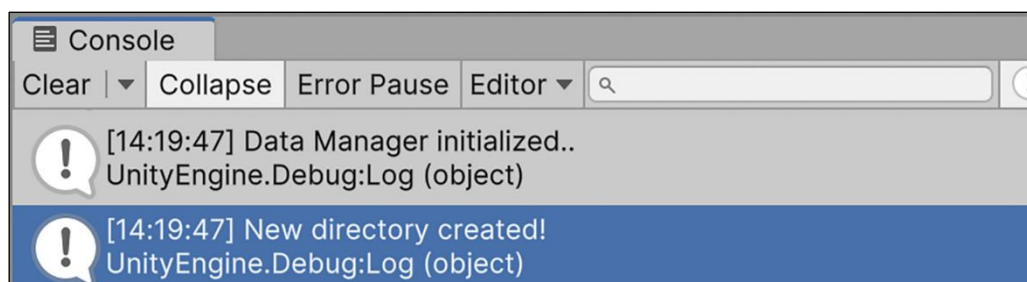
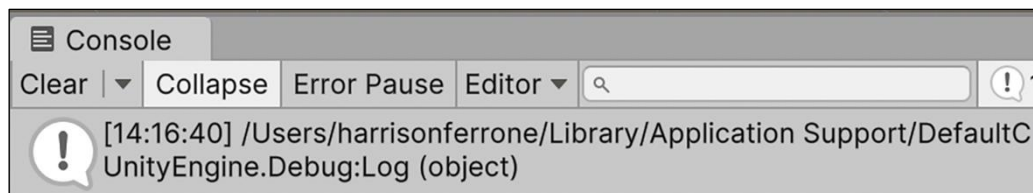
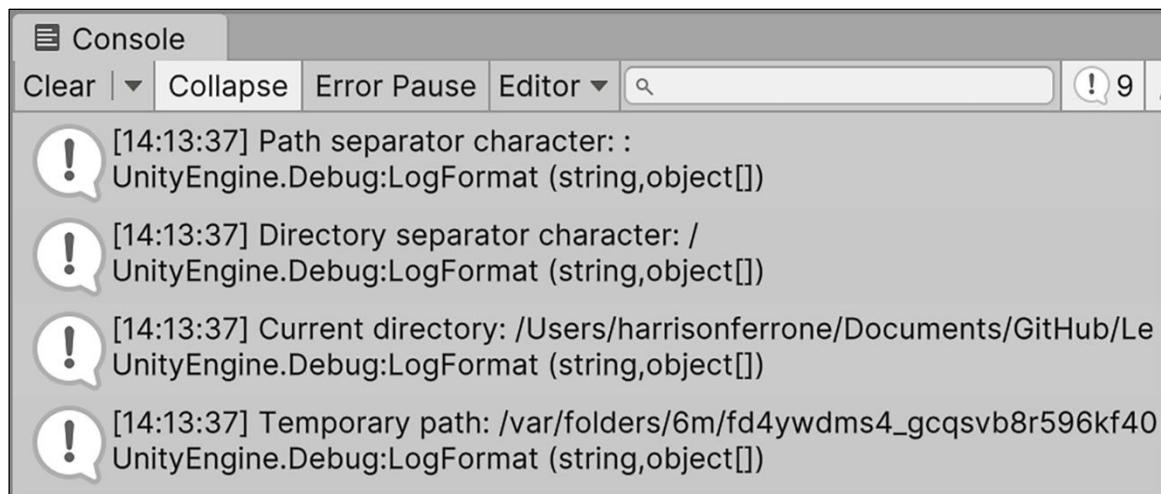


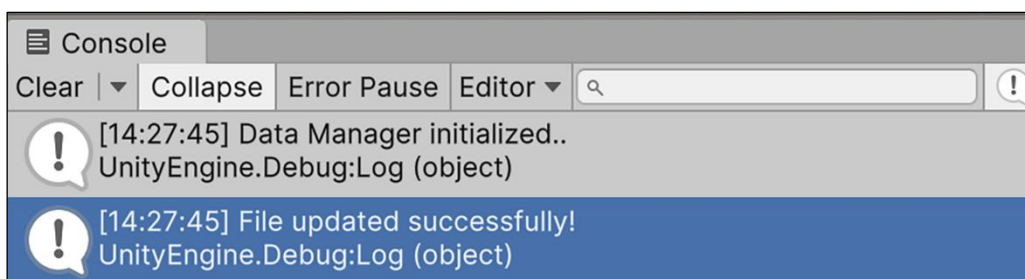
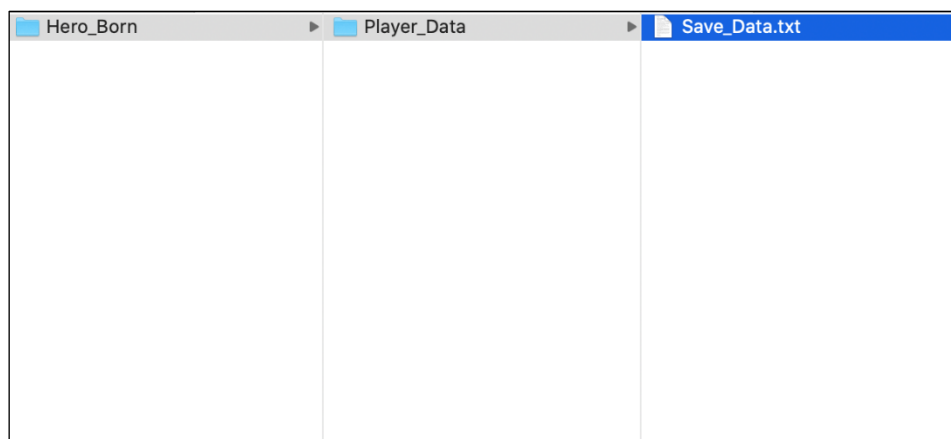
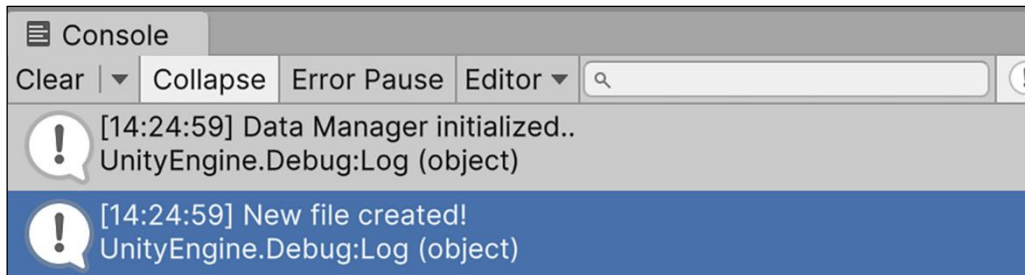
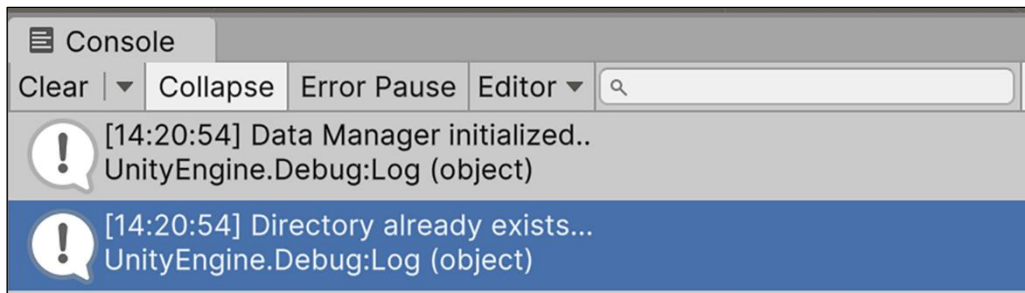
Chapter 11: Specialized Collection Types and LINQ

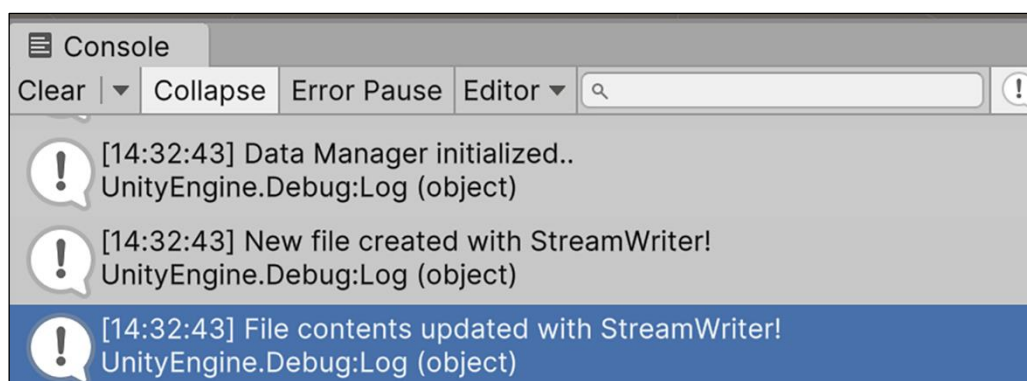
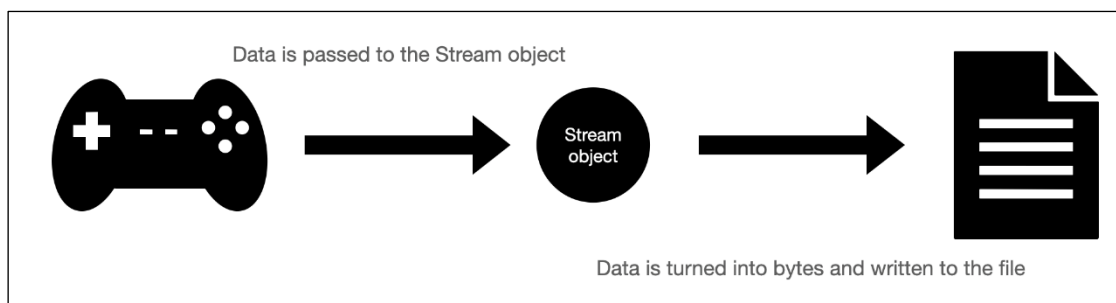
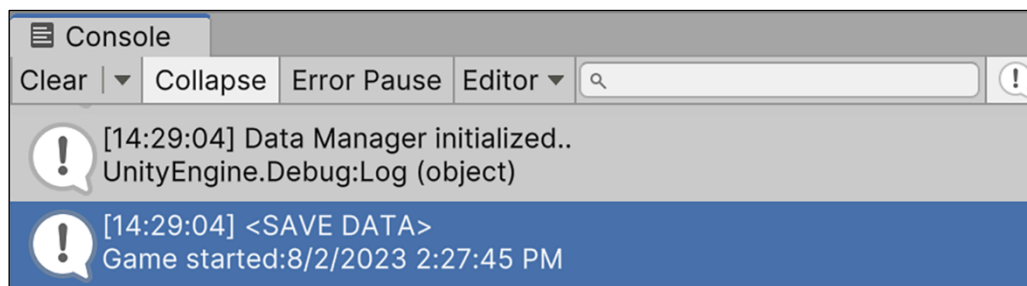
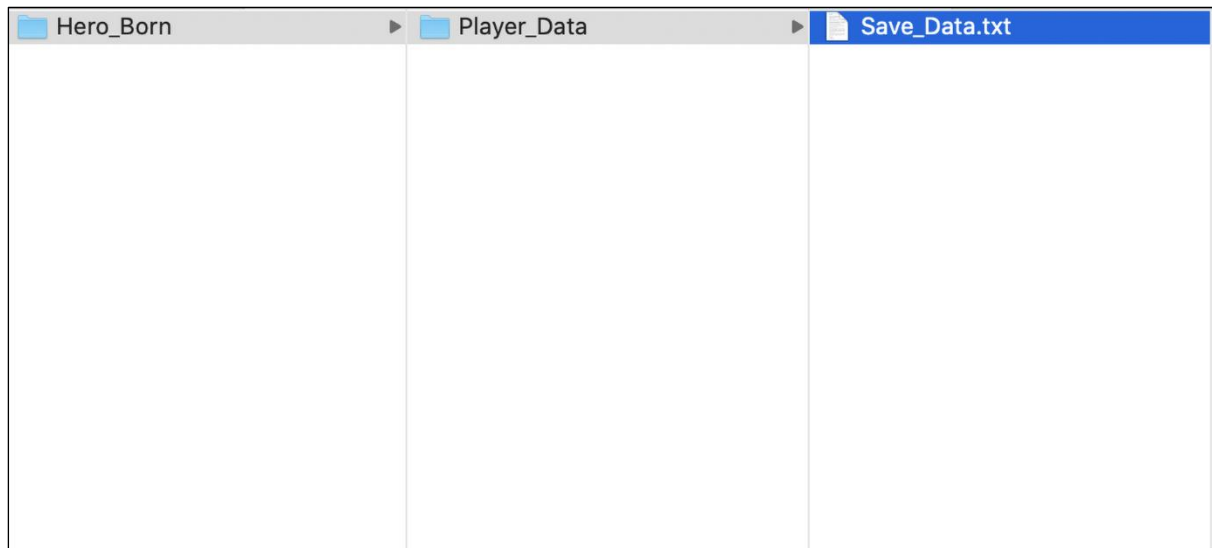


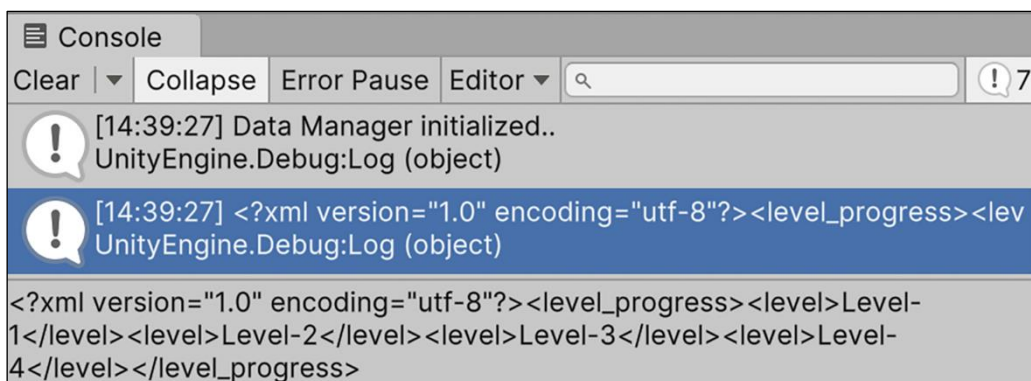
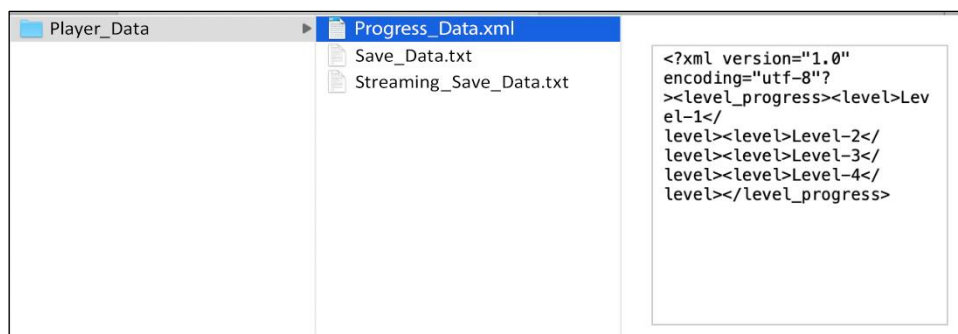
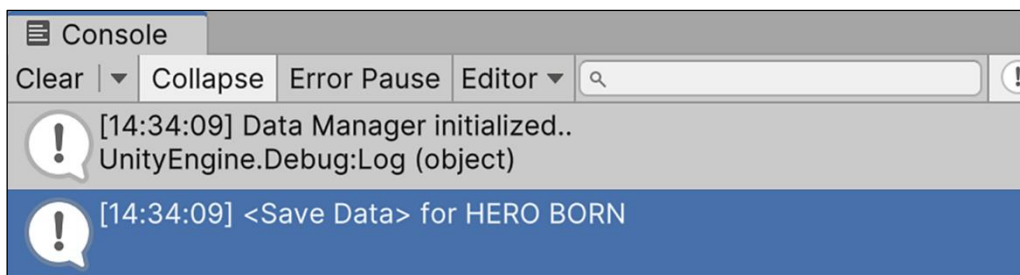
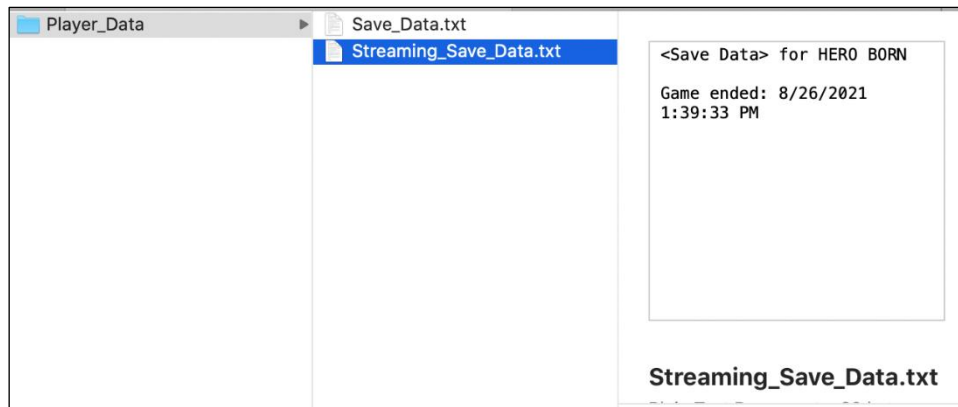


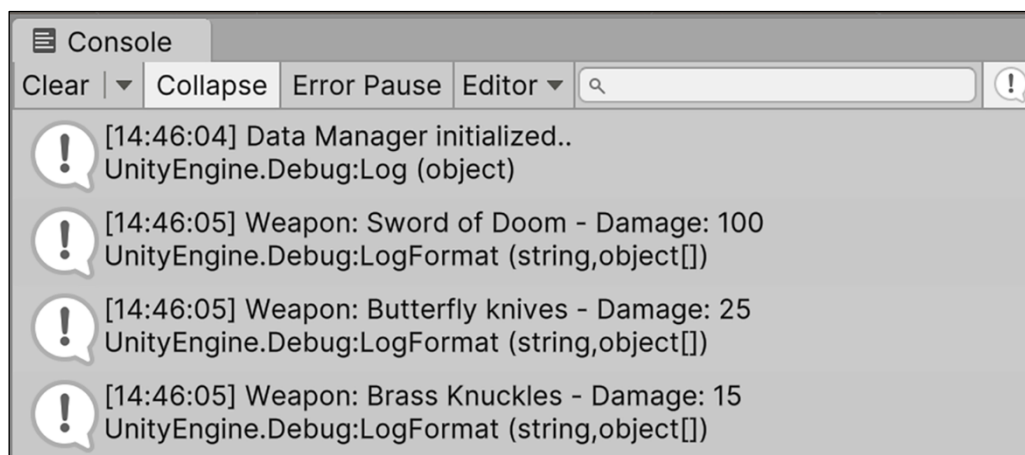
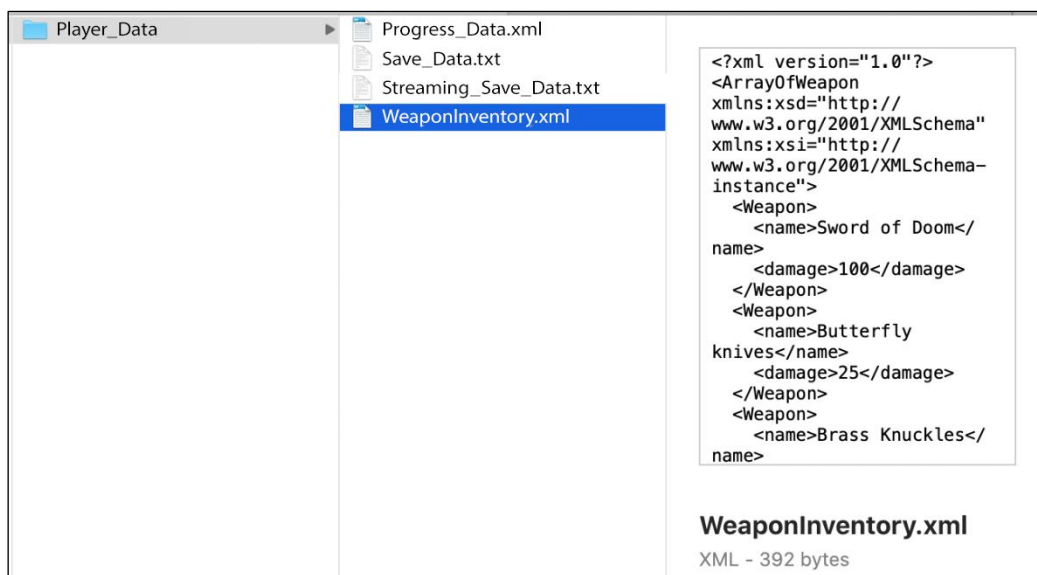
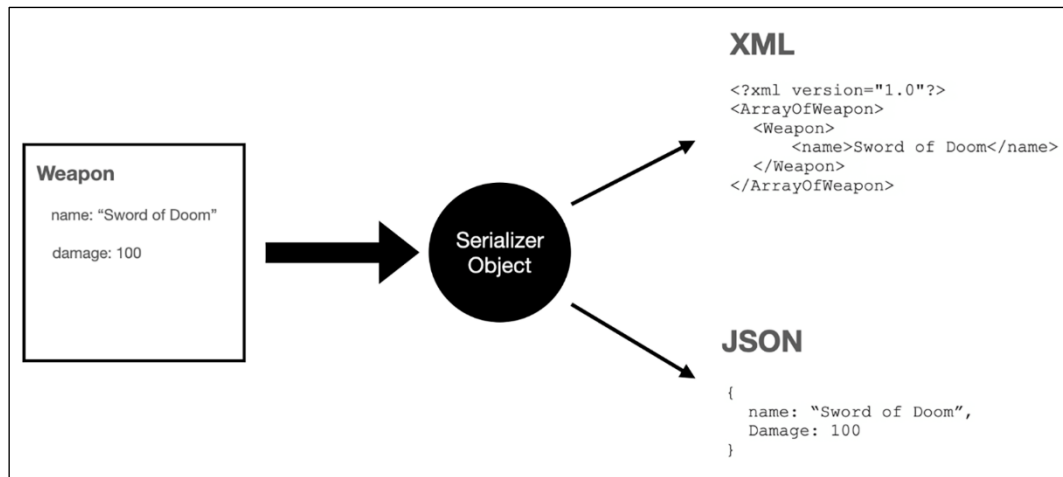
Chapter 12: Saving, Loading, and Serializing Data











Player_Data

Progress_Data.xml

Save_Data.txt

Streaming_Save_Data.txt

WeaponInventory.xml

WeaponJSON.json

```
{  
  "name": "Sword of Doom",  
  "damage": 100  
}
```

WeaponJSON.json

Plain Text Document - 51 bytes

Player_Data

Progress_Data.xml

Save_Data.txt

Streaming_Save_Data.txt

WeaponInventory.xml

WeaponJSON.json

```
{}
```

WeaponJSON.json

Plain Text Document - 3 bytes

Player_Data

Progress_Data.xml

Save_Data.txt

Streaming_Save_Data.txt

WeaponInventory.xml

WeaponJSON.json

```
{  
  "inventory": [  
    {  
      "name": "Sword of  
Doom",  
      "damage": 100  
    },  
    {  
      "name": "Butterfly  
knives",  
      "damage": 25  
    },  
    {  
      "name": "Brass  
Knuckles",  
      "damage": 15  
    }  
  ]  
}
```

WeaponJSON.json

Plain Text Document - 282 bytes



[15:10:51] Directory already exists...
UnityEngine.Debug:Log (object)



[15:10:51] Weapon: Sword of Doom - Damage: 100
UnityEngine.Debug:LogFormat (string,object[])



[15:10:51] Weapon: Butterfly knives - Damage: 25
UnityEngine.Debug:LogFormat (string,object[])



[15:10:51] Weapon: Brass Knuckles - Damage: 15
UnityEngine.Debug:LogFormat (string,object[])

Directory already exists...

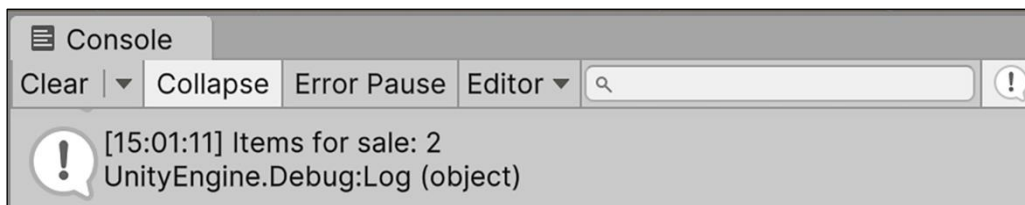
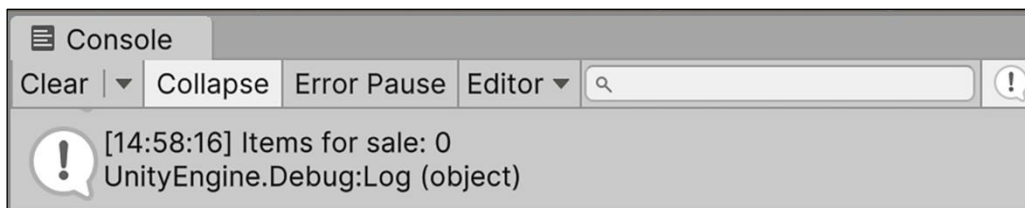
UnityEngine.Debug:Log (object)

DataManager:NewDirectory () (at [Assets/Scripts/DataManager.cs:73](#))

DataManager:Initialize () (at [Assets/Scripts/DataManager.cs:56](#))

DataManager:Start () (at [Assets/Scripts/DataManager.cs:47](#))

Chapter 13: Exploring Generics, Delegates, and Beyond



```
var itemShop = new Shop<string>();  
itemShop.AddItem(35);
```

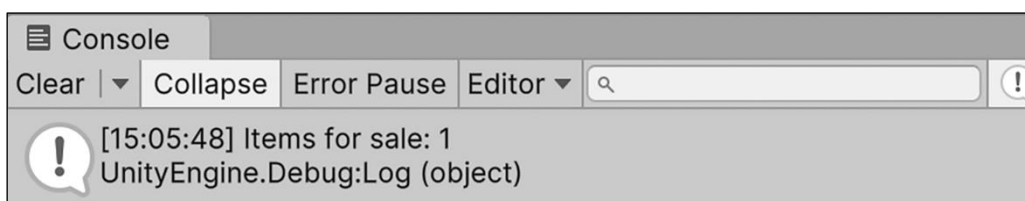
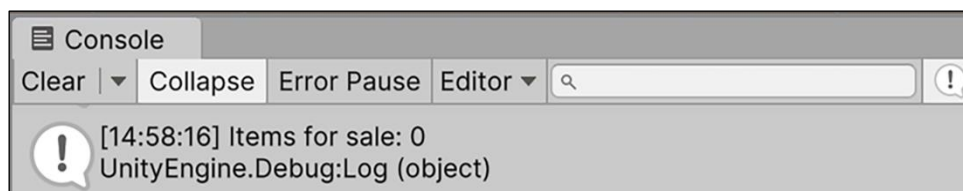
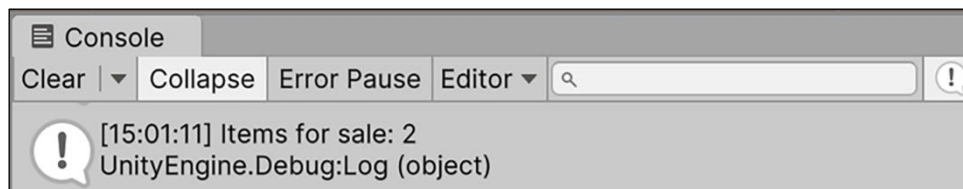


struct System.Int32

Represents a 32-bit signed integer.


[CS1503](#): Argument 1: cannot convert from 'int' to 'string'

[Show potential fixes](#)




```
var itemShop = new Shop<Collectable>();
itemShop.AddItem(new Potion());
itemShop.AddItem(new Antidote());
itemShop.AddItem("String");
```

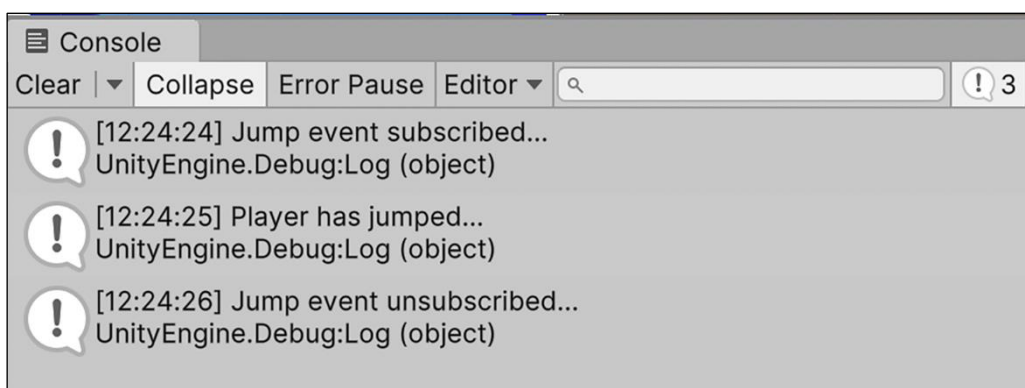
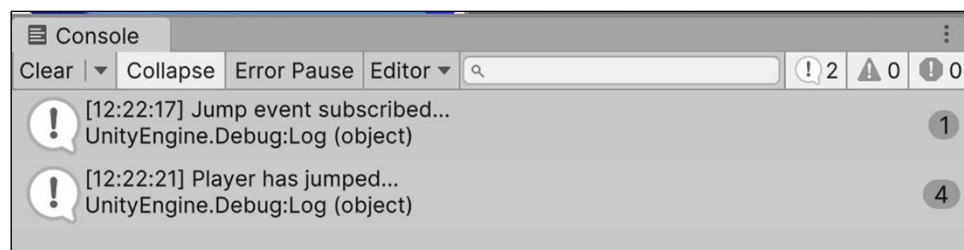
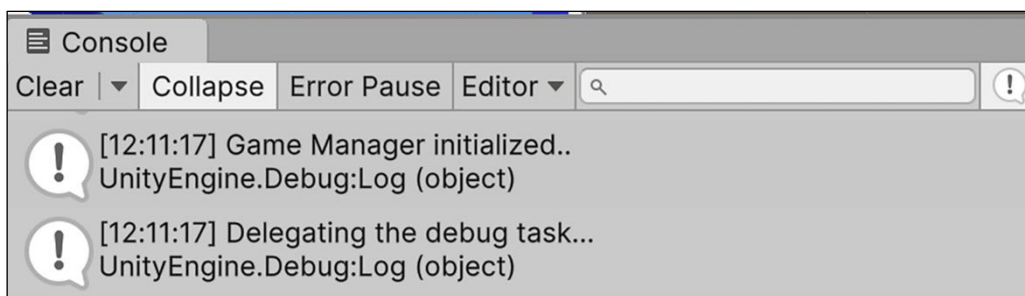
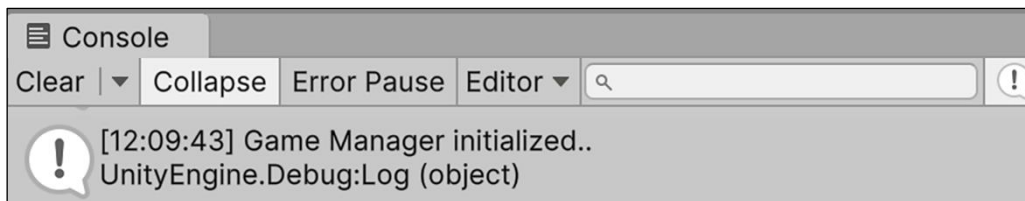
Debug.Log("I

 class System.String

Represents text as a series of Unicode characters.

[CS1503](#): Argument 1: cannot convert from 'string' to 'Collectable'

[Show potential fixes](#)





[14:44:50] ArgumentException: Scene index cannot be negative
Utilities.RestartLevel (System.Int32 sceneIndex) (at Assets/Scripts/Utilities.cs:37)



Console



Animation

Clear



Collapse

Error Pause

Editor



[14:53:41] Reverting to scene 0: System.ArgumentException : Scene index cannot be negative
at utilities.RestartLevel (System.Int32 SceneIndex) [0x00060] in /Users/harrisonferrone/Destk



[14:53:41] Level restart has completed...
UnityEngine.Debug:Log (object)