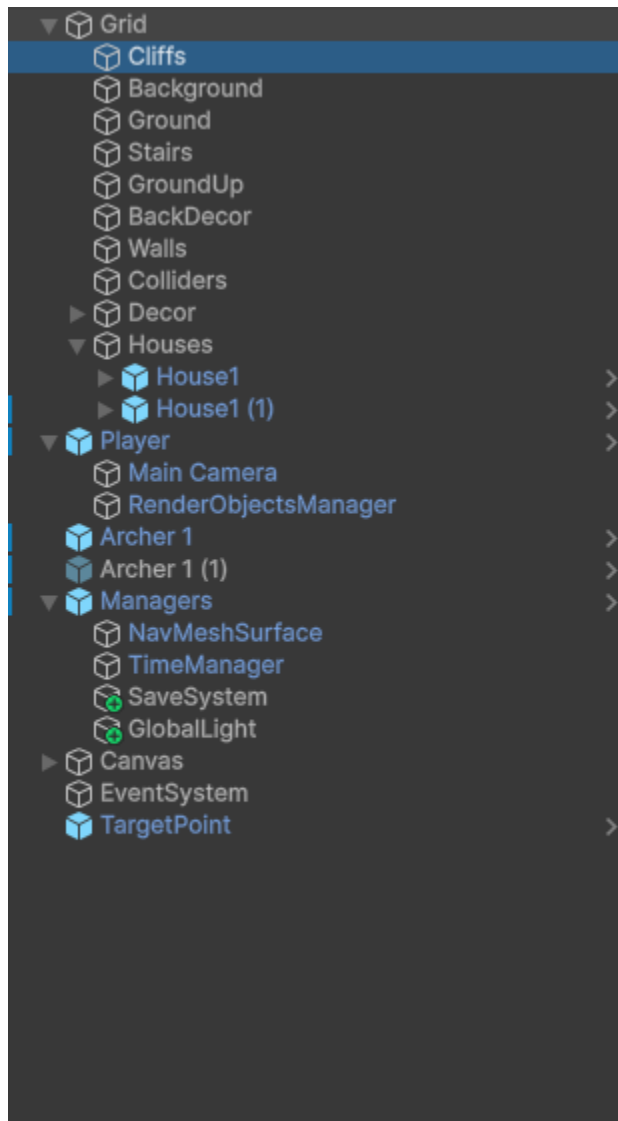# Map and Houses


1) **Cliffs** – Used to draw cliffs that act as the boundary of the playable area (i.e., the edge of the world).
2) **Background** – Used for drawing water or other tiles that appear *below* the main ground level, creating a depth effect.
3) **Ground** – Used for regular ground tiles.
4) **Stairs** – Used for placing stairs that the player can automatically ascend.
5) **GroundUp** – Similar logic to **Background**, but used for elevation effects above ground.
6) **BackDecor** – Decorative objects that are rendered *behind* the player.
7) **Walls** – Impassable wall tiles.
8) **Colliders** – Invisible walls used to block movement.
9) **Decor** – Decorative elements rendered *above* the player.

a) You can also add animated prefab child objects.

10) **Houses** – Acts as a container for buildings.
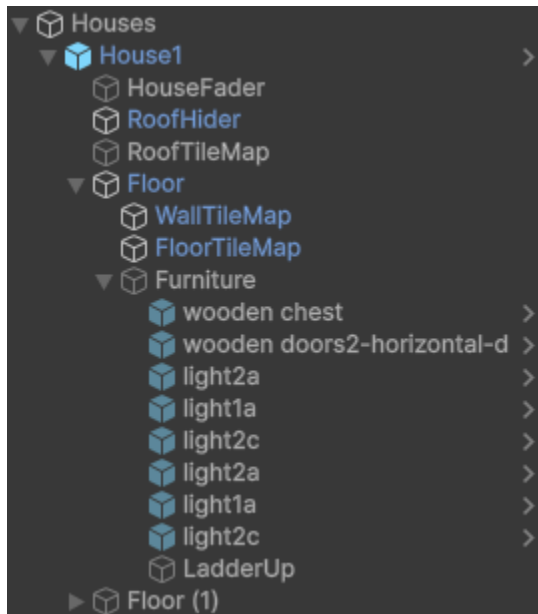
a) **RoofTileMap** – Represents the house roof, which hides automatically when the player enters the house.

b) **Floor** – A container for each floor of the house, containing:

i) **WallTileMap** – House walls.

ii) **FloorTileMap** – House floor.

iii) **Furniture** – For decorations (you can also add animated prefab children).

iv) For multi-story buildings, each floor must include a **Ladder** to allow the player to move between floors.
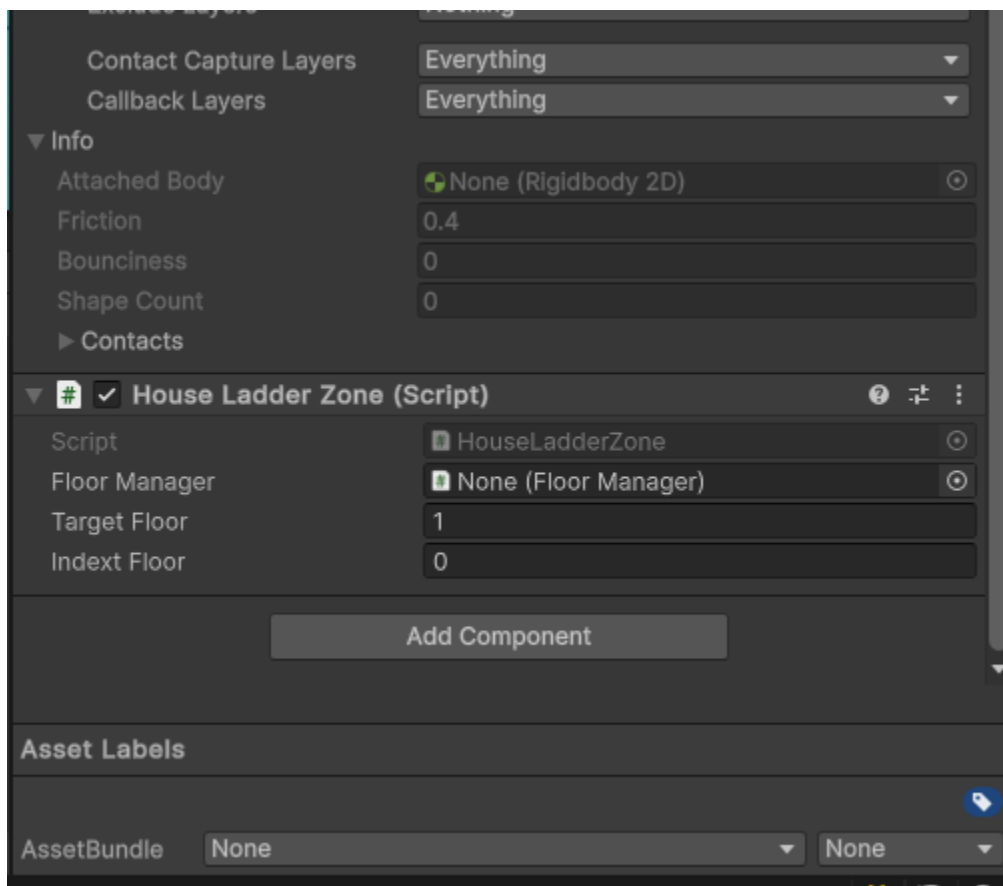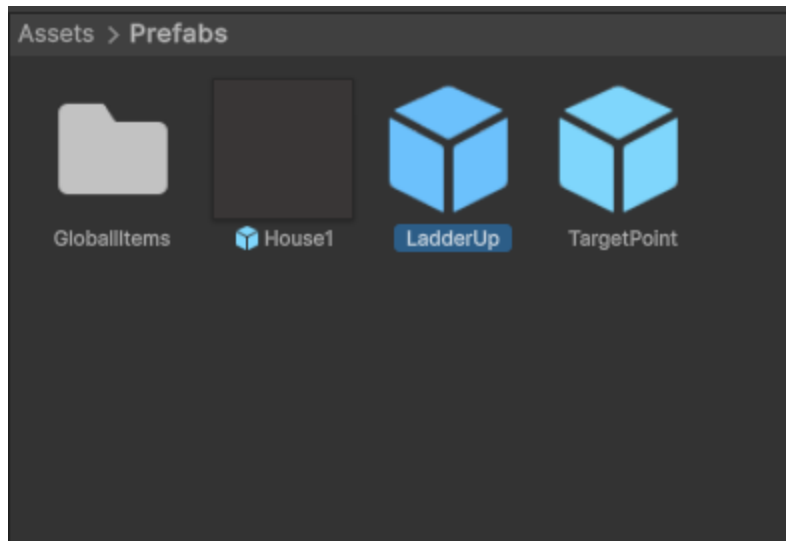
Each **Ladder** must have a `HouseLadderZone` component with the following settings:

- **FloorManager** – The house this ladder belongs to.

- **TargetFloor** – The floor this ladder leads to.

- **IndexFloor** – The current floor the ladder is on (starting from 0).
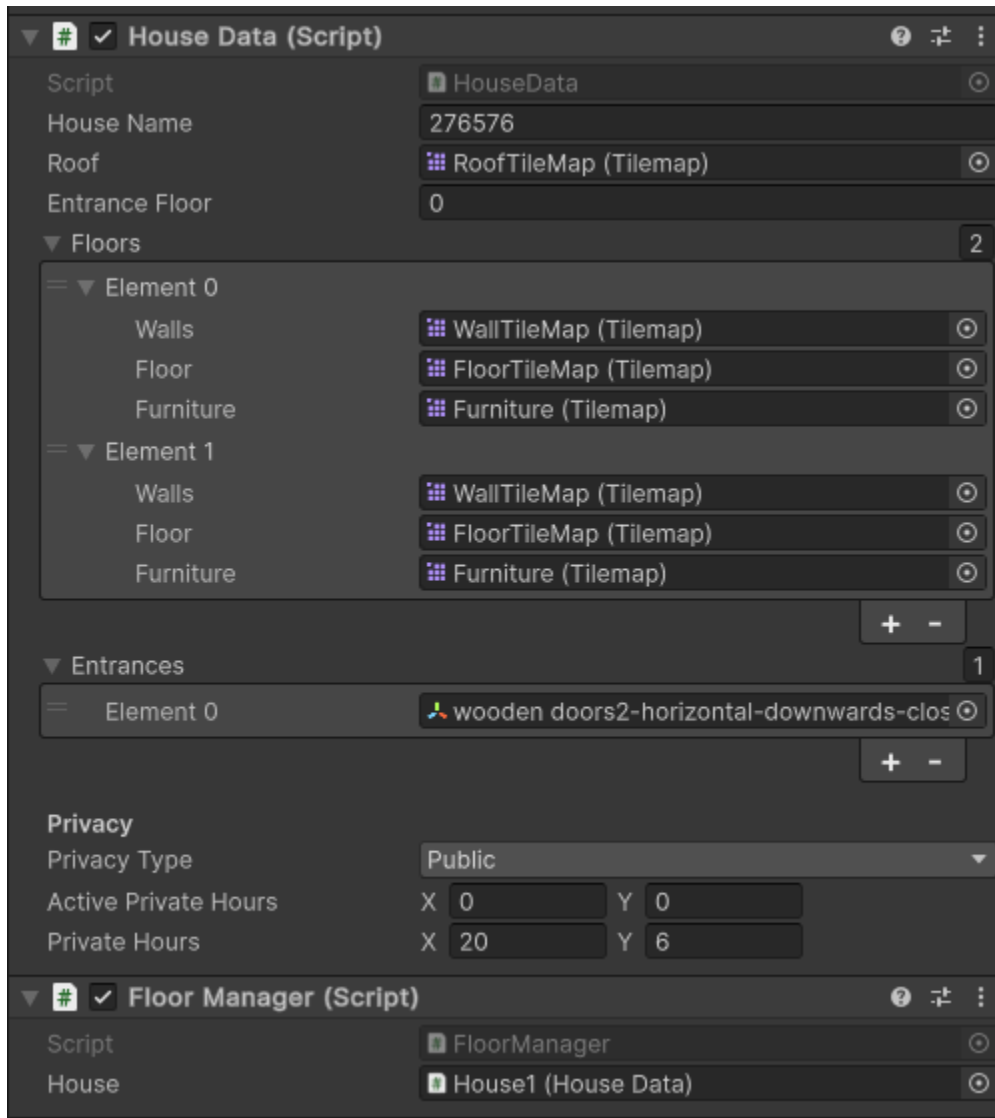
Each house has a **HouseData** object containing:

- **Roof** – The roof object.

- **EntranceFloor** – The index of the main entrance floor (e.g., set to 1 if the ground floor is considered a basement).

- **Floors** – A collection of all the house's floors.

- **Entrances** – Entry points into the house.

Privacy Settings

- Used to configure whether a house is public or private.

- `PrivacyType`: `Public`, `Private`, or `PrivateScheduled`.

- `PrivateHours`: The time range during which the house is considered private (used only with `PrivateScheduled`).

You can use prefabs from the **Prefabs** folder, where most settings are already configured.
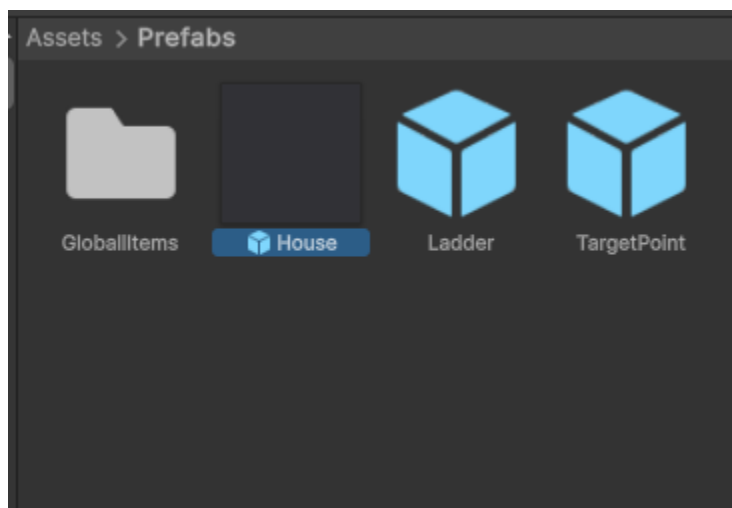
NPCs and Schedules

NPC behavior is controlled by the `NPCController` script.
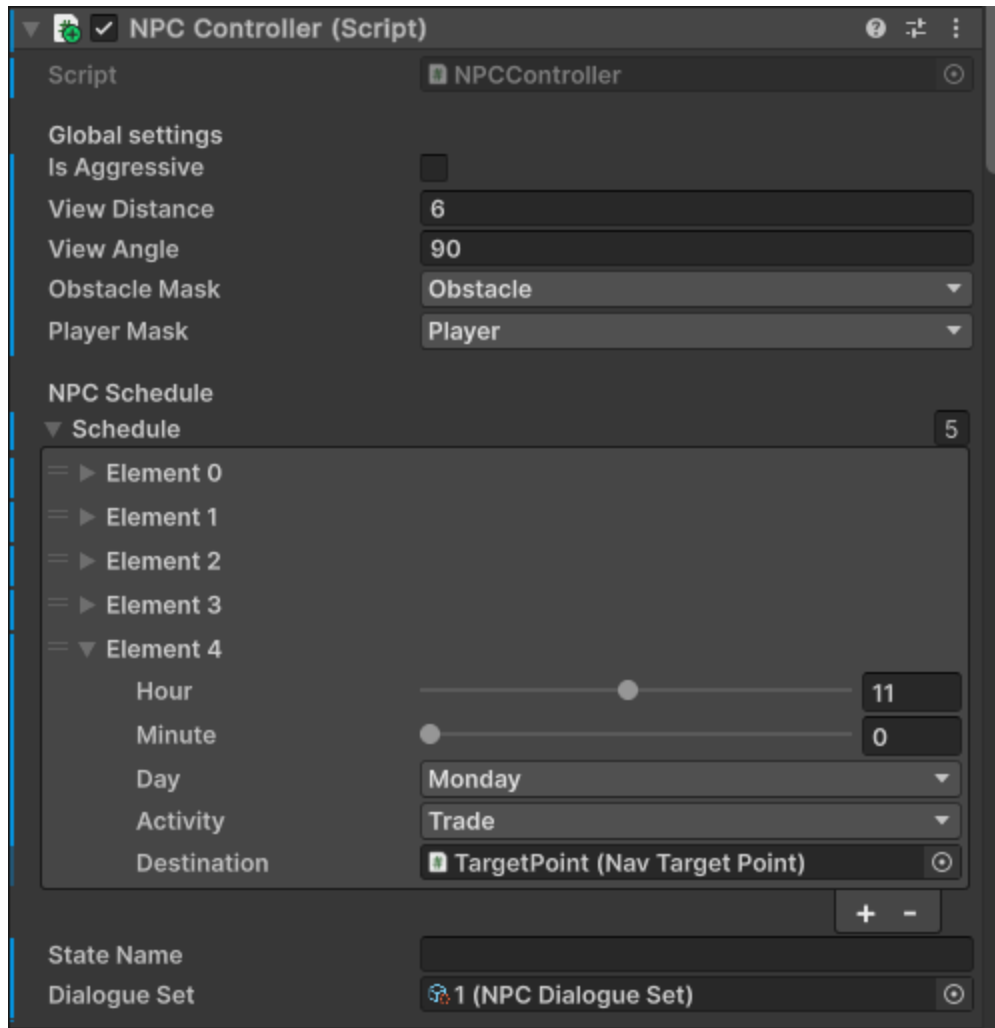
a)
b) **isAggressive** – If enabled, the NPC will chase the player as soon as they are spotted. (This will likely be reworked in the "Factions & Reputation" phase.)
c) **View Distance** – NPC vision radius.
d) **View Angle** – NPC field of view angle.
e) **Obstacle & Player Mask** – Generally do not need to be changed.

Schedule – A list of the NPC's daily activities:

- **Hour/Minute** – When the event starts.
- **Day** – Day of the week. If events repeat daily, duplicate them for each day.
- **Activity** – The type of action performed at that time:
  - `Sleep` – NPC is sleeping.
  - `Work` – NPC is working.
  - `Idle` – NPC stands in place.
  - `Wander` – NPC roams randomly.

- ○ `Trade` – NPC is available for trading.
- ○ `Guard` – NPC guards a location (must define the position).
- ○ `Patrol` – Like Guard, but moves around a route.
- ○ `Hide` – NPC is hidden and cannot be interacted with.
- ○ `Hunt` – NPC roams and will chase the player upon spotting them.
- ○ `Chill` – NPC casually wanders near a point within a small radius.
- **TargetPoint** – An invisible dummy point used for pathfinding (created as a prefab for easier reuse).
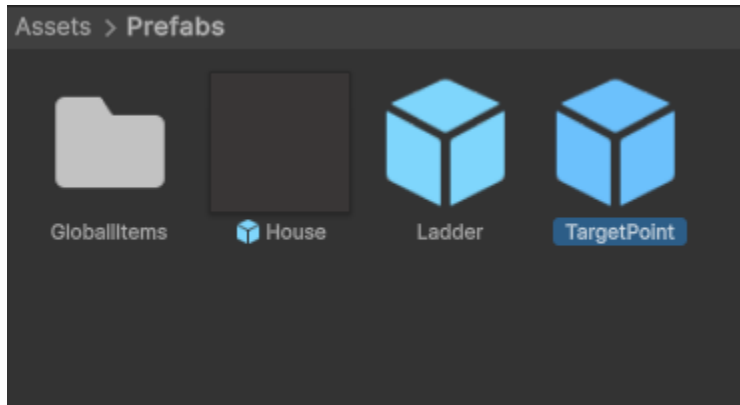
Each point has the following additional data if placed inside a house:
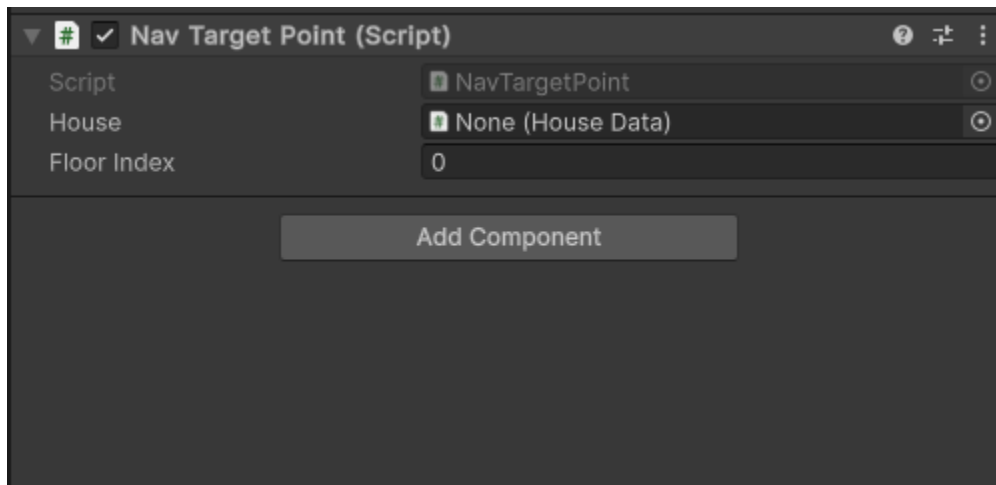
House – The house it belongs to.

FloorIndex – Which floor the point is on.
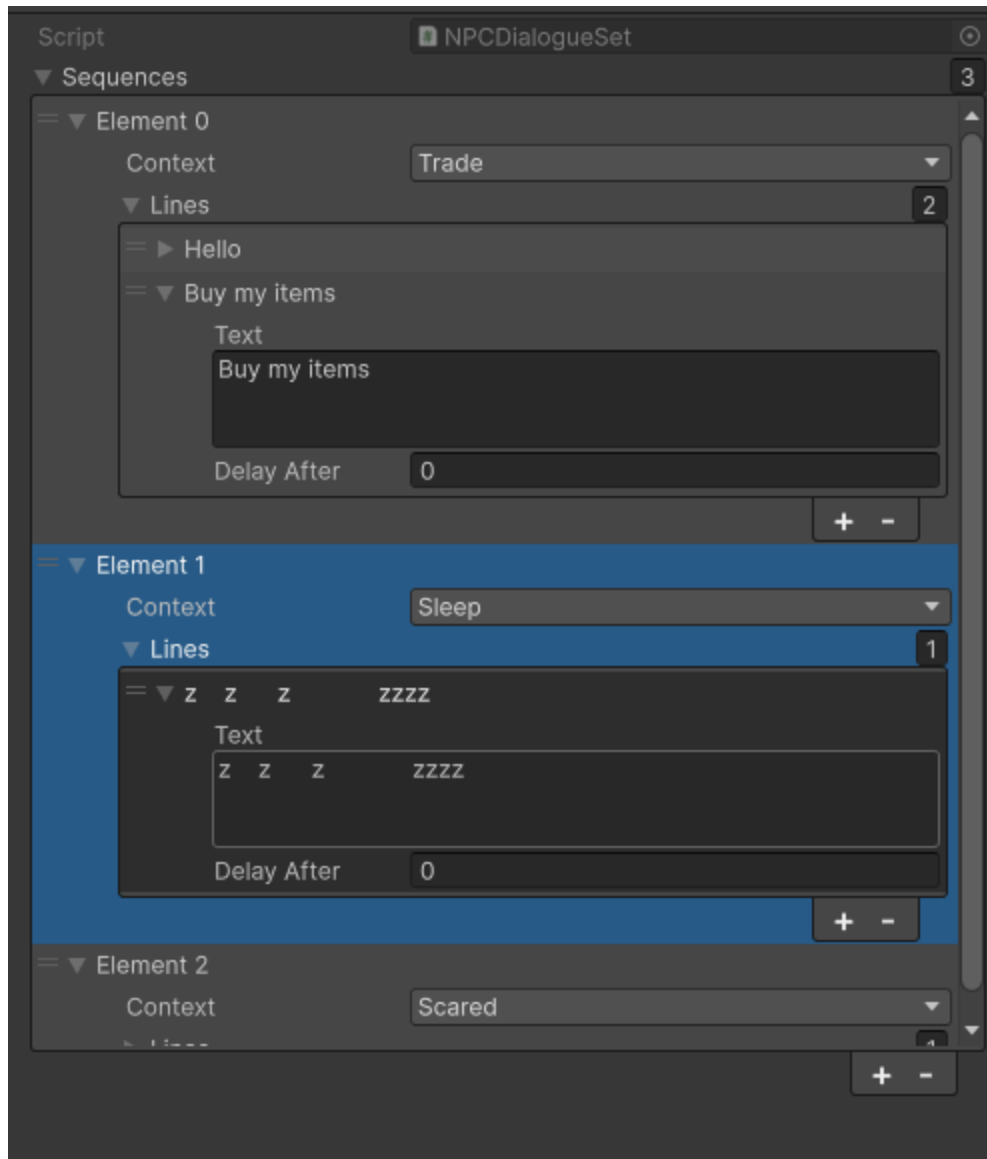(If the point is outside, these fields can be ignored.)

StateName – Used for debugging.

DialogueSet – A set of phrases that the NPC says periodically depending on their current state. For example, if trading, they might say "Come buy something!"
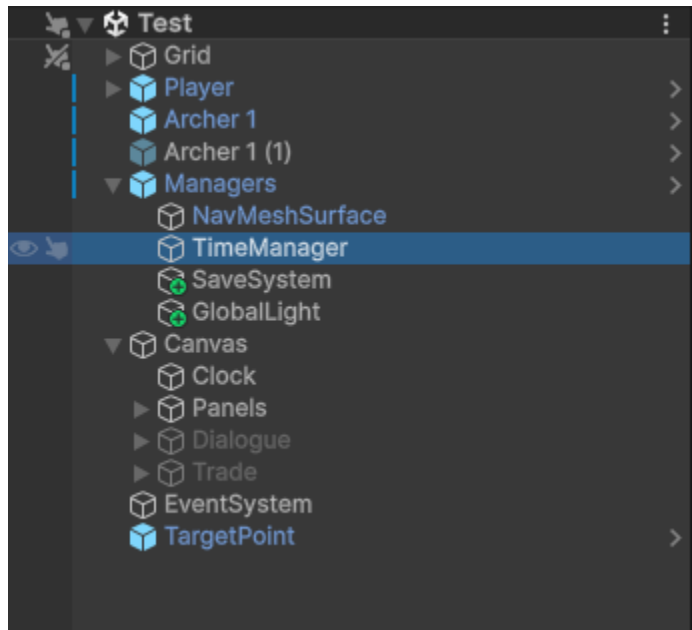
To create a dialogue set:
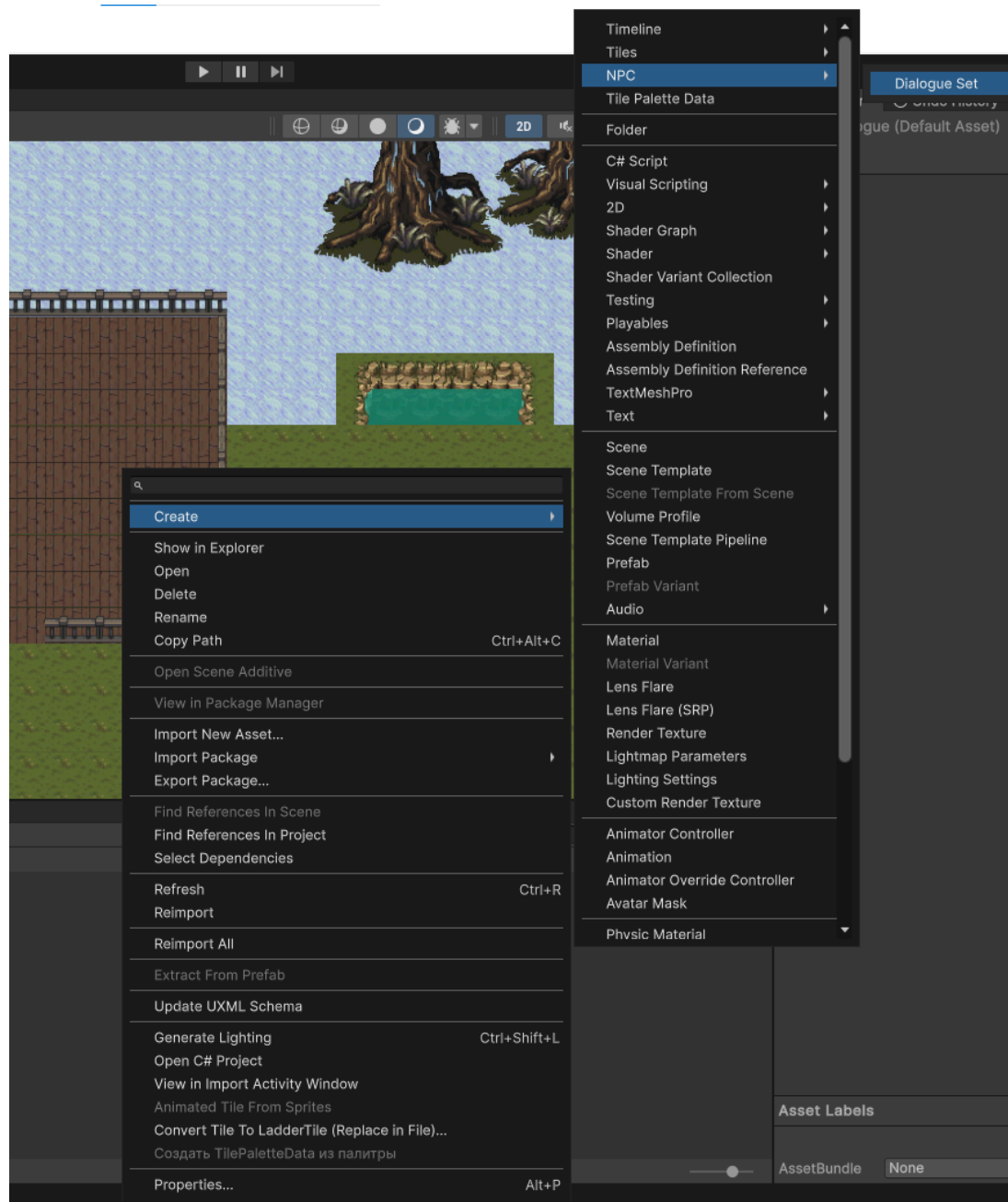
- **Context** – The state in which the lines are used. Matches the NPC activity types:
  - `Idle` – Also used for `Wander` and `Chill`.
  - `Work`
  - `Trade`
  - `Guard` – Also used for `Patrol`.
  - `Sleep`
  - `Hunt`
  - `Scared`
- **Lines** – The lines of dialogue spoken in that state.

This field is optional – you can leave it empty or reuse sets between NPCs.
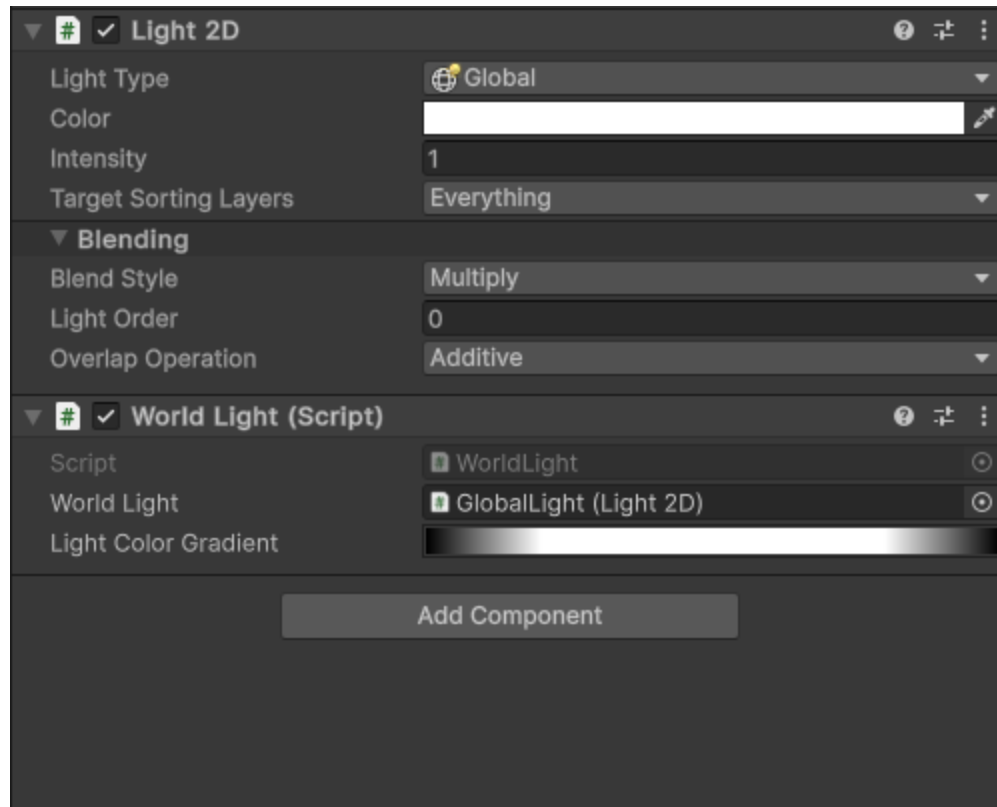
Global settings

Under **Managers**, the `TimerManager` controls in-game time.
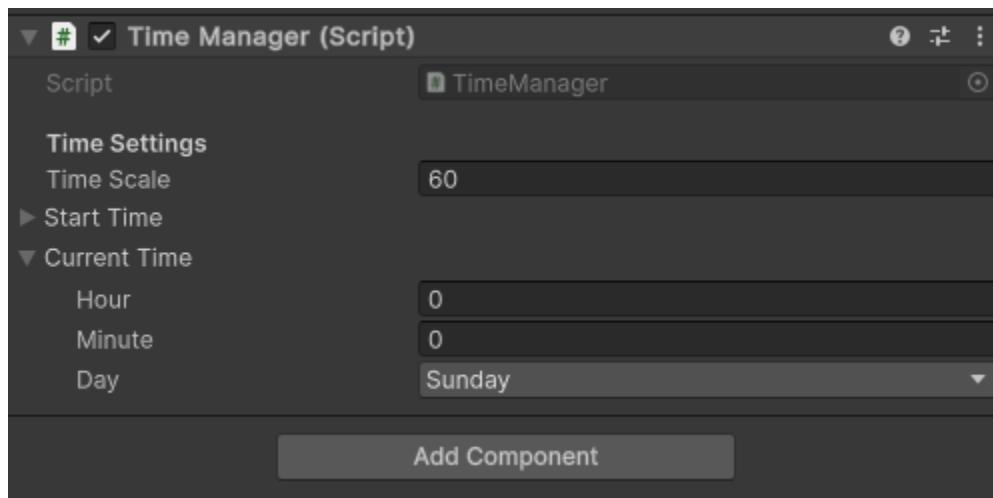
Time Scale – Determines how fast in-game time passes compared to real time (e.g., how many real seconds equal one in-game second).

The **GlobalLight** object controls lighting.

Light Color Gradient – A gradient for smooth lighting transitions over time (0% = 00:00, 50% = 12:00, 100% = 24:00).
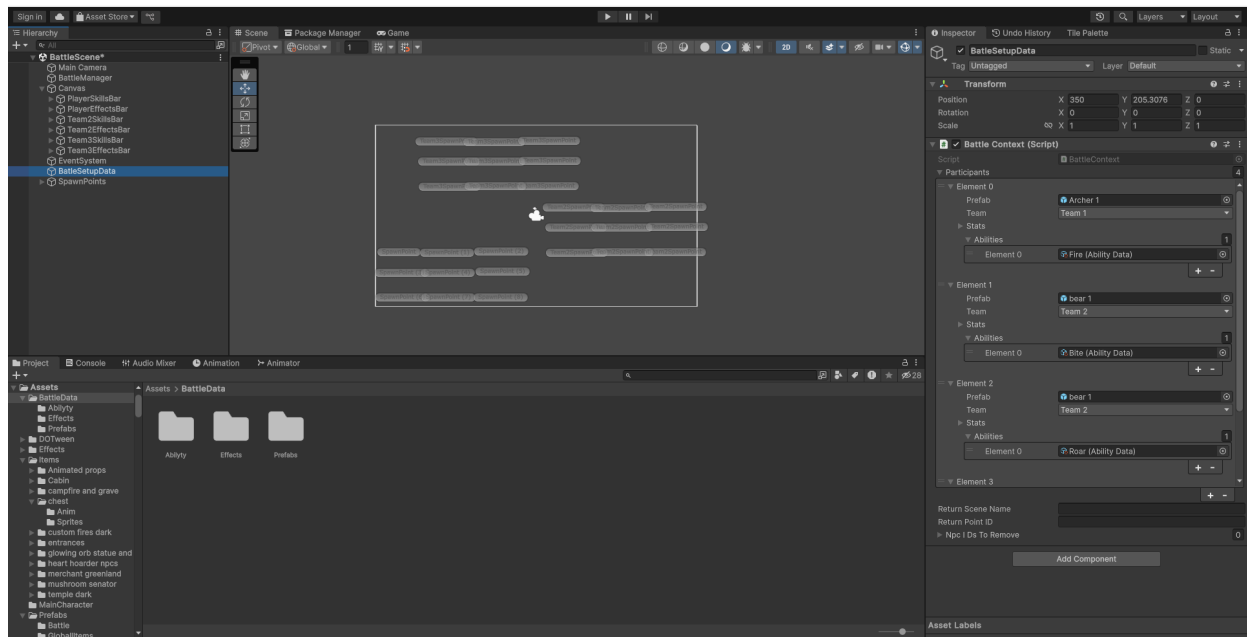
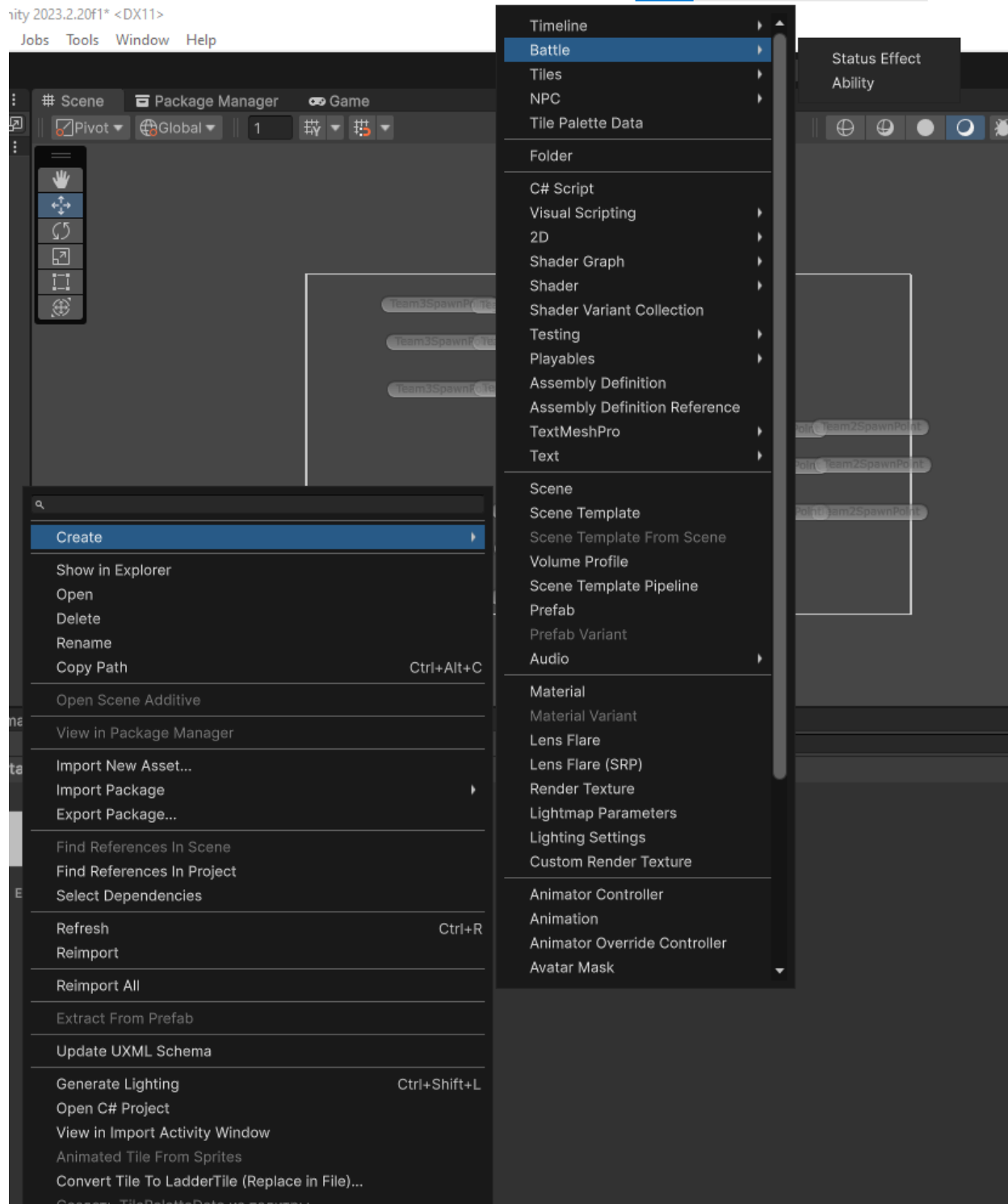To set up light sources (lamps, etc.), add a `Light2D` component and configure its glow.
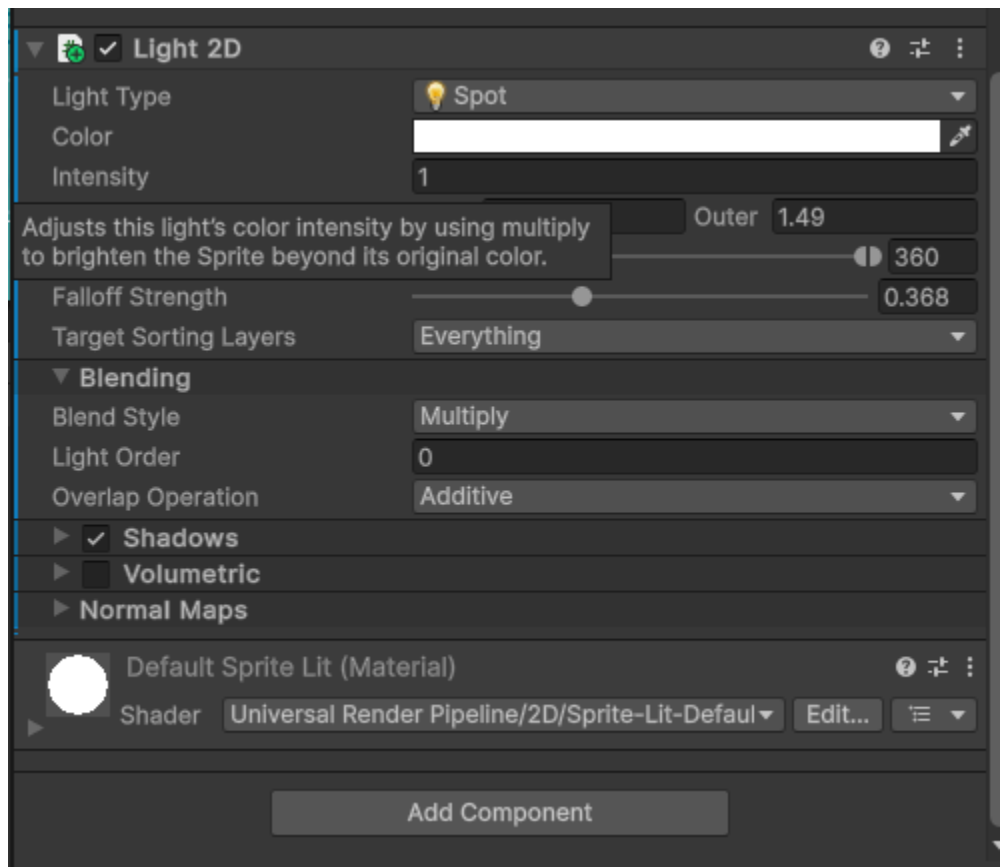
Battle System

To prepare the battlefield, it is enough to select the character that will be created at the beginning of the battle from the BattleData/Prefabs folder, select which team he will be on and

add the abilities he will have

To create abilities or effects you need to select the appropriate menu:

each ability will have its own parameters:

icon - the ability icon that will be displayed on the UI panel
In the Effects tab, you can add effects that will be superimposed by this ability.
If the ability in the Summon Prefab field is specified from the character's BattleData/Prefabs
folder, then when using the ability, that character will be summoned (if there is free space for it)
Animation Triger If the ability for characters has a special animation, then the animation name
must be specified differently (see below for how to set up animations correctly)

Script	StatusEffect
Effect Name	
Icon	Icon34_0
Type	Affliction
Duration	3
Damage Per Tick	5
Weakness To Damage Type	Water
Weakness Multiplier	0.5
Prevent Action	☐
Prevent Magic	☐
Prevent Physical	☐
Silence	☐
Sleep	☐
Is Feared	☐
Force Ally Targeting	☐
Random Targeting	☐
Block Next Ability	☐
Double Magic	☐
Double Physical	☐
Regen HP	☐
Regen Amount	0
Cast Speed Multiplier	1
Magic Damage Modifier	1
Physical Damage Modifier	1
Resistance Modifier	1
▼ Combo Effects	1

▼ Element 0
Other Effect	None (Status Effect)
Resulting Effect	None (Status Effect)
Remove Originals	☐

| + | - |

| Trigger | On Tick |

combo effects is used for combinations of effects, in the Other field it is indicated which effect is combined with and in the Resulting field the effect that we get at the output is indicated
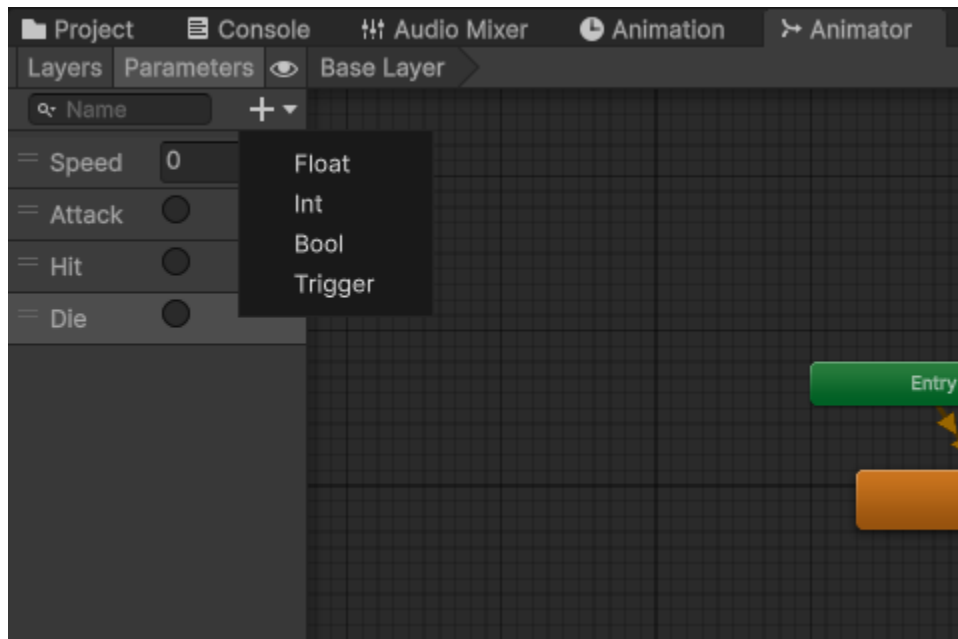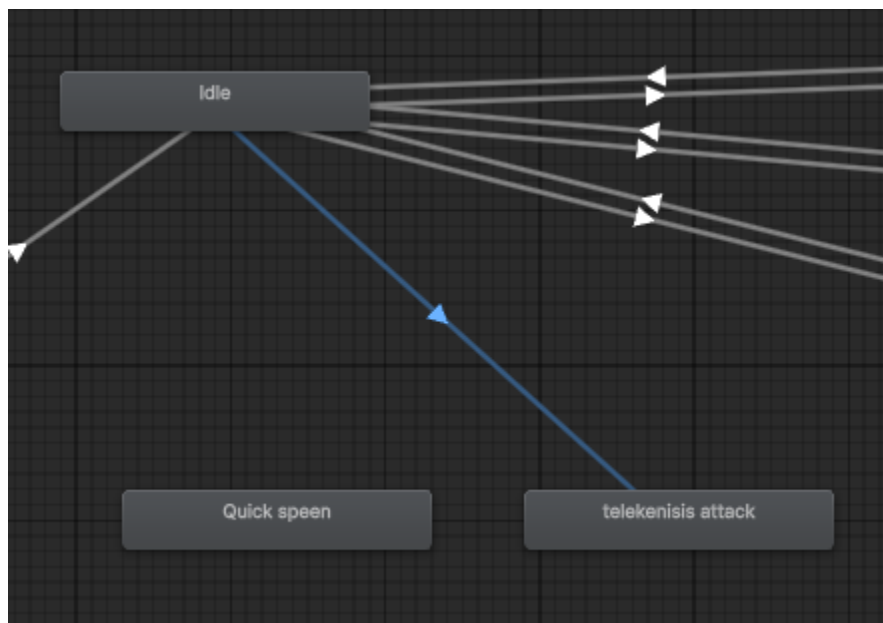
Trigger - remote field

Animations

each prefab has an Animator
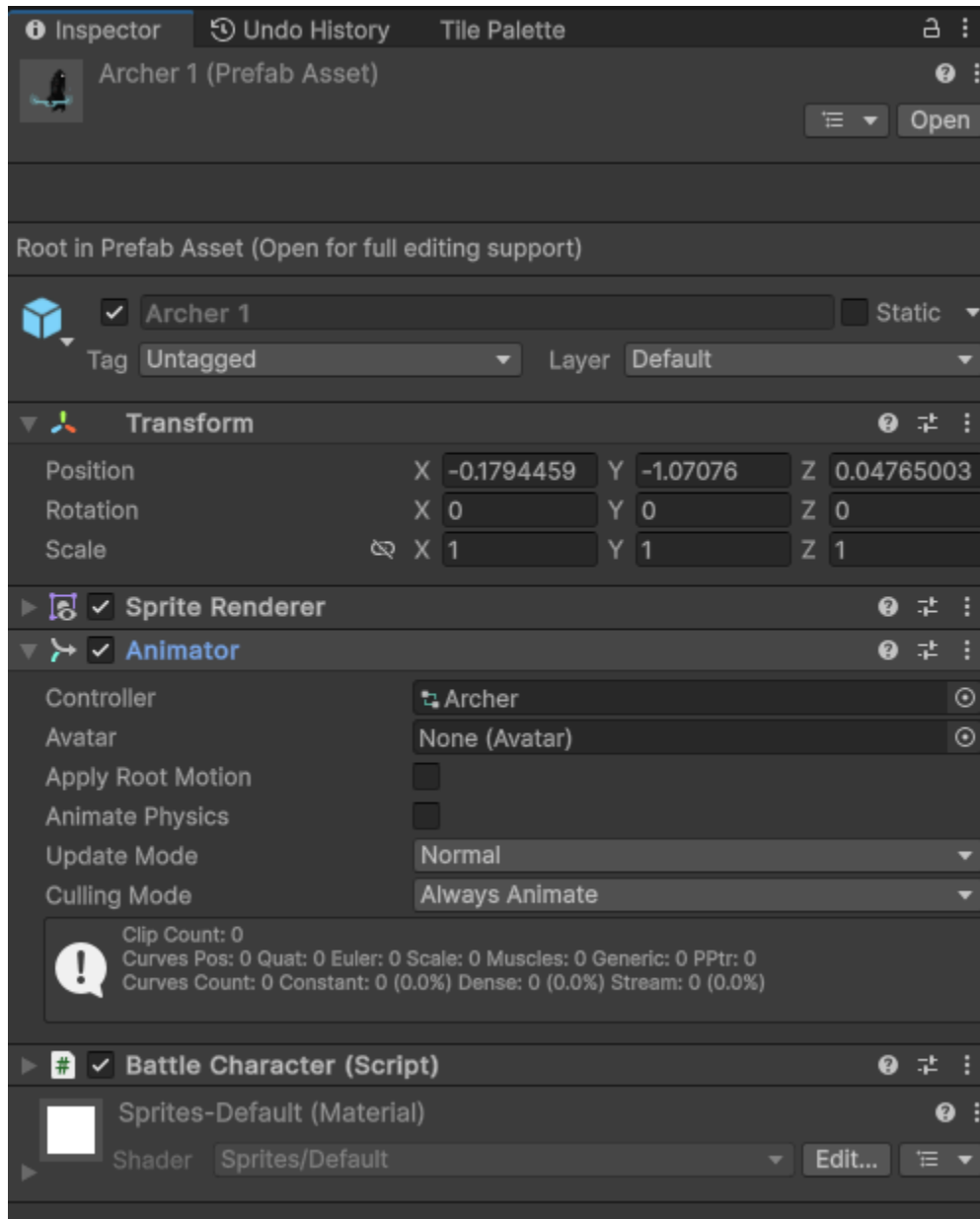
Simply go to the Animatior window or double-click in the Controller field.

create a new trigger and name it whatever you want the transition to animation to be called, for example "Telekenisis"

create a transition from the idle animation to the desired animation,
click on the arrow, remove Has Exit Time
and in Conditions specify the name of the newly created trigger

Idle -> telekenisis attack
1 AnimatorTransitionBase

Transitions	Solo	Mute
Idle -> telekenisis attack	☐	☐

Idle -> telekenisis attack

Has Exit Time ☐

▶ Settings

| 0:00 | 0:05 | 0:10 | :15 | 0:20 | 0:25 | 1:00 | 1:05 | 1:10 | 1:15 | 1:20 | 1:25 |

Idle   Idle
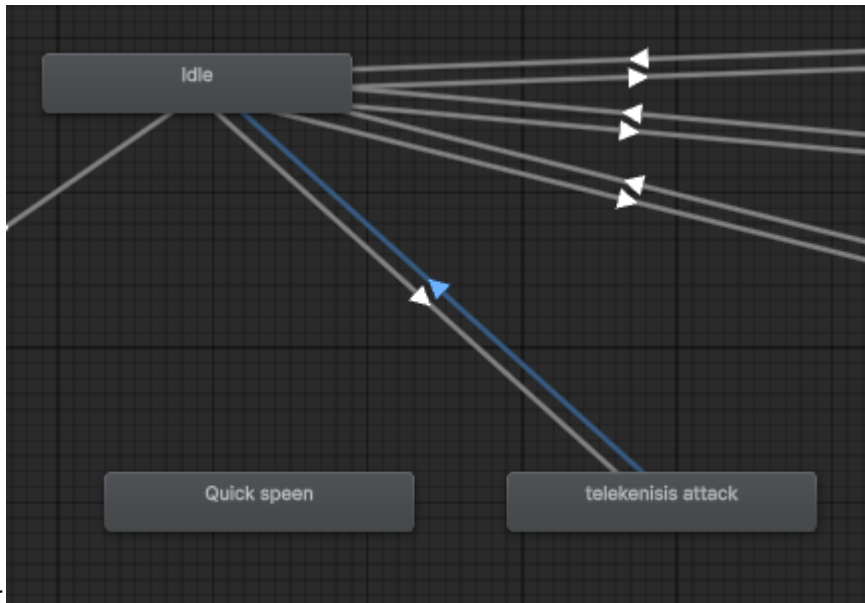
telekenisis attack

Conditions
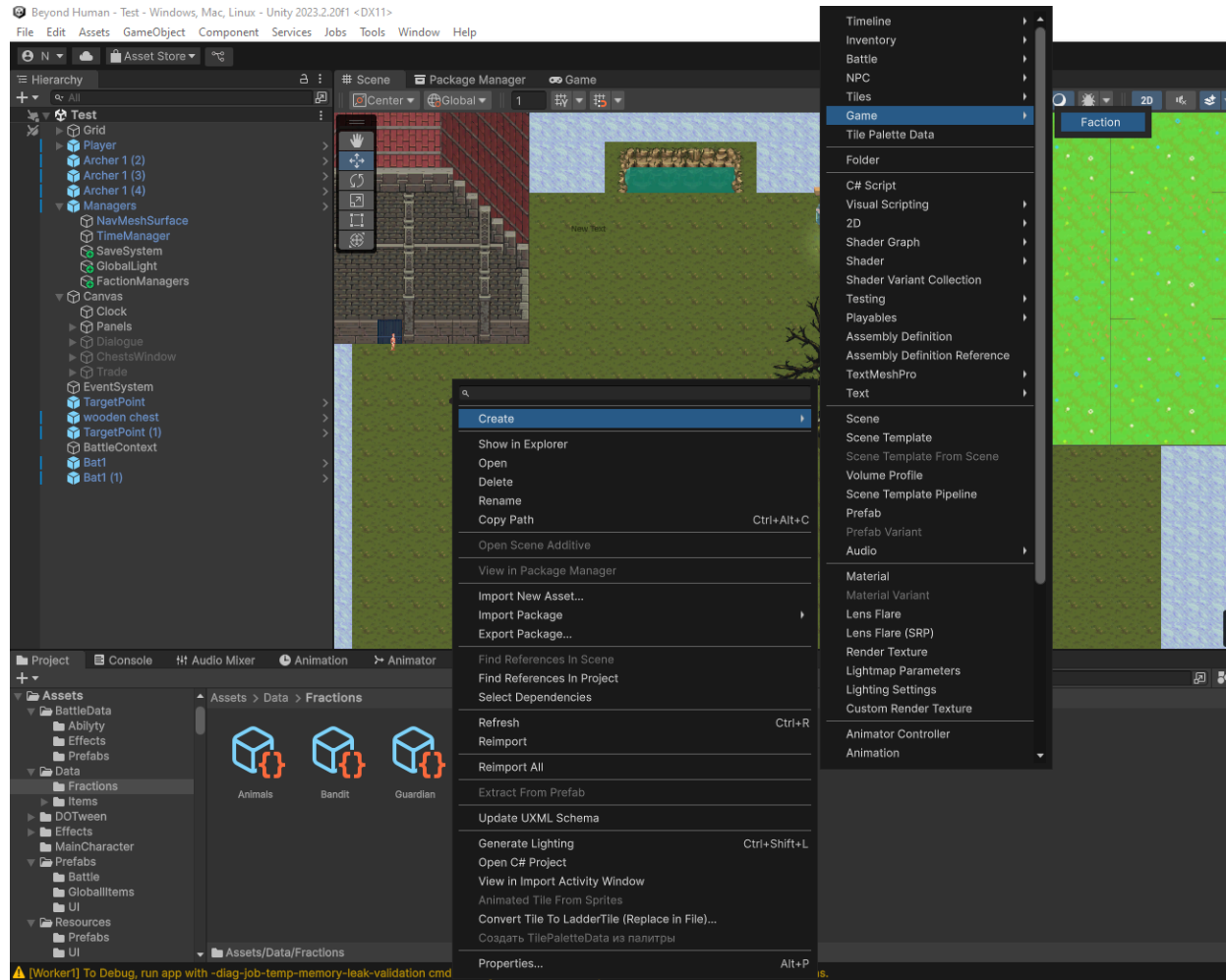
= Telekenisis ▼

+ −

and you definitely need to remember to make a return arrow, but you don't need to do anything

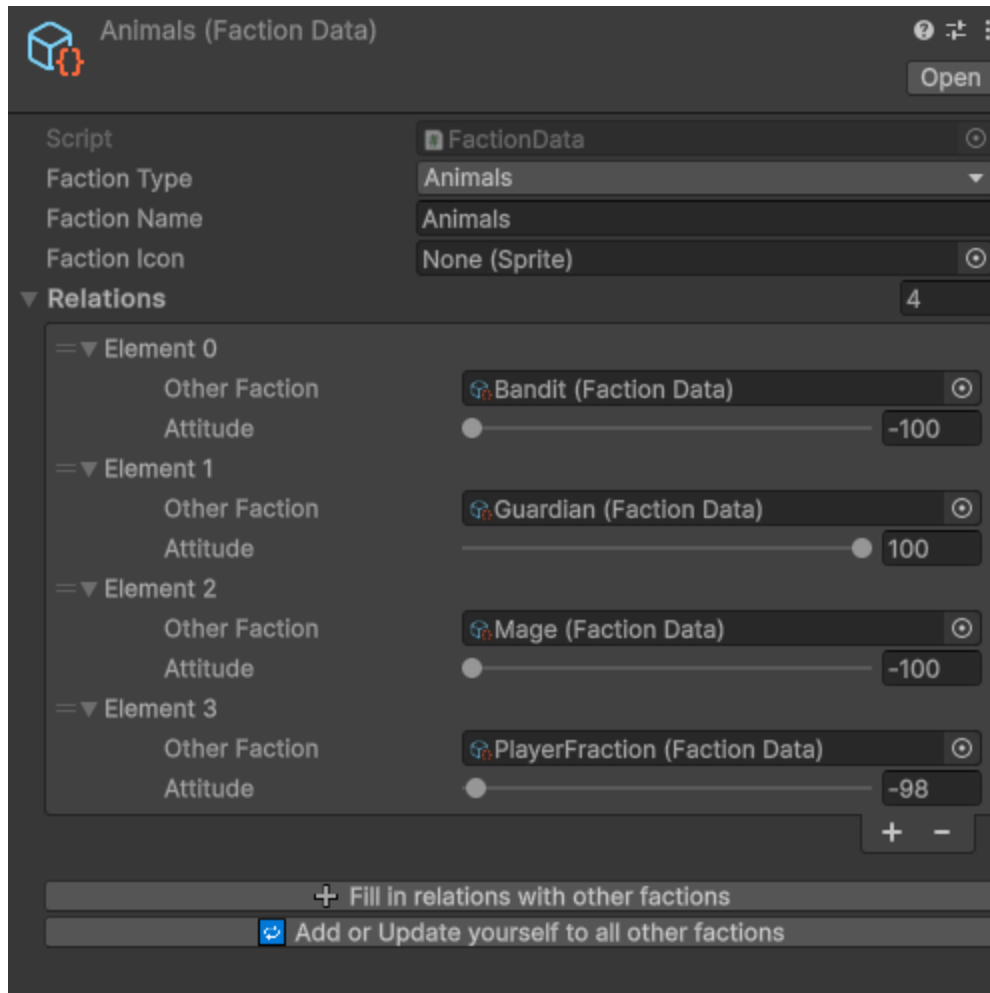

with it

Factions and battle start

Create Factions

To create a new faction, I recommend doing it inside the folder:
Assets/Data/Factions
Choose: **Create → Game → Faction**

This is what the faction creation window looks like — here you define the type, name, icon, and relationships with other factions.
Each faction must be manually added and assigned a relationship (e.g., Friendly, Neutral, Hostile) with the others.

IMPORTANT: The relationship values must be mirrored — if Faction A is Friendly to Faction B, then Faction B must also be Friendly to Faction A.

To make this easier, I've added two buttons:

● The first button goes through all factions and sets the current faction's relationships based on how others relate to it.

● The second button adds this faction to all others (if it's not already listed), letting you quickly populate missing entries.
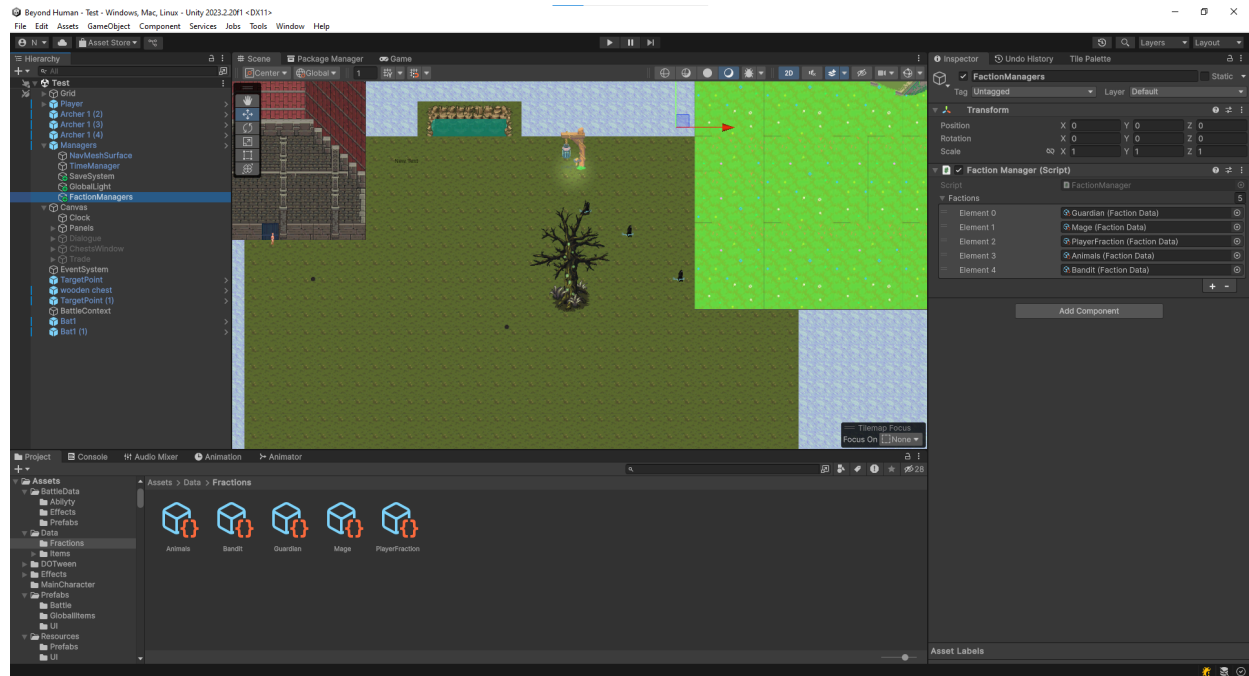
To ensure factions work properly in the game, make sure all created factions are registered in the **FactionManager**.
This is required for proper battle preparation and faction logic.

settings to prepare for battle

NPC Controller (Script)

Script	NPCController

Global settings
Is Aggressive	☐
View Distance	6
View Angle	360
Obstacle Mask	Obstacle ▼
Player Mask	Player ▼

NPC Schedule
▶ Schedule	7
State Name	

StatsForBattle
▼ Battle Participant Data

Prefab	Bat1 ◎

▼ Stats

Max HP	100
Current HP	100
Current Shield	0
Air Resistance	0
Water Resistance	0
Fire Resistance	0
Earth Resistance	0
Electric Resistance	0
Ice Resistance	0
Poison Resistance	0
Blunt Resistance	0
Piercing Resistance	0
Curse Resistance	0
Holy Resistance	0
Air Damage	0
Water Damage	0
Fire Damage	0
Earth Damage	0
Electric Damage	0
Ice Damage	0
Poison Damage	0
Blunt Damage	0
Piercing Damage	0
Curse Damage	0
Holy Damage	0

▼ Abilities 1

Element 0	Bite (Ability Data) ◎

+ -

Here's how the updated **NPCController** looks now:

- **Prefab** – a reference to the prefab used during battle scenes

- **Stats** – starting stats such as HP at the beginning of the battle and base additional resistance and damage

- **Abilities** – the list of skills/abilities this NPC will use during combat

- **IsDead (checkbox)** – indicates whether the NPC is dead; if true, the NPC will not appear in the scene on the next load (not recommended to modify manually)

- **Faction Type** – assigns the NPC to a specific faction

- **NPC ID** – a unique identifier automatically assigned when creating an NPC; used for saving/loading data

```
            //add more reasons as needed
    }
    public enum FactionType
    {
        None,
        Player,
        Guards,
        Bandits,
        Mages,
        Undead,
        Animals
        // Add more factions as needed
    }
}
```
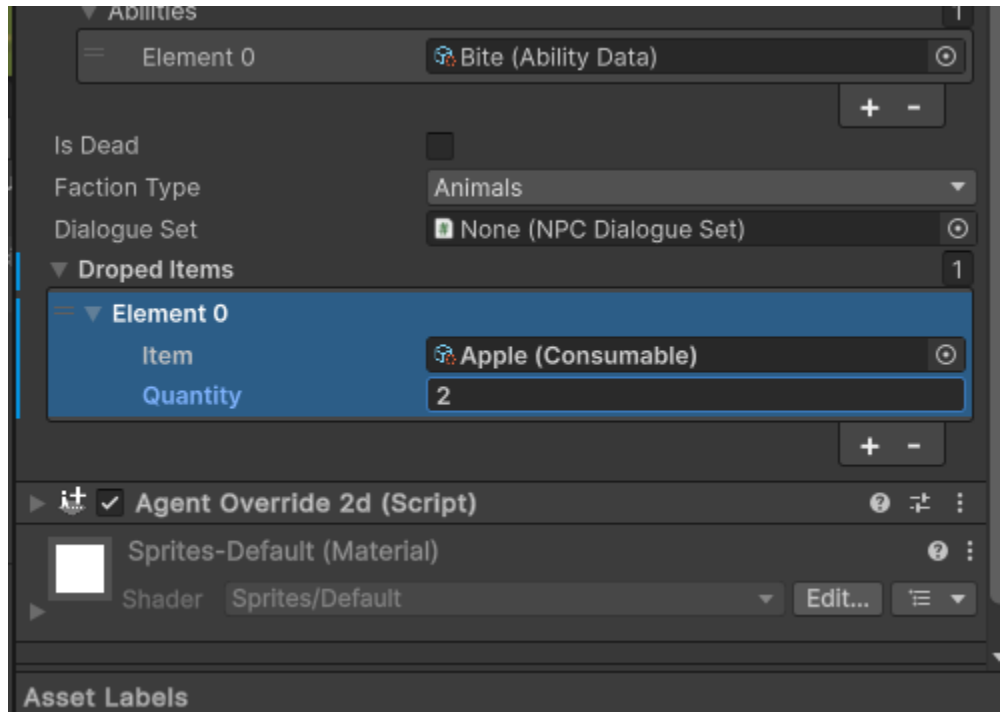
o add or modify the list of factions, open the **NPCEnums** script located at: **Assets/Scripts/Abstracts/NPCEnums.cs**
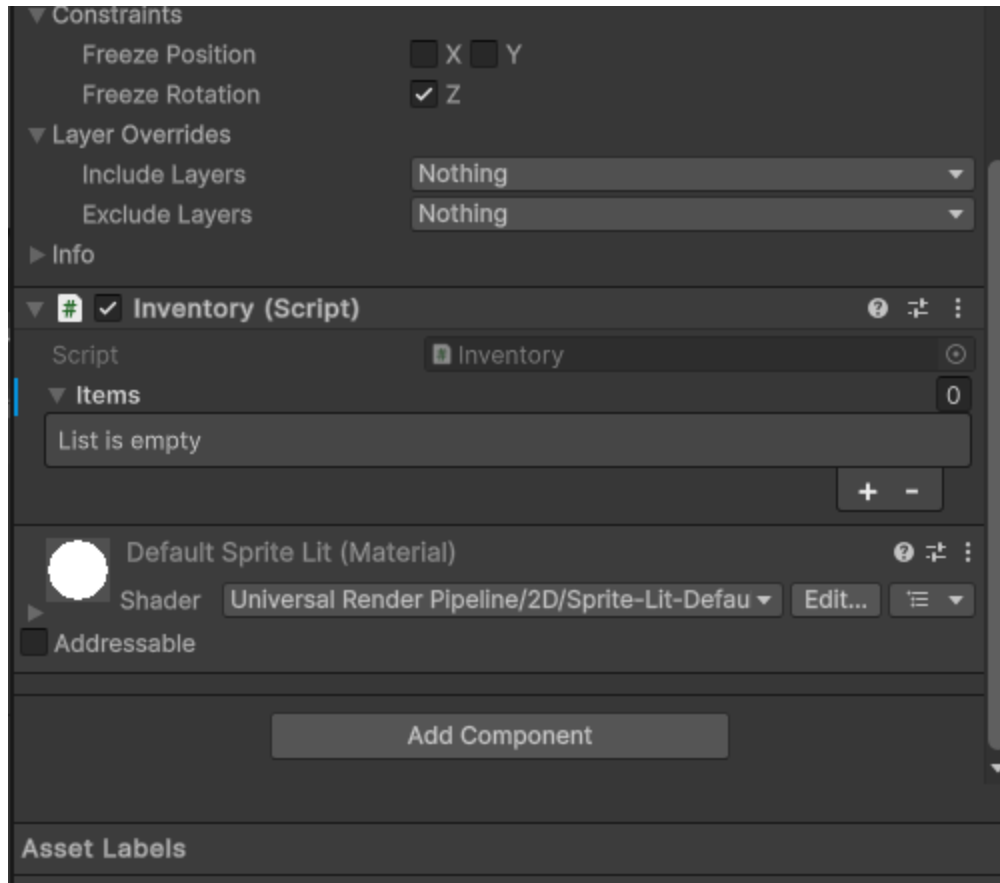Find the FactionType enum and add the new faction names to that list.

Party window and inventory

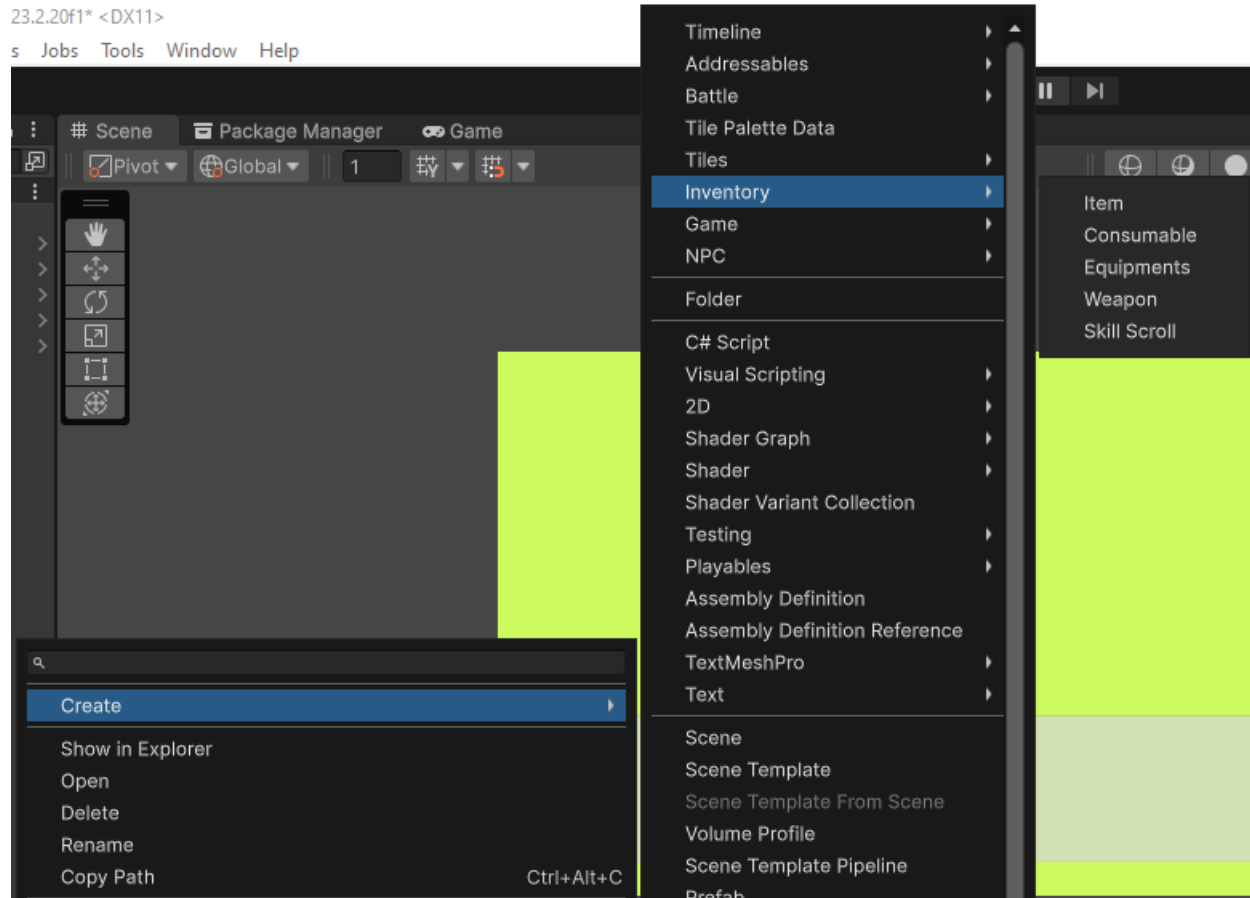Dropped items - items that will drop from the NPC when he dies in battle
Item - drop item
Quantity - the number of items that will drop

the player has a script that stores information about what items the player has

Creating items

Folders for storing items: Assets/Data/Items

items that can be created are in Create->Inventory

Расходники:

Apple (Consumable)

Open

☑ Addressable Apple   Select

Group   📁 Default Local Group   ⊙

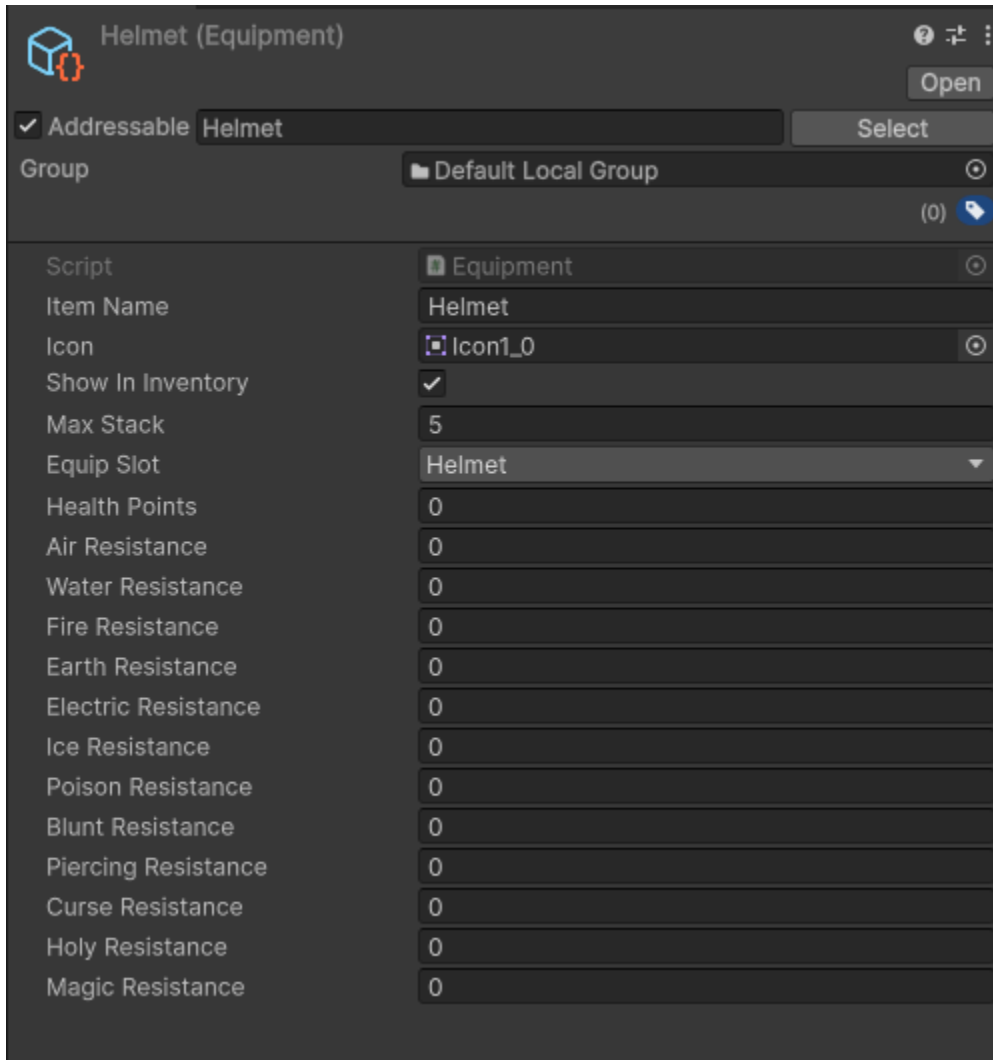   (0) 🏷

Script	🗎 Consumable	⊙
Item Name	Apple	
Icon	🖾 3_0	⊙
Show In Inventory	☑	
Max Stack	10	
Health Gain	0	
Hunger Gain	50	
Thirsty Gain	0	

Equipment:

equipSlot-what slot does this item belong to

Weapon:
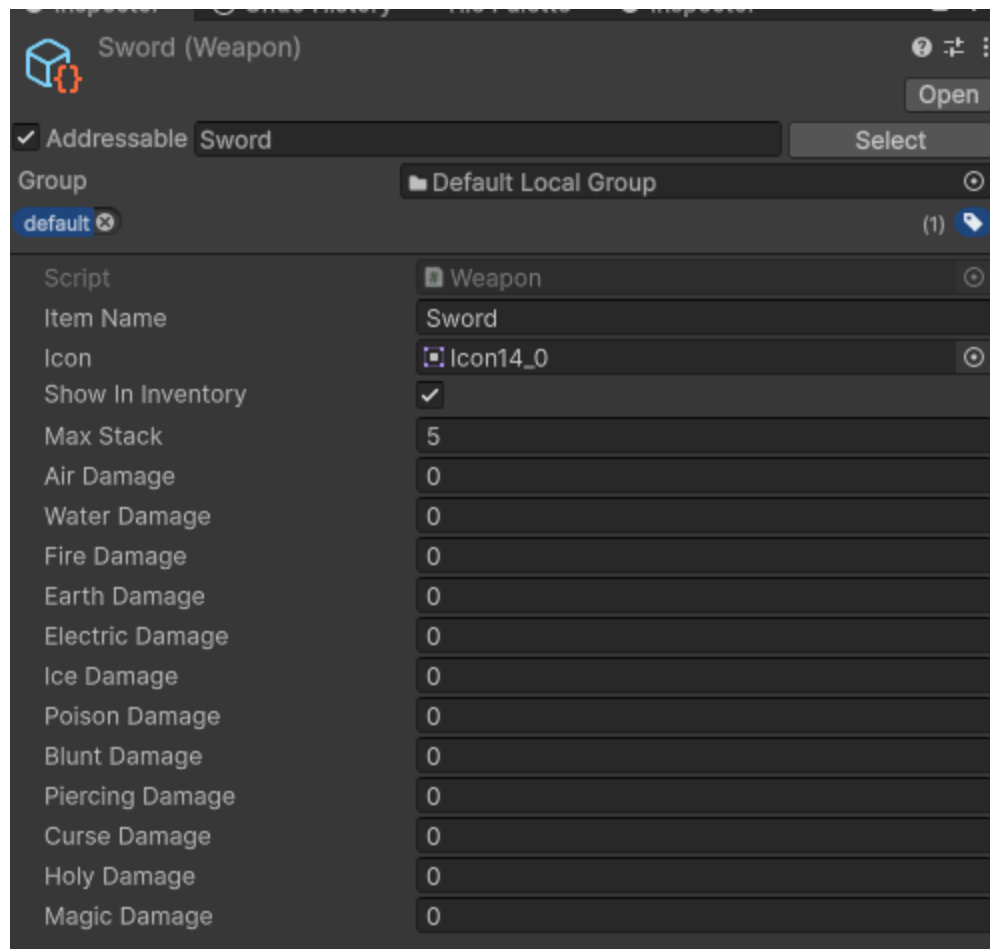
Sword (Weapon)		
Addressable	Sword	Select
Group	📁 Default Local Group	⊙
default ✖		(1) 🏷
Script	📄 Weapon	⊙
Item Name	Sword	
Icon	▣ Icon14_0	⊙
Show In Inventory	✓	
Max Stack	5	
Air Damage	0	
Water Damage	0	
Fire Damage	0	
Earth Damage	0	
Electric Damage	0	
Ice Damage	0	
Poison Damage	0	
Blunt Damage	0	
Piercing Damage	0	
Curse Damage	0	
Holy Damage	0	
Magic Damage	0	

Spels:

Ability - reference to the ability from Assets/BattleData/Ability that is used in the battle scene