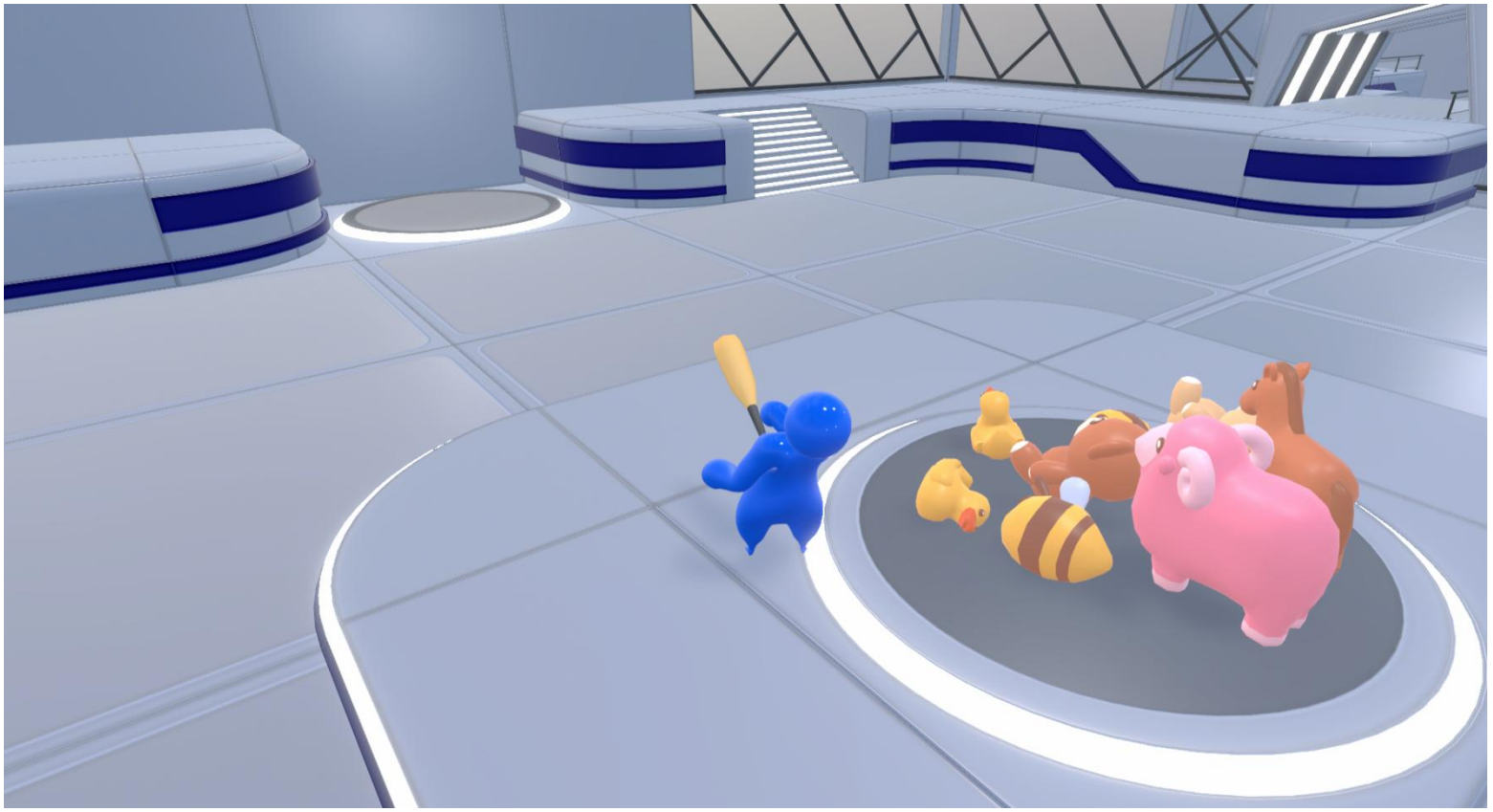


OodlesParty 1.2.1 Documentation

Introduction

Oodles Party is an exciting physics-based fighting party game that allows players to engage in intense hand-to-hand combat, wield weapons, and cooperate to manipulate physical props. The game's physics are powered by Unity's built-in physics engine.

Network synchronization is crucial for physics-based fighting games. In **Oodles Party**, the server calculates all physics, ensuring a smooth and consistent experience for players. The network synchronization is based on the Mirror framework.

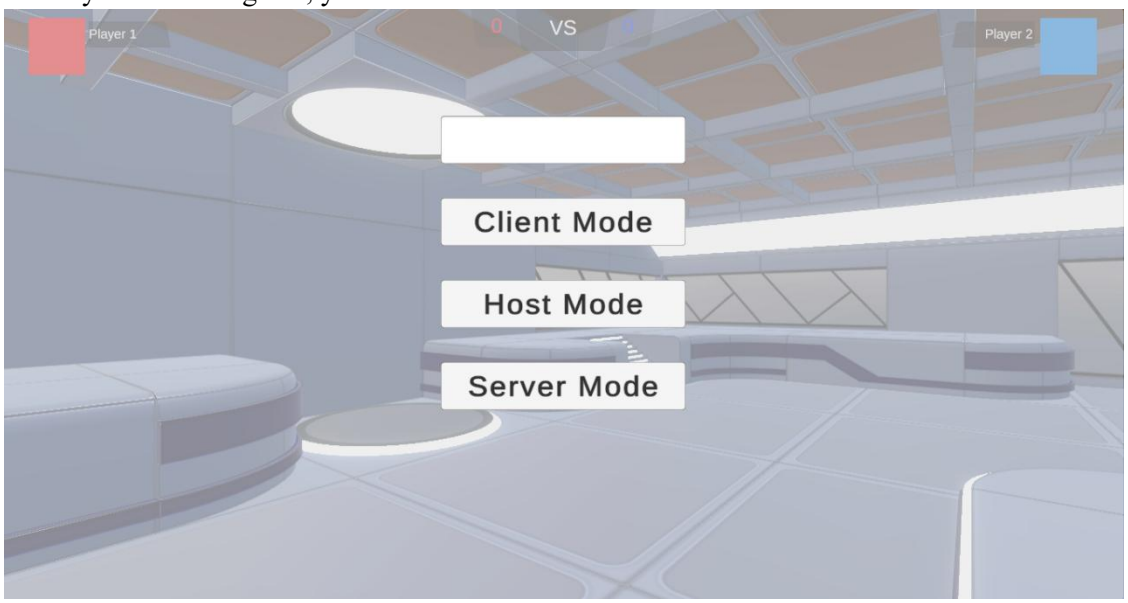
Table of Contents

Part I: Configuration and Running

Before starting the online mode, please open the StartScreen scene and choose the online option in the mode selection. Note: Oodles Party has been separated from the original code for single-player functionality, and the first single-player test scene, Local Demo, has been added. Currently, only character animation control has been incorporated, and upcoming versions will gradually support complete combat features.

1. Choose Mode

When you enter the game, you'll encounter the Mode selection menu:



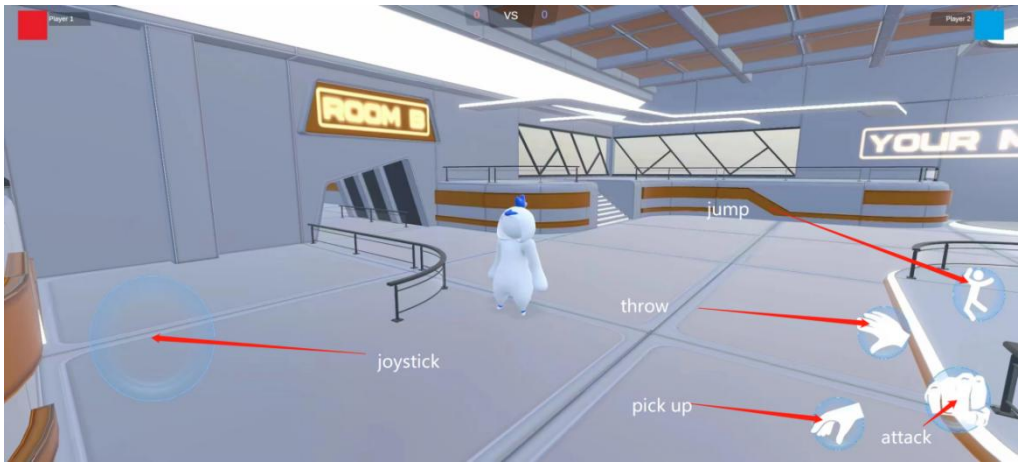
- Choose Host Mode to directly start the game as both the server and the client.
- Choose Server Mode to initiate a server, allowing other clients to connect.
- Choose Client Mode after entering the target server's IP address to connect to the server.

2. HowTo Play

On the Editor or PC

- KeyboardW,A, S, D: Movement
- Mouse Left: Attack
- Mouse Right: Pick up
- Keyboard T: Throw

2. On Mobile



Part II: Customizing Character Physics

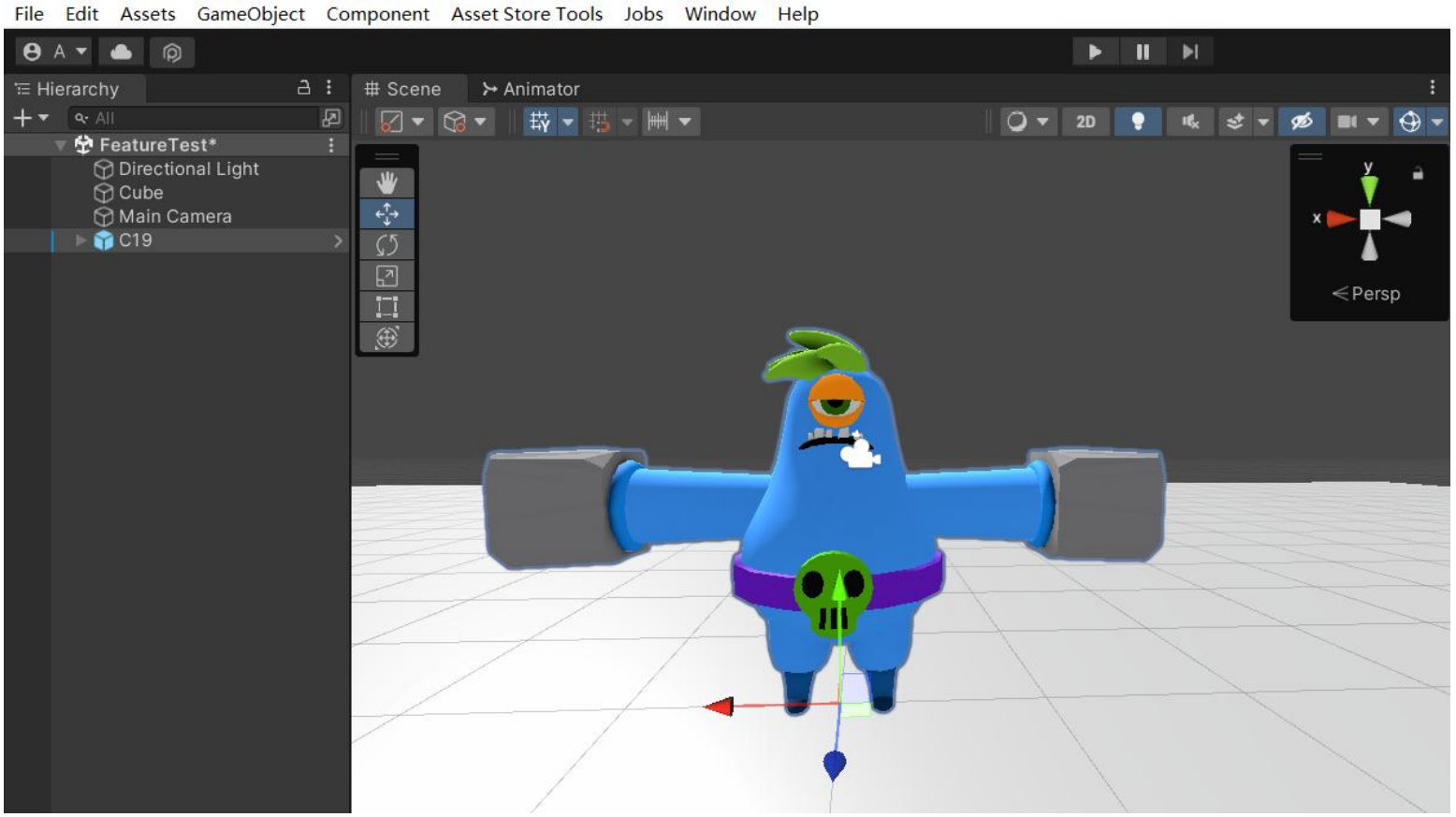
Introduction to Active Ragdoll

Oodles Party employs an active ragdoll character physics animation solution. Active ragdoll is a game animation technology used to achieve more realistic physical effects. Unlike traditional ragdoll techniques that create a limp body when a character dies, active ragdoll retains control, resulting in more natural and expressive actions.

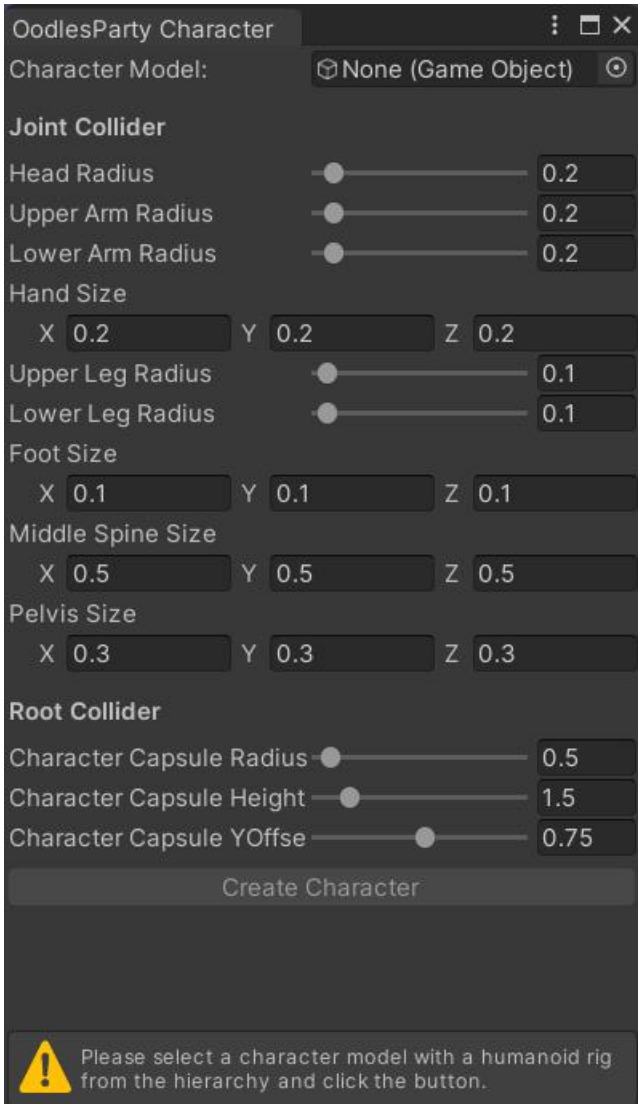
The following steps will guide you on customizing your game characters:

2.1 Generate Characters

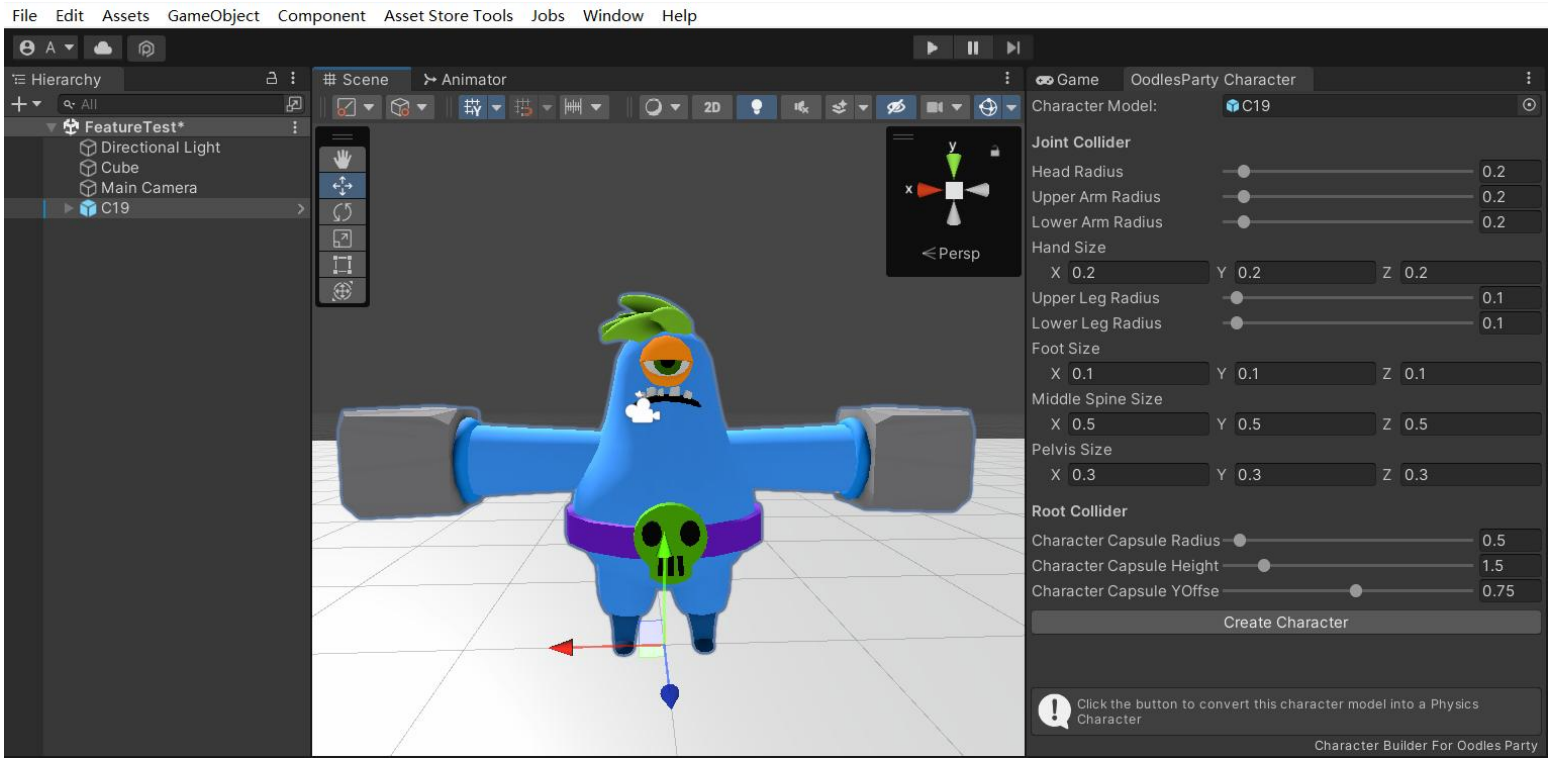
Open a new scene and place a Humanoid character model into it.



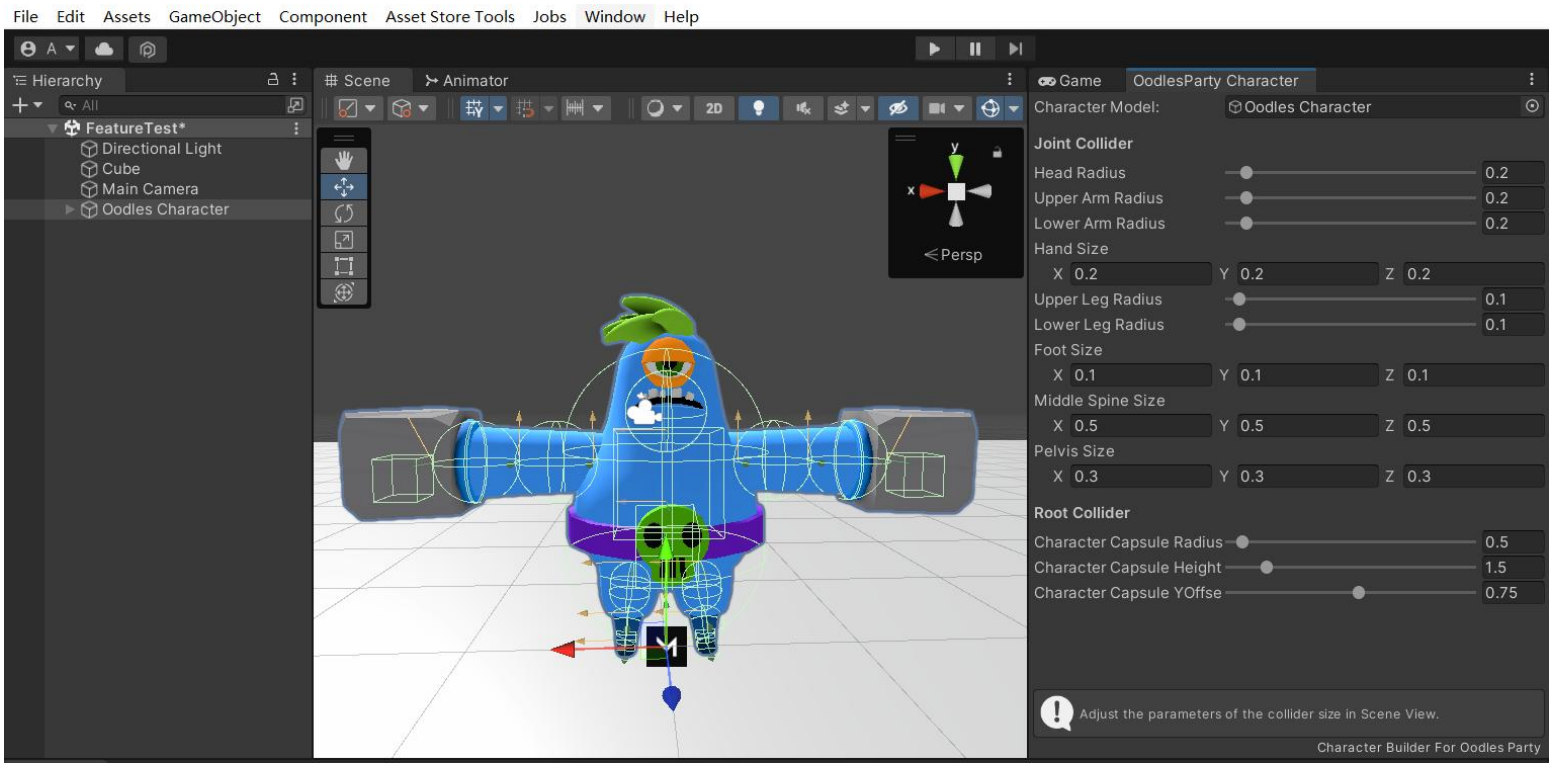
Open the Window -> OodlesPartyCharacter panel.



Next, select the character model in the Hierarchy.

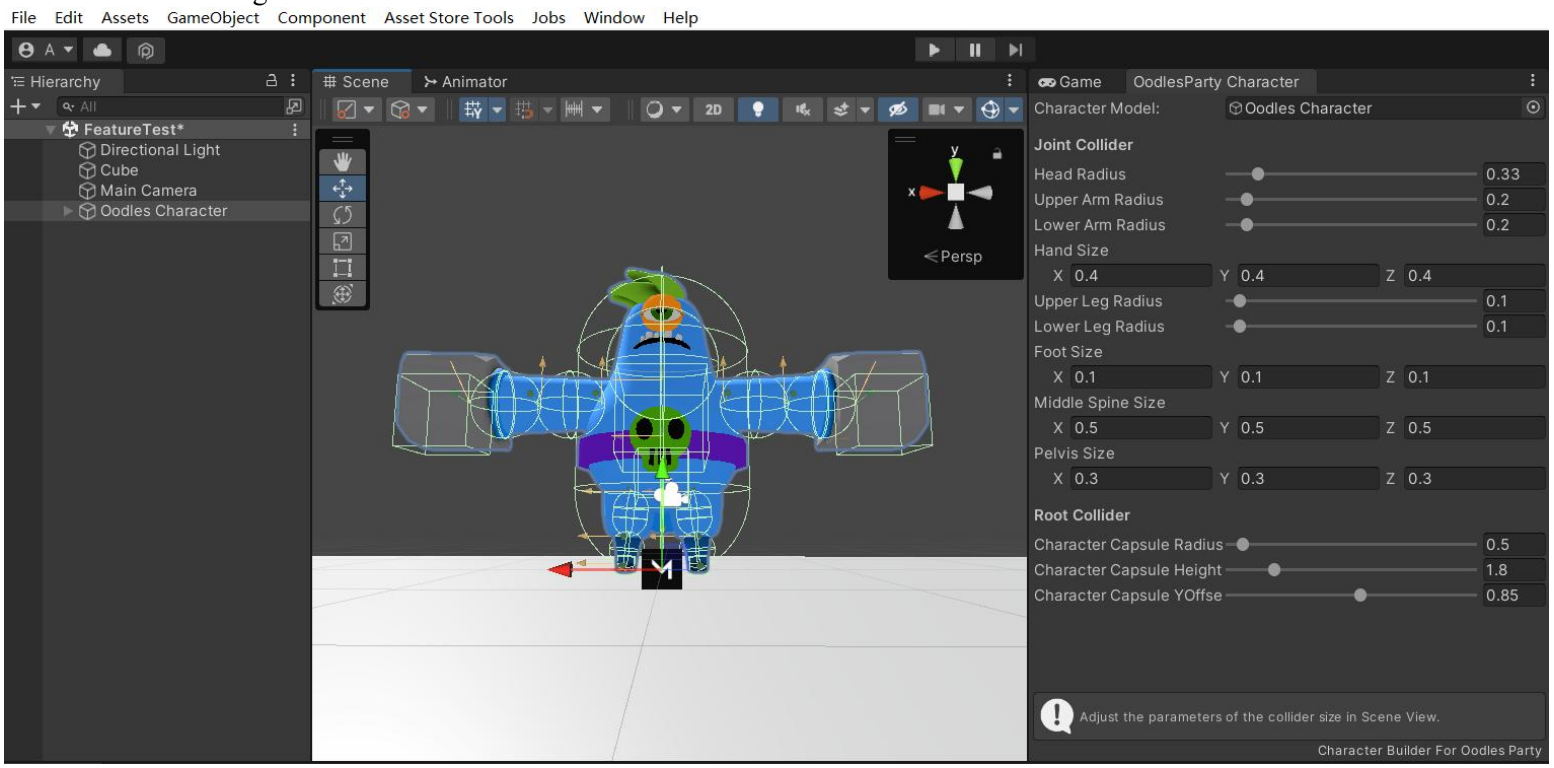


Click the 'Create Character' button, and you will see the generated physical character.



2.2 Joint Bounding Box

The default parameters may not be suitable for your model size. In this case, select the newly generated physical character, compare the size of the physics bounding box with the Scene View, and adjust the parameters on the OodlesParty Character panel to match the bounding box with the model.



Tips:

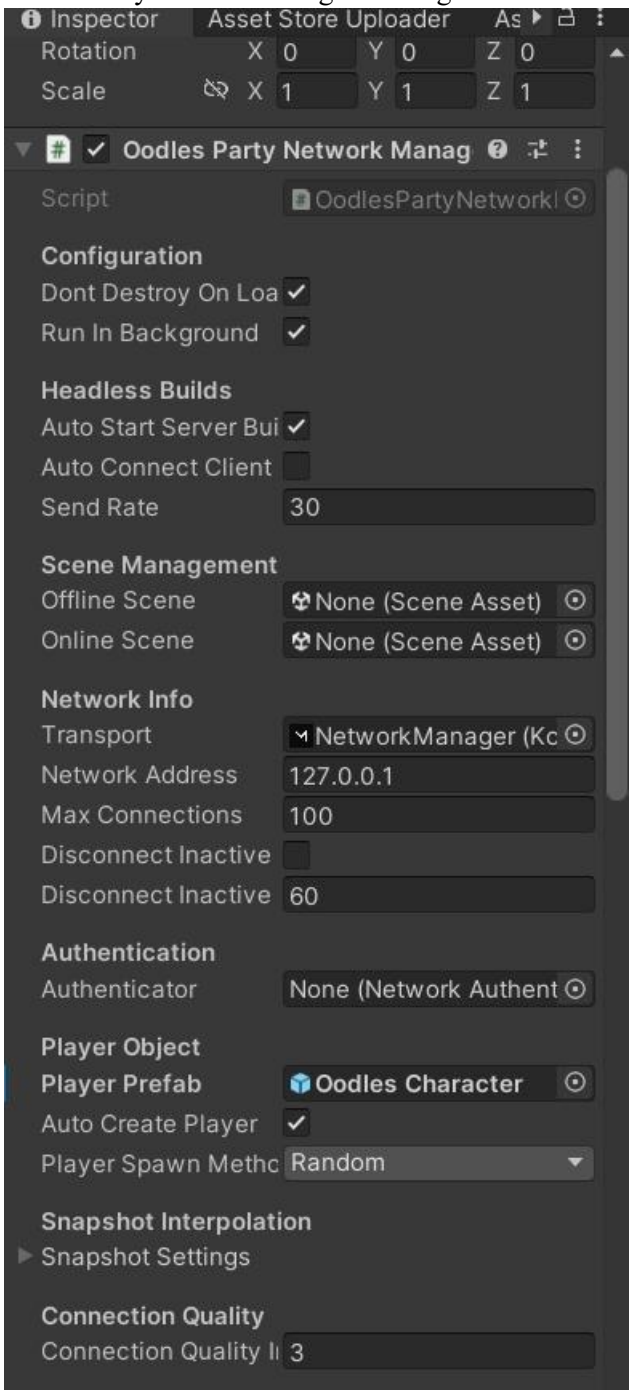
1.The Root Collider is a collider that encompasses the entire character. Please adjust the Radius and Height to precisely cover the dimensions of the character. It is recommended to set the Character Capsule Y Offset slightly less than half of the Character Capsule Height.

2.It is advisable to slightly reduce the size of the bounding boxes for the legs and feet. This can help avoid distortion in actions caused by the bounding boxes being too large and squeezing against each other.

3.Ensure that the sizes of all bounding boxes are not too small to prevent them from getting stuck during physics calculations due to mutual nesting.

2.3 Place it in the games

Give a name you like to the newly generated physical character, and then save it as a Prefab.Then place this Prefab into the OodlesPartyNetworkManager in the game scene.



Now you can start playing.

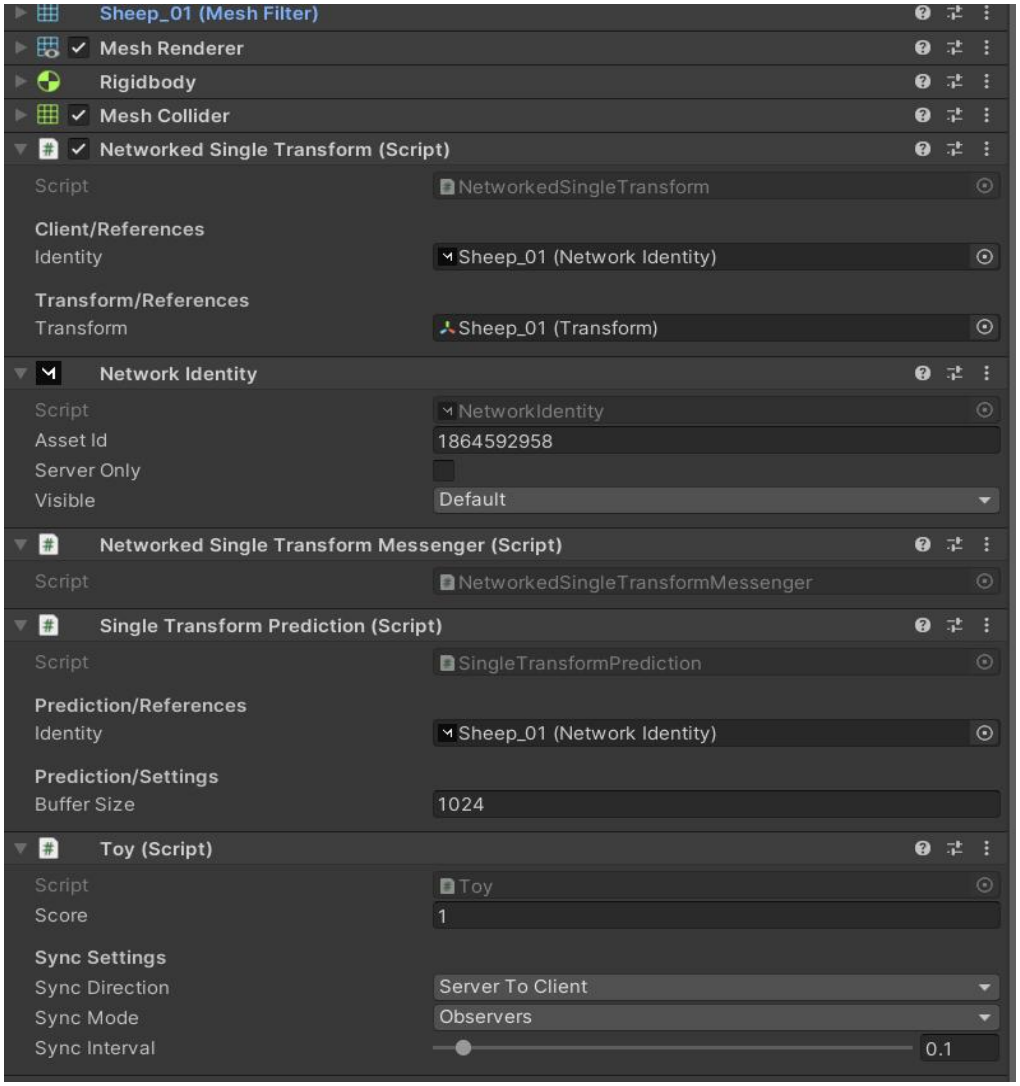
Part III: Creating Level Elements

Oodles Party offers simple but exciting game modes, and you can even create your own. Learn how to play the "Gang" mode, where red and blue teams compete to score by collecting toys. GamePlayManager.cs implements the gameplay logic, including picking up items, throwing them, and engaging in combat. The following steps will guide you on creating level elements:

3.1 CustomizingToys

Toys are scoring tools; by placing a Toy in the specified area, you can earn points for your team. To configure Toys, add RigidBody and Collider components. Then, add the following components:

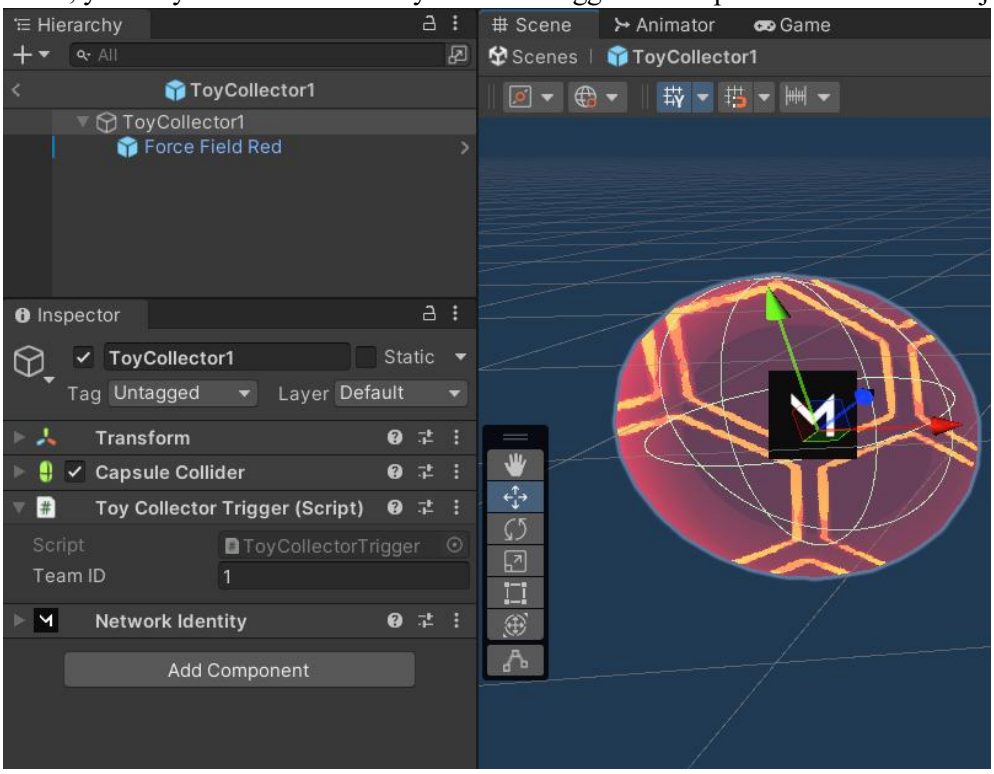
- NetworkedSingleTransform.cs
- NetworkedSingleTransformMessenger.cs
- SingleTransformPrediction.cs
- Toy.cs



You can set the score for each Toy using the Score parameter.

3.2 CustomizingToy Collectors

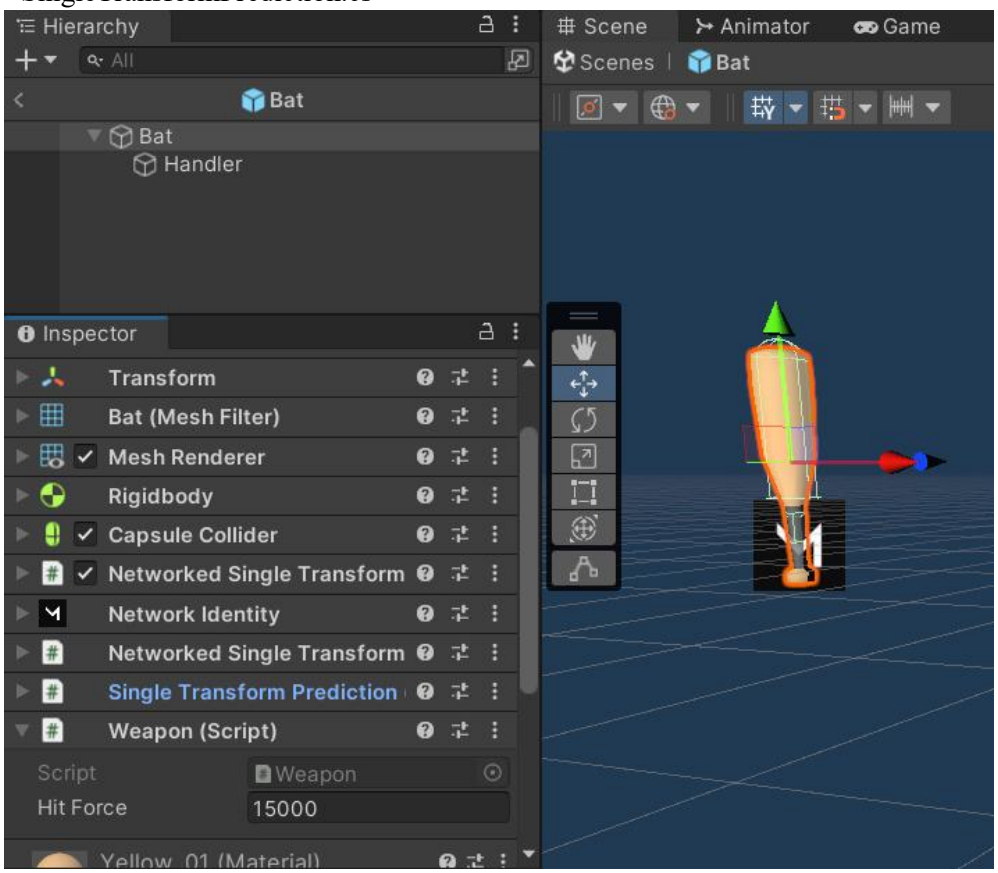
Toy Collectors are used to detect Toy collisions and assign points to the corresponding team. Since they don't need to move, you only need to add the ToyCollectorTrigger.cs component to the GameObject and set the Team ID.



3.3 Customizing Weapons

Configuring weapons is a bit more complex, as they can be picked up and moved by players. Add the following network synchronization components:

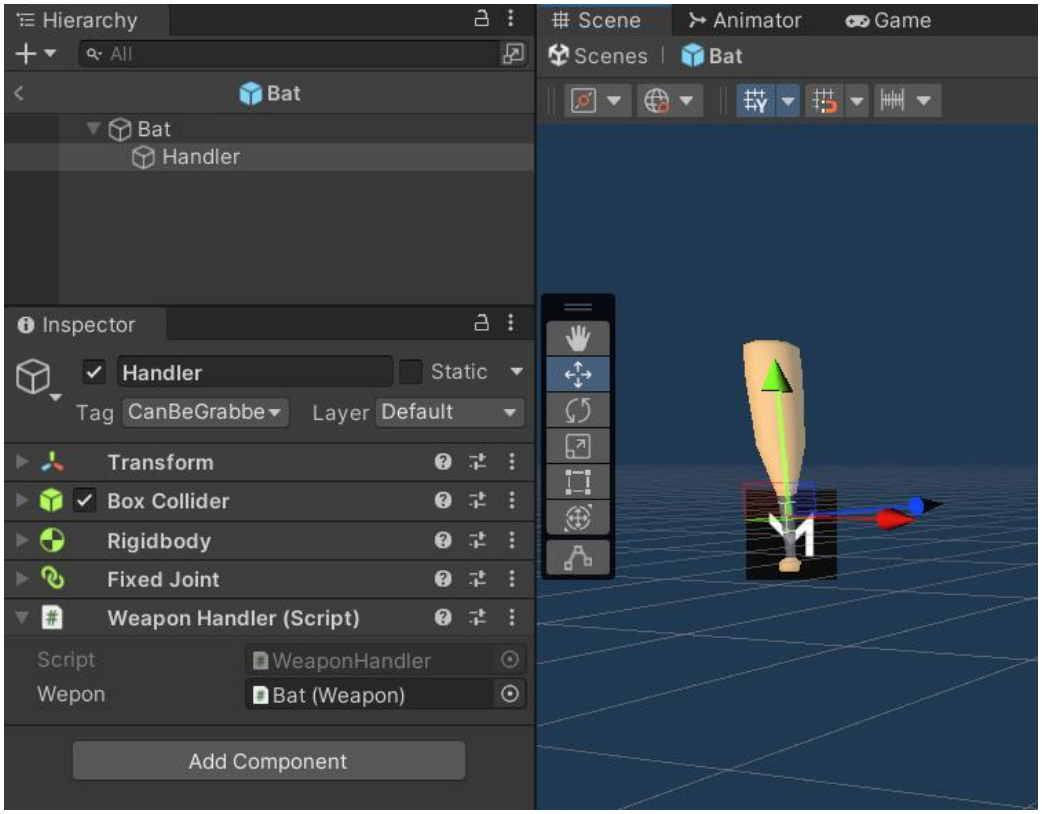
- NetworkedSingleTransform.cs
- NetworkedSingleTransformMessenger.cs
- SingleTransformPrediction.cs



Weapons require collision detection, so add a Rigidbody and Collider to the root GameObject. Then, add the Weapon.cs script component and set the Hit Force parameter to adjust the strength of the weapon.

Finally, create a child GameObject as the handle, responsible for collision detection with the player's hand. This GameObject should include:

- Rigidbody
- Collider
- Joint
- WeaponHandler.cs



3.4 Customizing Game Logic

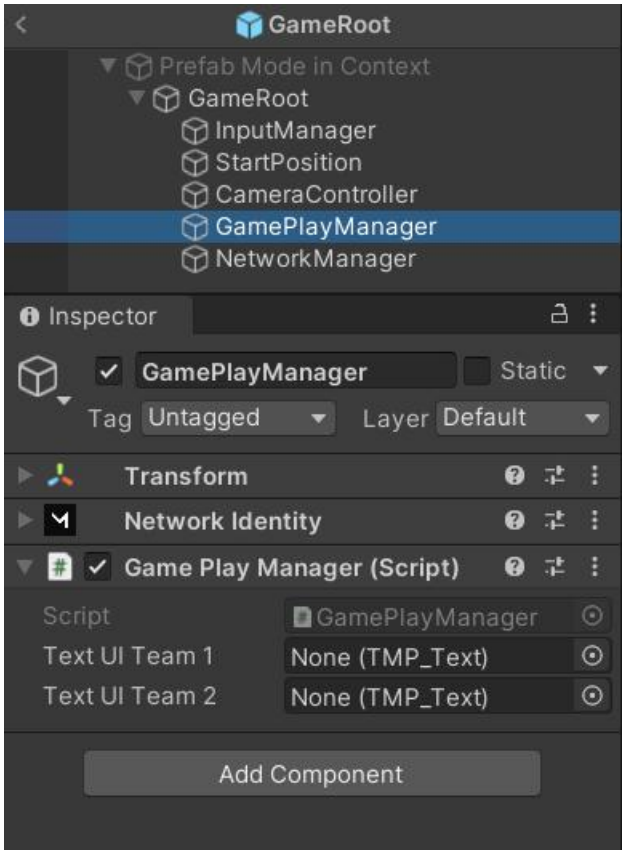
The core game logic is implemented in GamePlayManager.cs. You can extend it by implementing these interfaces:

- public bool CanGrab(PlayerController pc, GameObject obj): Check if grabbing is allowed.
- public void OnGrabSomething(PlayerController pc, GameObject obj): Handle object grab events.
- public void OnReleaseSomething(PlayerController pc, GameObject obj): Handle object release events.
- public void OnThrowSomething(PlayerController pc, GameObject obj, Vector3 dir, bool twoHands): Handle object throwing events.
- public void OnHandAttackSomething(PlayerController pc, Collision col): Handle hand attack events.
- public void OnWeaponAttackSomething(PlayerController pc, Weapon wp, Vector3 dir, GameObject obj): Handle weapon attack events.

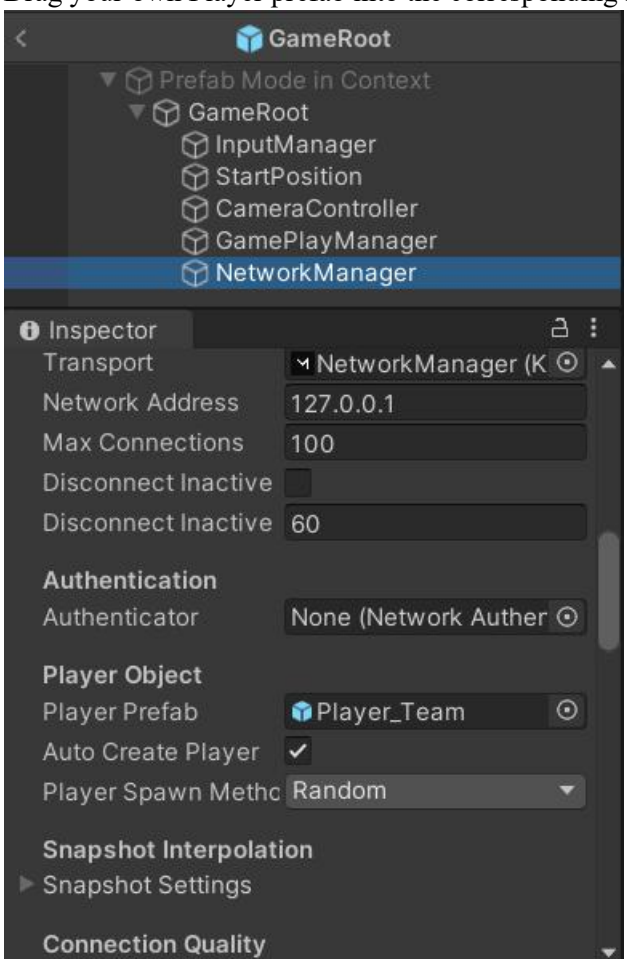
3.5 Customizing Scene

Place OodlesParty/Prefabs/Game/GameRoot.prefab in the scene.

- Configure GamePlayManager:
  Drag the UI elements used to display scores into Text UI Team 1 and Text UI Team 2 in the scene.



•Configure NetworkManager:
Drag your own Player prefab into the corresponding location.



You are now ready to expand and customize your game!