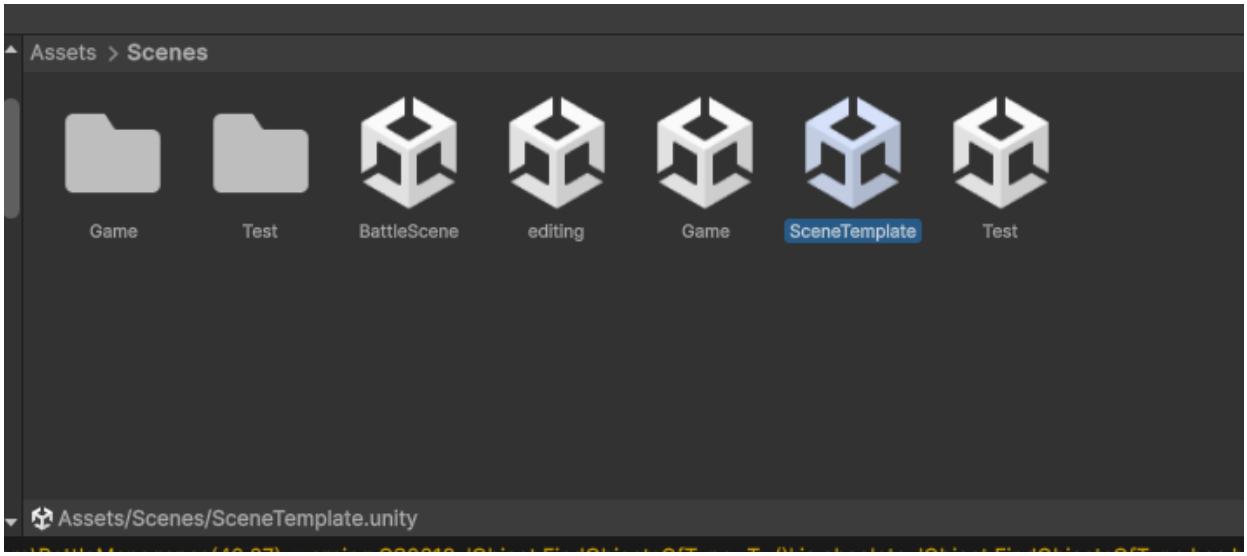
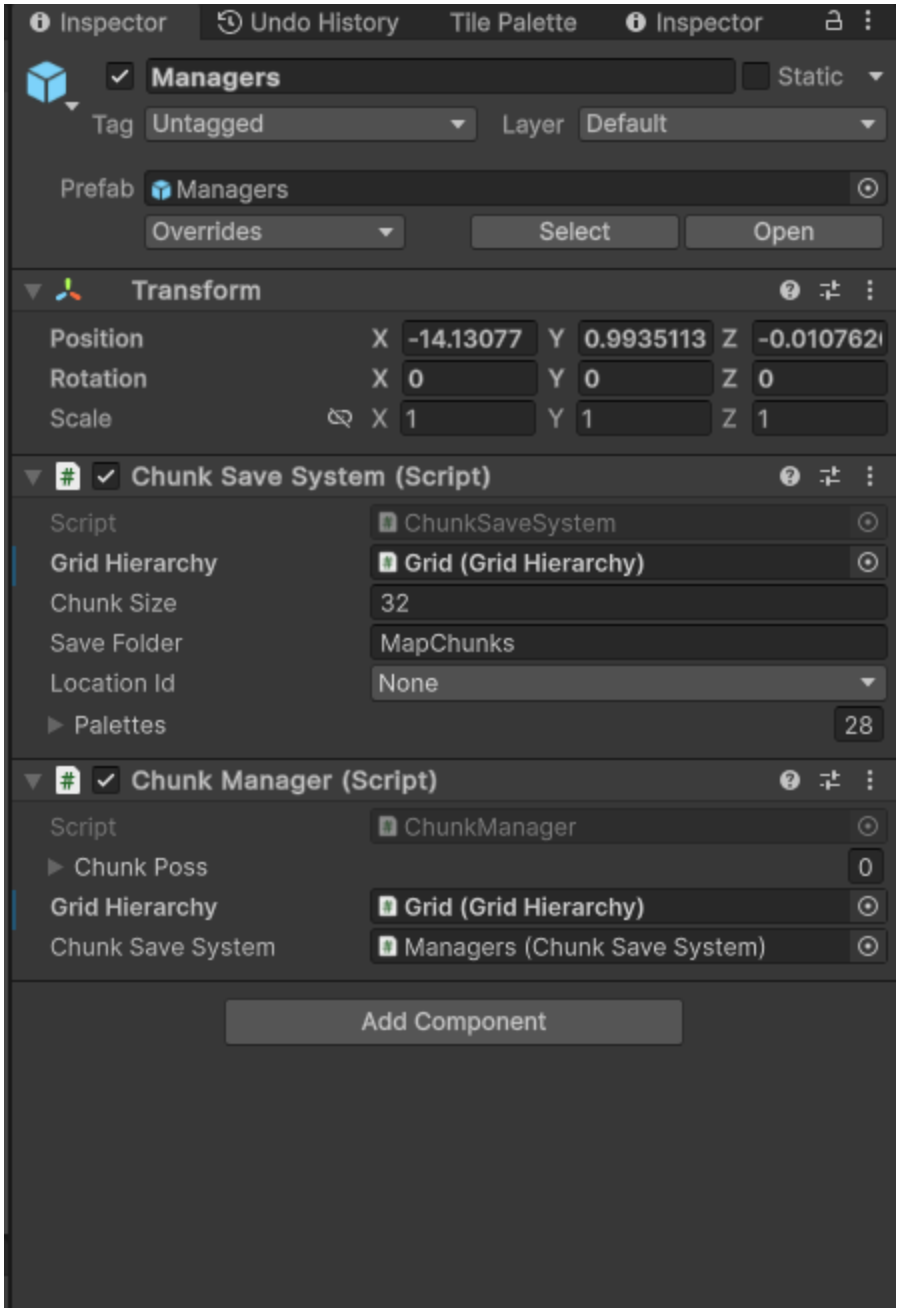
Map and houses

Creating maps

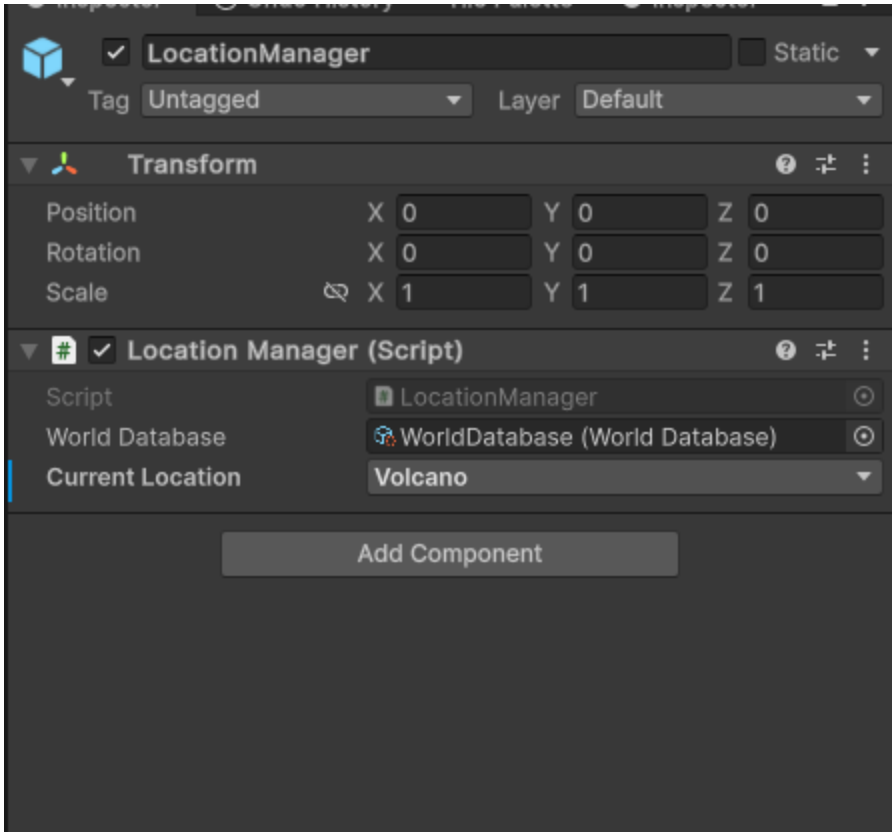


To create a new map, simply duplicate the SceneTemplate, which already contains everything you need.

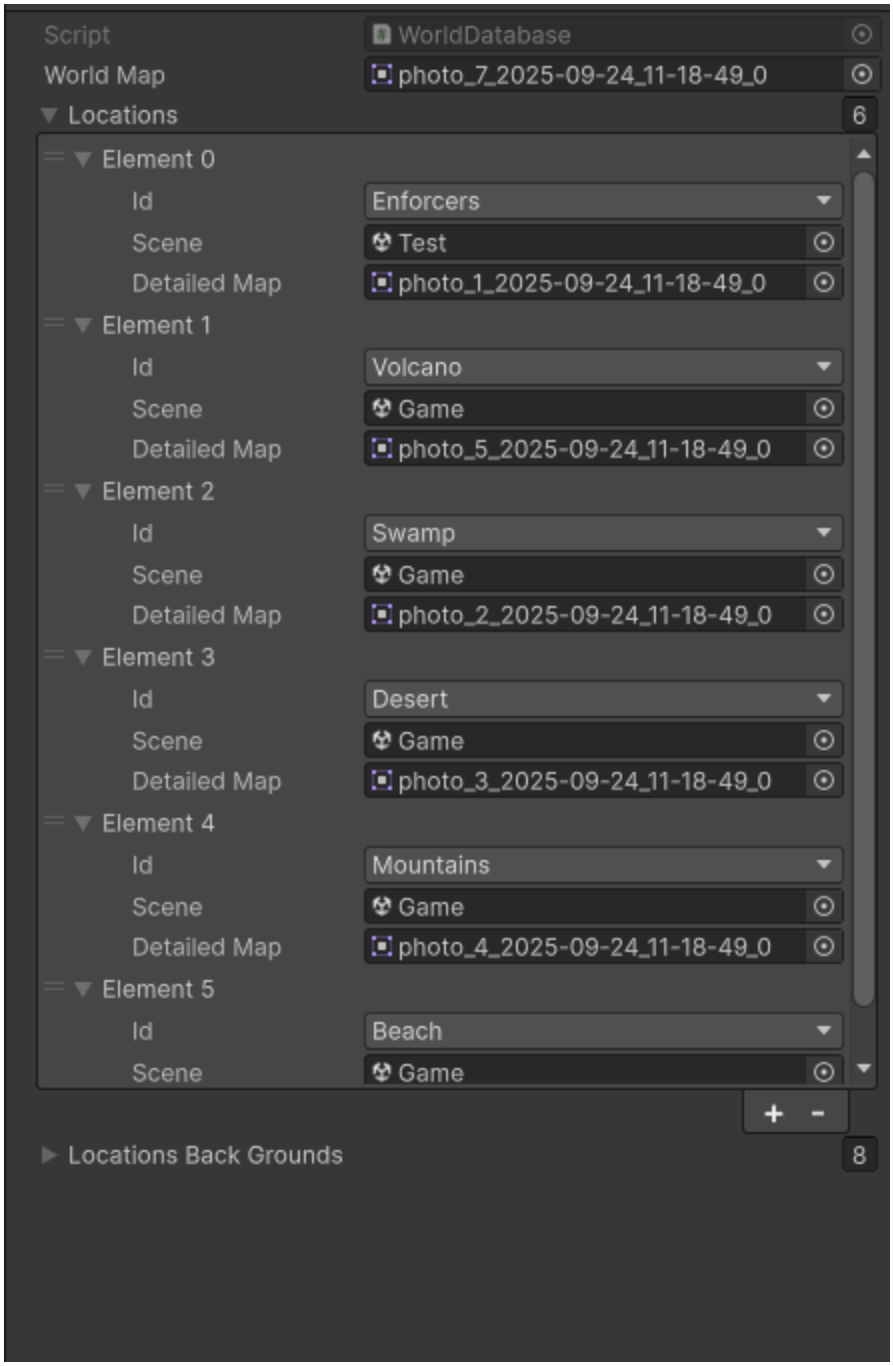


In LocationId, you must specify which location this scene belongs to. To expand/change the list of location names, you must find “public enum LocationId” in the ChunkSaveSystem script. and just add a new name separated by commas

In the child objects, you need to find the Location Manager, which also needs to specify the corresponding CurrentLocation

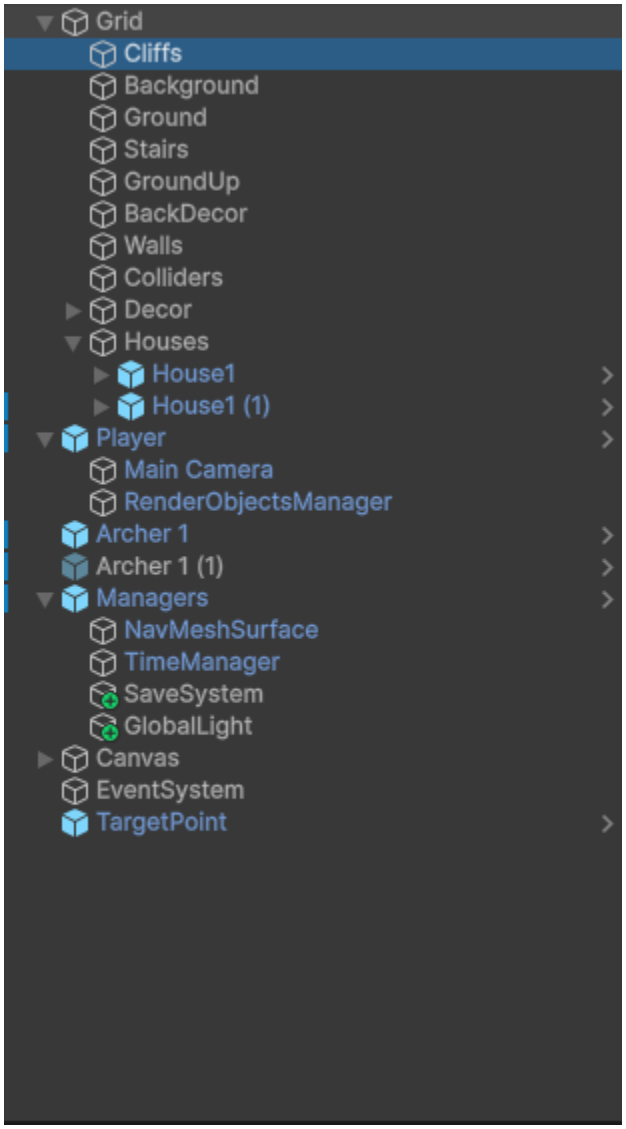


In the Data folder, you need to find WorldDataBase, which serves as a container for storing names and types of scenes.

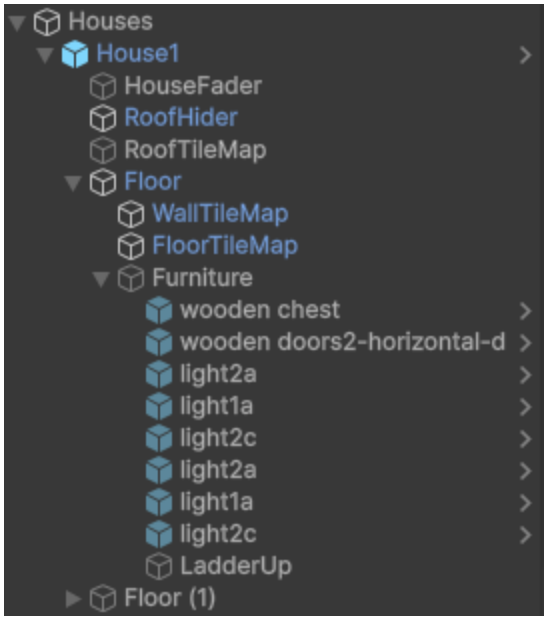


(This script is also needed for the map and background in the battle, more to come)
Simply drag the created scene into a new element in Locations, also specifying the type of location to which it belongs (It is important that the location type in WorldDataBase matches the location type in the scene itself in Managers)

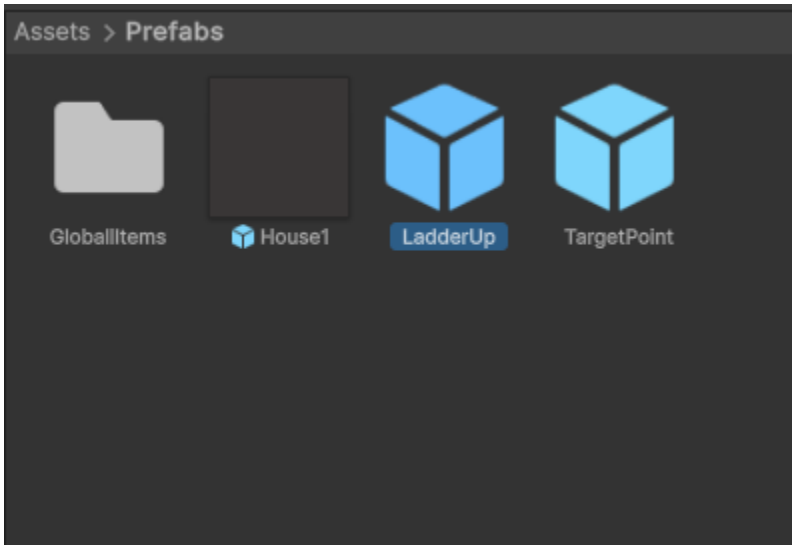
Map settings

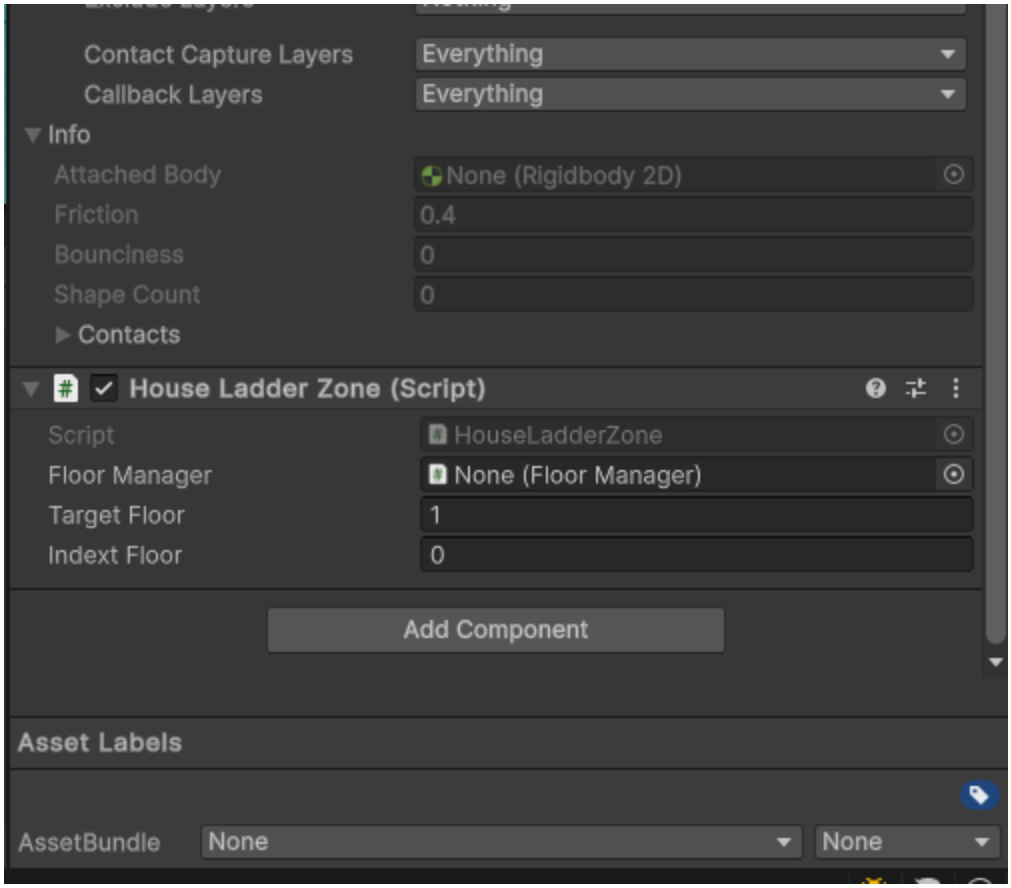


1) Cliffs - used to draw rocks that act as boundaries for the play area, i.e., the edge of the earth.
2) Background - used to draw water and other tiles that will be drawn below the main ground, needed for the effect of depth
3) Ground - used for normal ground
4) Stairs - used for stairs so that the player automatically climbs them
5) GroundUp - logic like Background, but for the elevation effect
6) BackDecor - used for decorative objects that are drawn behind the player.
7) Walls - used for walls that the player cannot pass through
8) Colliders - used for invisible walls
9) Decor - for decorative objects that are drawn on top of the player
    a) You can also add child prefab objects that are animated.
10) Houses - used as a container for houses



a) RoofTileMap is needed for the roof of the house, which is hidden when the player enters the house.
b) Floor is a container required for each floor that has a
    i) WallTileMap - for home walls
    ii) FloorTileMap - for floors
    iii) Furniture for decorations, you can also add child objects and animated prefabs
    iv) For multi-story buildings, it is necessary to add a “Ladder” to each floor; it is necessary for the player to move between floors at the level





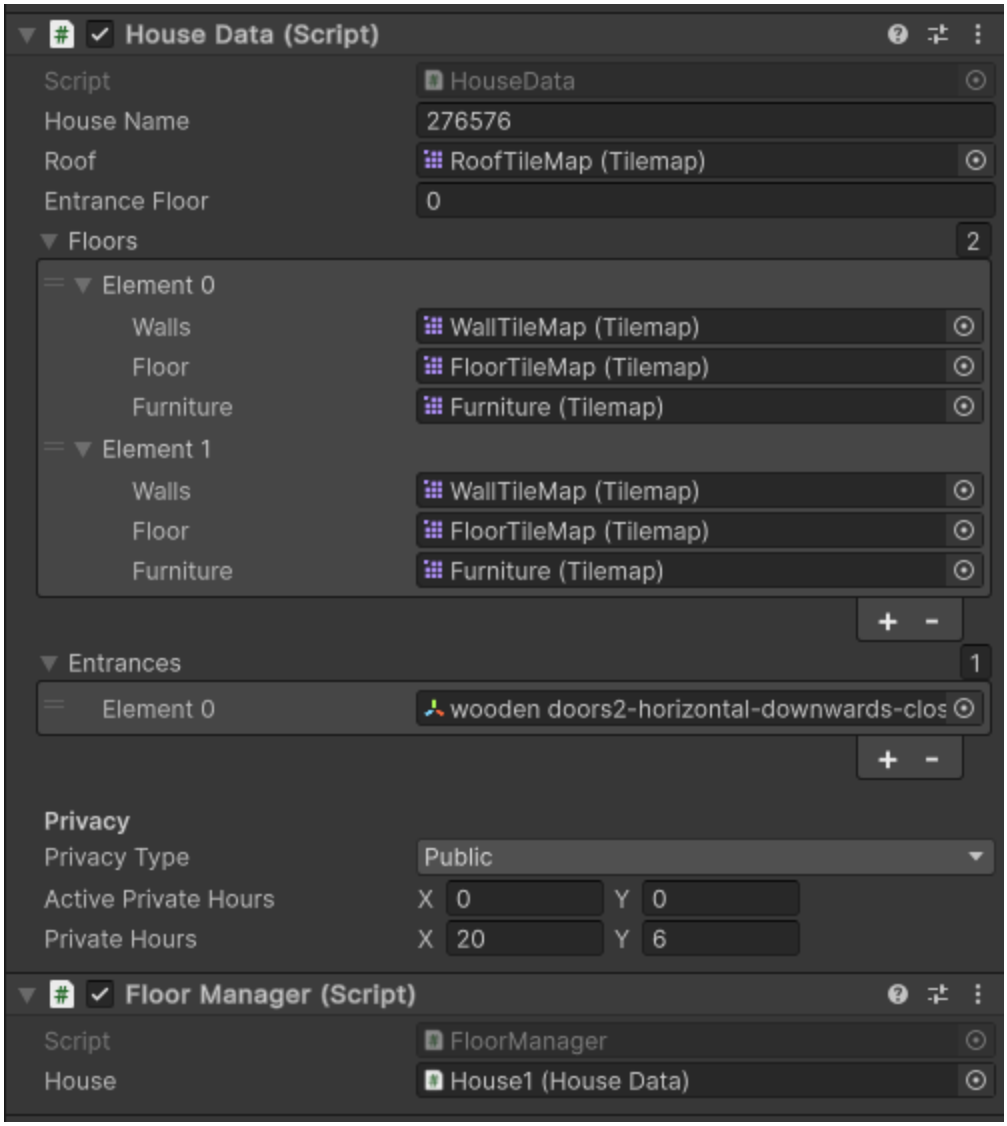
Each staircase has a HouseLadderZone component in which you need to specify

FloorManager - the house in which the staircase is located

TargetFloor - the floor to which the stairs lead

IndexFloor - the floor on which the staircase is located

(floors are numbered starting from 0)



Each house has a HouseData object which contains

Roof - the roof of a house

EntranceFloor - the index of the floor that is the main entrance (you can specify 1 so that the 0th floor is considered the basement)

Floors is a collection of all the floors in the house.

Entrances are the entry point to a house.

Privacy

required to set up home publicity, PrivaceType: Public/Private/PrivateScheduled

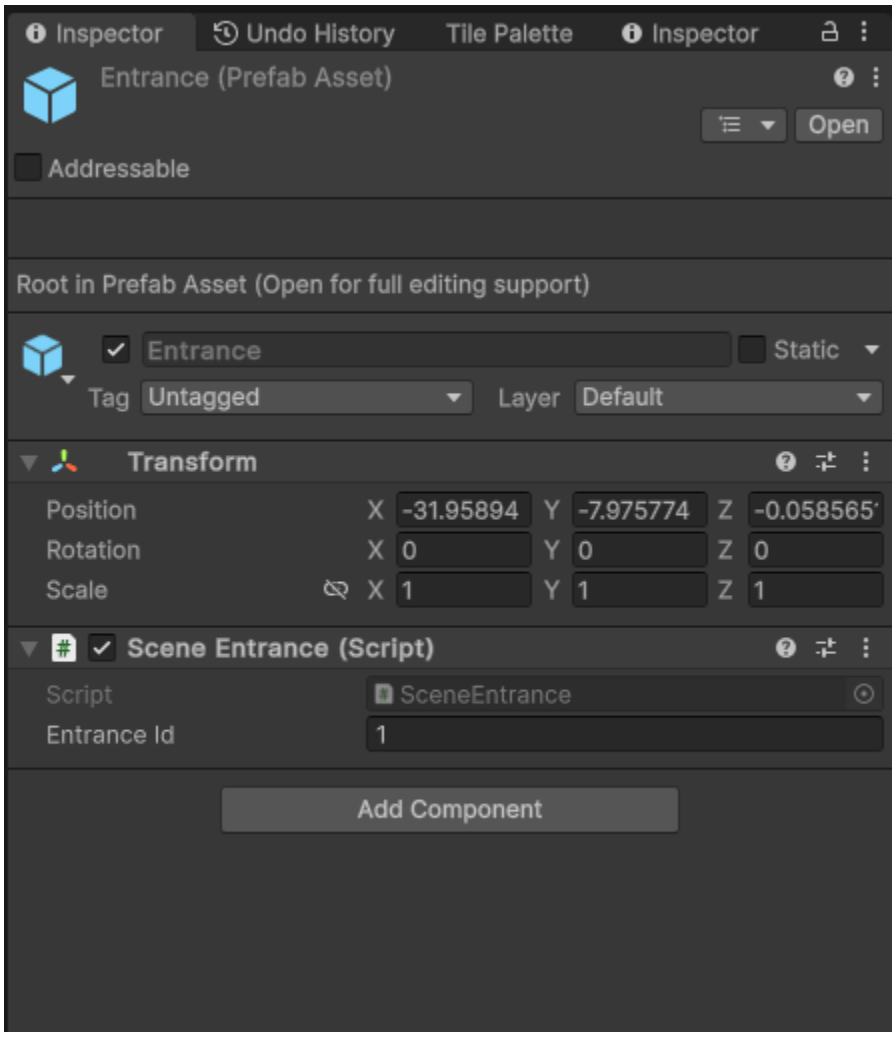
ActivePrivateHours - removed

PrivateHours - the time from which to which the house will be private, used only for PrivateScheduled

You can take a house from the Prefabs folder, which has most of the settings already set.

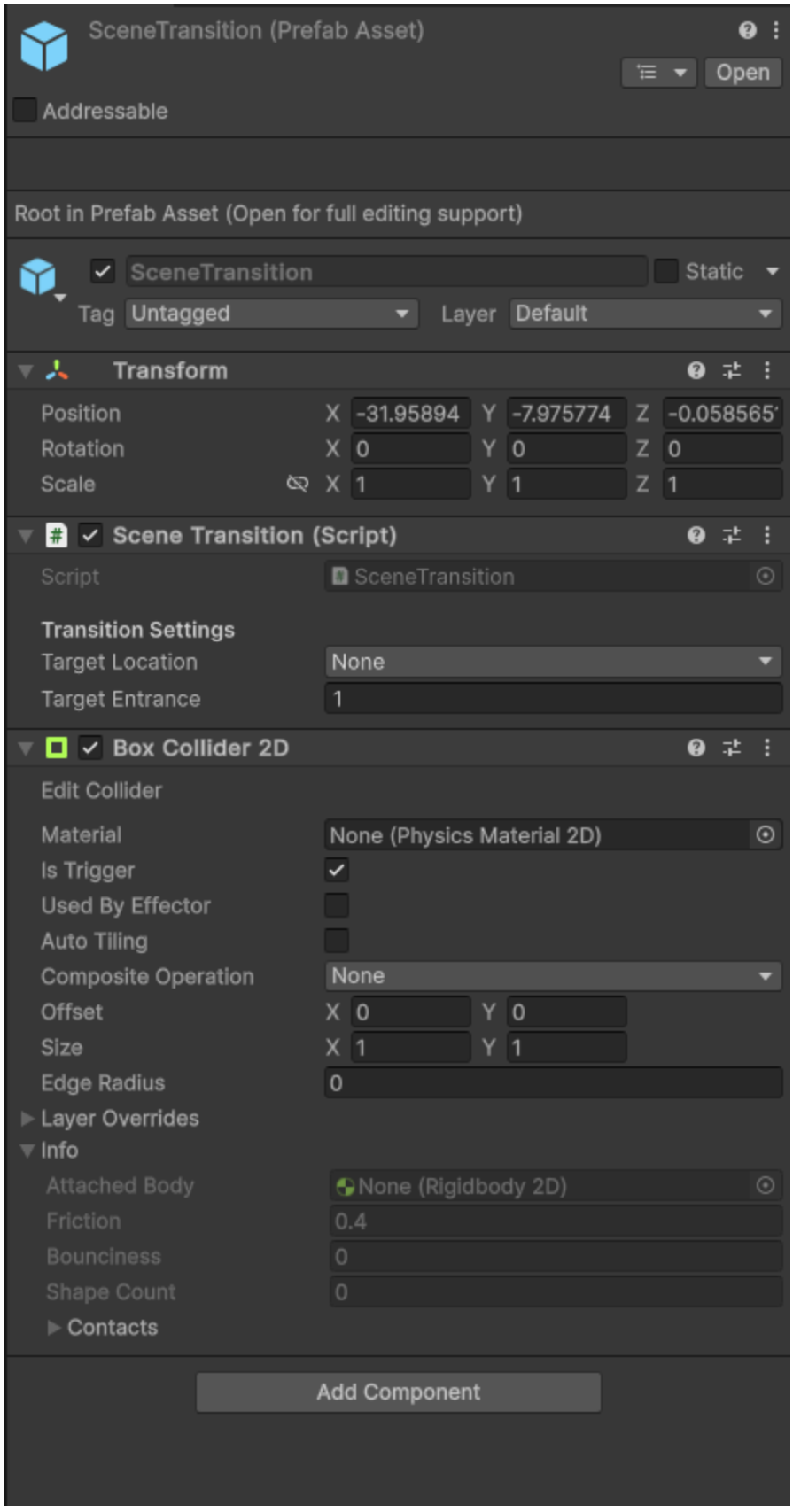
World objects

All objects that can be placed in the world for other systems are located in the Prefabs folder.



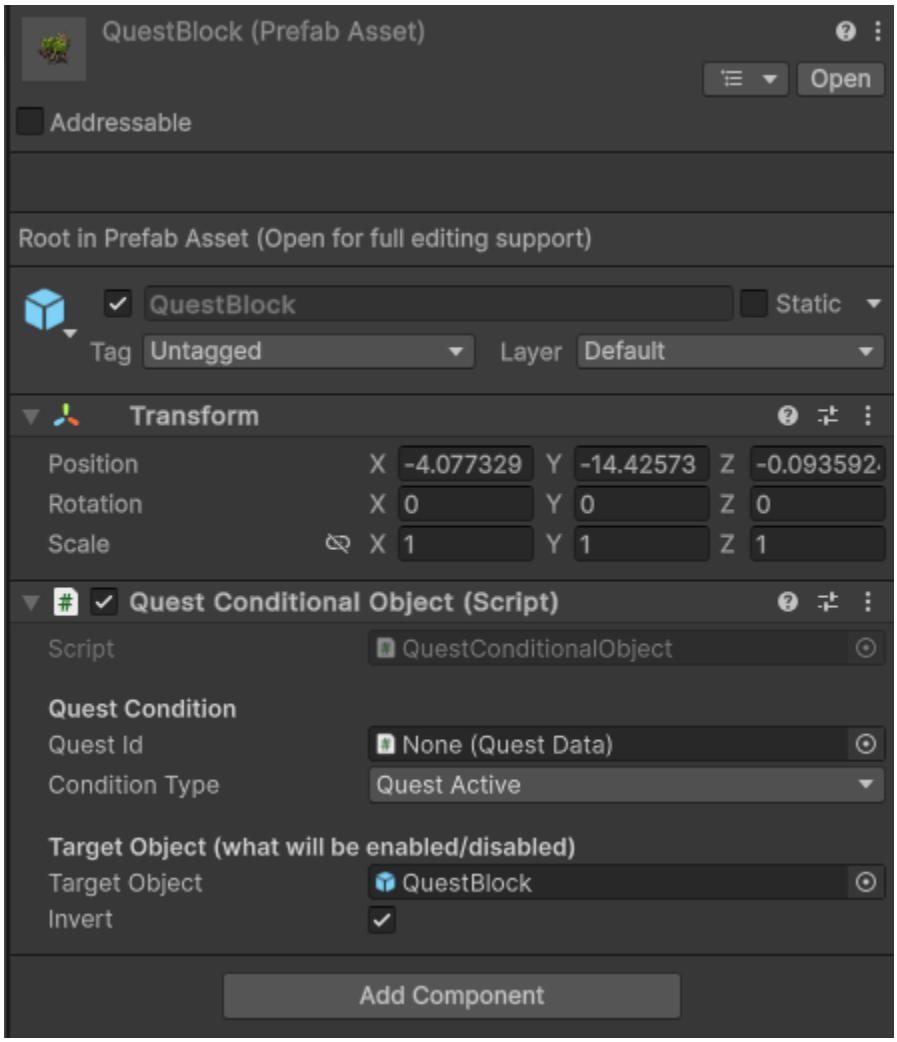
Entrance is needed so that when transitioning between scenes, the player appears in the position where the given object is located.

EntranceId is necessary so that we have multiple entrances on the scene and the player appears at the right one.

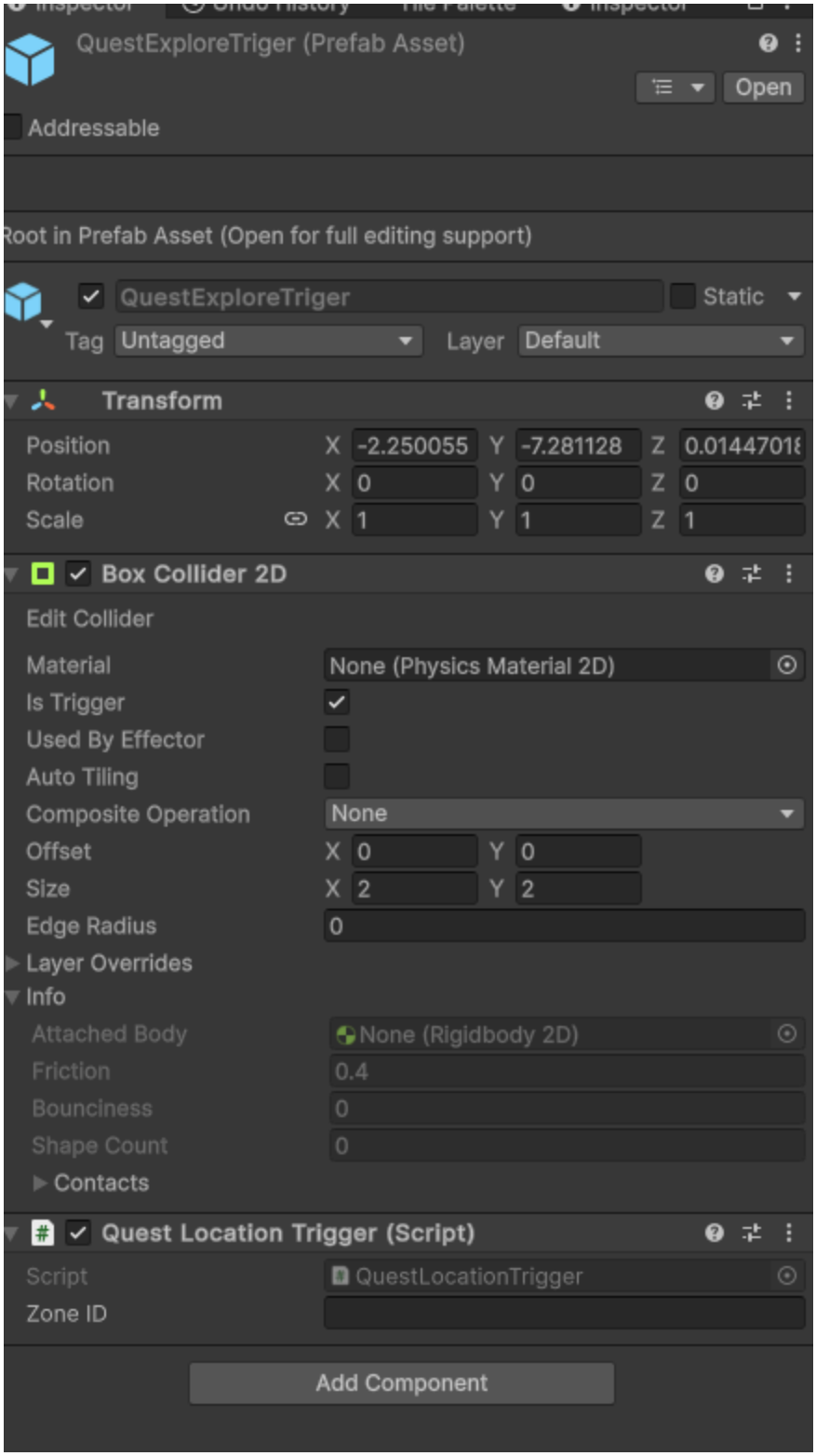

SceneTransition (Prefab Asset)
Open
Addressable
Root in Prefab Asset (Open for full editing support)
SceneTransition
Static
Tag
Untagged
Layer
Default
Transform
Position
X -31.95894
Y -7.975774
Z -0.058565
Rotation
X 0
Y 0
Z 0
Scale
X 1
Y 1
Z 1
Scene Transition (Script)
Script
SceneTransition
Transition Settings
Target Location
None
Target Entrance
1
Box Collider 2D
Edit Collider
Material
None (Physics Material 2D)
Is Trigger
Used By Effector
Auto Tiling
Composite Operation
None
Offset
X 0
Y 0
Size
X 1
Y 1
Edge Radius
0
Layer Overrides
Info
Attached Body
None (Rigidbody 2D)
Friction
0.4
Bounciness
0
Shape Count
0
Contacts
Add Component

SceneTransition is needed so that when entering it, the player moves to the location specified in it, to the specified EntranceId

Quest objects



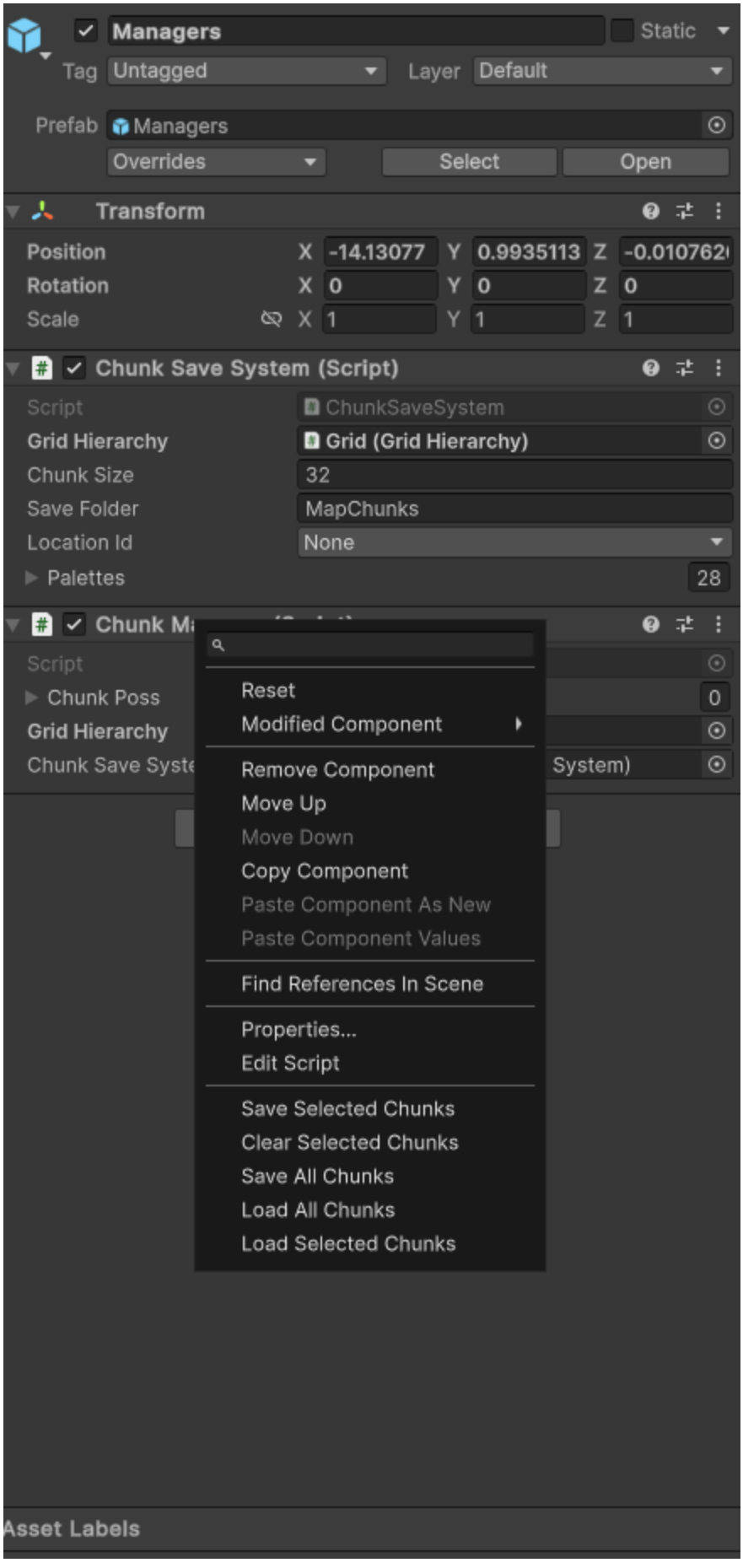
QuestBlock is necessary so that we have areas blocked until the selected quest in Quest ID is completed (More about quests below)
For visual design, it is enough to specify various decorative objects as a child object (the trees in it serve as an example)
Invert is needed to turn on or off objects for obstruction



QuestExploreTriger is necessary so that when a player enters it, the quest that serves to visit a location that will have the same ZoneID is completed

Preservation of Peace

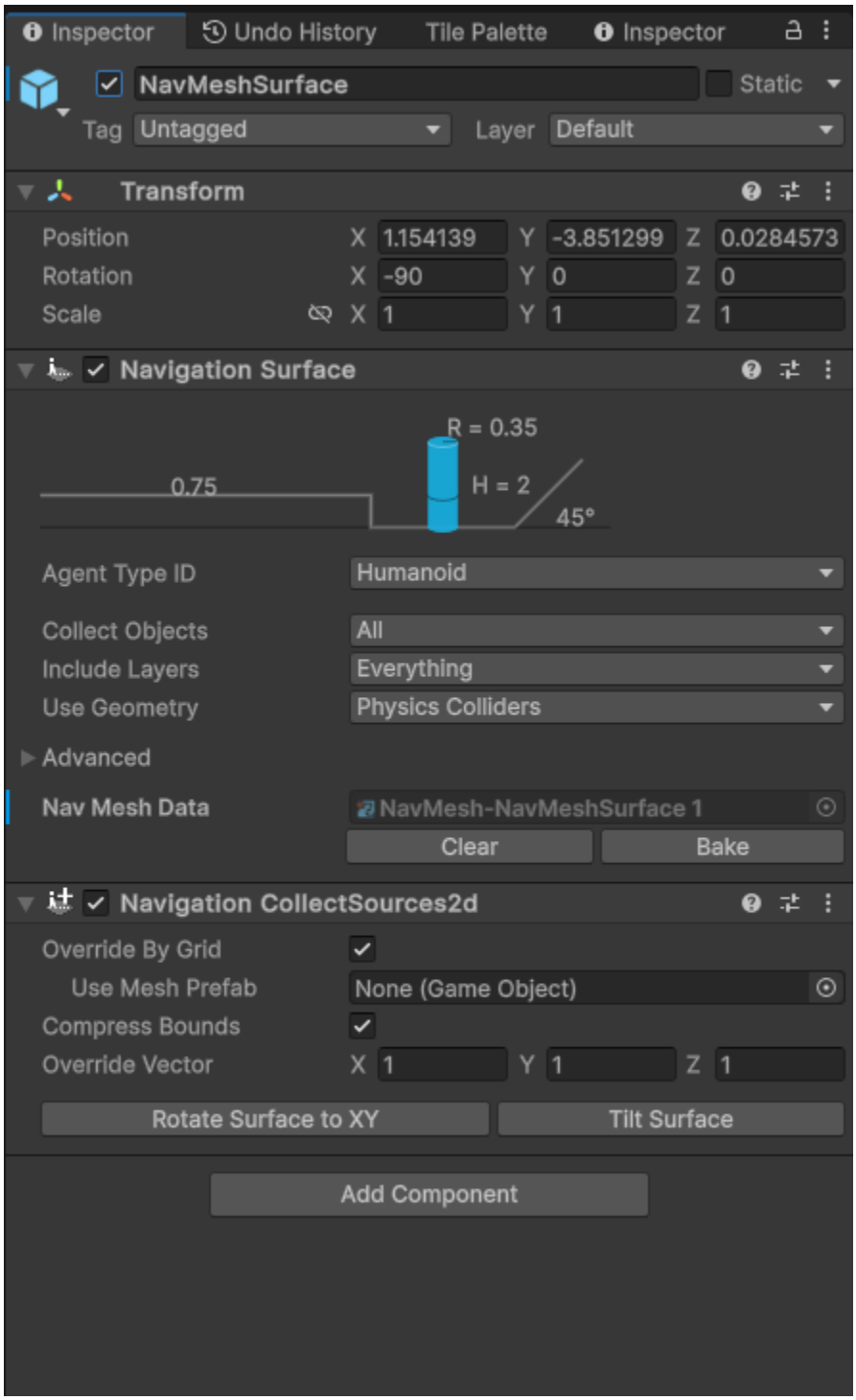
When you right-click on ChunkManager


Managers
Static
Tag
Untagged
Layer
Default
Prefab
Managers
Overrides
Select
Open
Transform
Position
X -14.13077
Y 0.9935113
Z -0.010762
Rotation
X 0
Y 0
Z 0
Scale
X 1
Y 1
Z 1
Chunk Save System (Script)
Script
ChunkSaveSystem
Grid Hierarchy
Grid (Grid Hierarchy)
Chunk Size
32
Save Folder
MapChunks
Location Id
None
Palettes
28
Chunk Ma
Script
Chunk Poss
0
Grid Hierarchy
Chunk Save Syste
System)
Reset
Modified Component
Remove Component
Move Up
Move Down
Copy Component
Paste Component As New
Paste Component Values
Find References In Scene
Properties...
Edit Script
Save Selected Chunks
Clear Selected Chunks
Save All Chunks
Load All Chunks
Load Selected Chunks
Asset Labels

There will be additional buttons in the form
Save Selected Chunks - saves the chunks specified in Chunk Poss
Clear Selected Chunks (Chunk Poss)
Save All Chunks - saves all possible chunks
Load All Chunks - loads all possible chunks
Load Selected Chunks - loads the selected chunks into Chunk Poss
Chunk Poss automatically adds modified chunks, so there is no need to add them after changing
All chunks will be saved in the corresponding folder with their location.

Navigation for NPCs

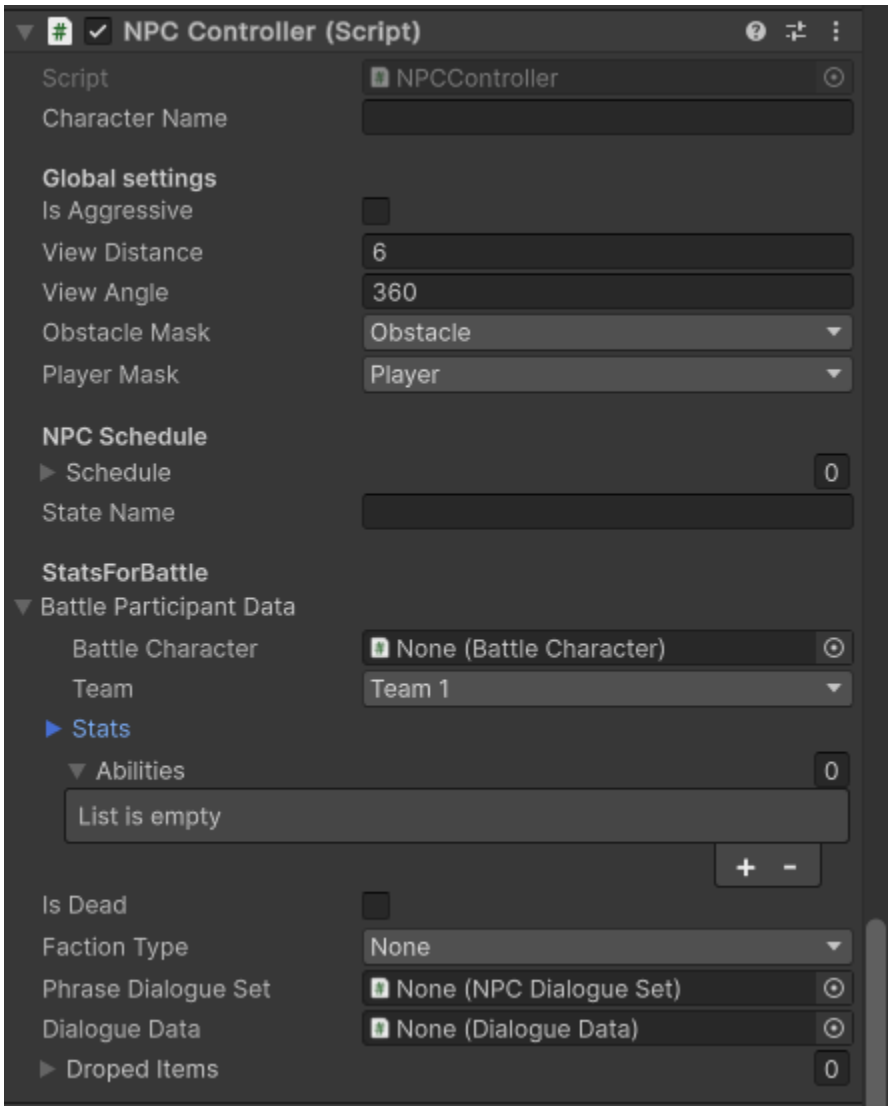
After the map is drawn, you need to find it in the child objects



NavMeshSurface in which you need to press the “Bake” button
This component is necessary so that NPCs can move around the designated area.

NPCs and schedules

For basic NPC control, the NPCController script is used.

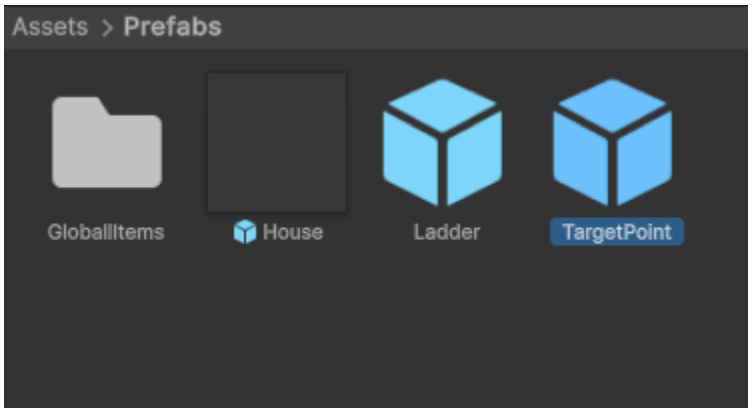


isAggressive is a checkbox that determines whether the character will pursue the hero as soon as he enters his line of sight (at the stage “Factions & Reputation"I think it will be redone)
View Distance controls the NPC's range of vision.
View Angel is responsible for the viewing angle
Obstacle & Player Mask does not need to be changed
Schedule - List of NPC schedules where

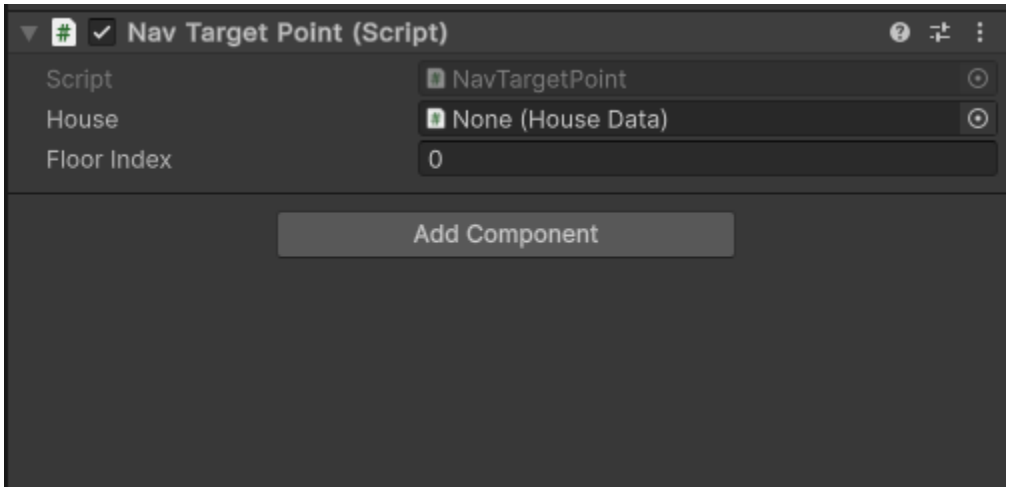
a) Hour/Minute - the time at which the event will begin for the NPC
b) Day - the day on which the NPC's event will begin (if he has the same event every day, you will need to duplicate it for each day separately)
c) Activity - an event that will be executed at a specified time
    i) Sleep is the state in which the NPC sleeps.
    ii) Work - the state in which it works

iii) Idle - a state in which it stands in one point
iv) Wander - a state in which an NPC simply wanders
v) Trade - a state in which you can trade with NPCs
vi) Guard - the state in which the NPC guards, you must specify the point at which he will stand
vii) Patrol - Same state as Guard, but instead of standing still, it will walk around
viii) Hide - A state in which the NPC is invisible and cannot be interacted with.
ix) Hunt - in this state, the NPC simply wanders around and if it sees a player, it begins to pursue them.
x) Chill - a state in which an NPC walks near a point within a small radius

d) TargetPoint - a dummy point that will not be visible, but it is needed for the player to run to it; it is created as a prefab for ease of cloning.



each point also has the following parameter



It is necessary if the point is located in the house so that the NPC can correctly walk around the house.

House - we indicate the house in which this point is located

FloorIndex is the field in which we indicate on which floor the point is located

If the point is located on the street, then these fields do not need to be touched.

StateName - used for debugging

Battle Particiant Data is a field that stores information for the battle where

Battle Character - a link to the prefab of the unit that will fight (located in the BattleData/Prefabs folder)

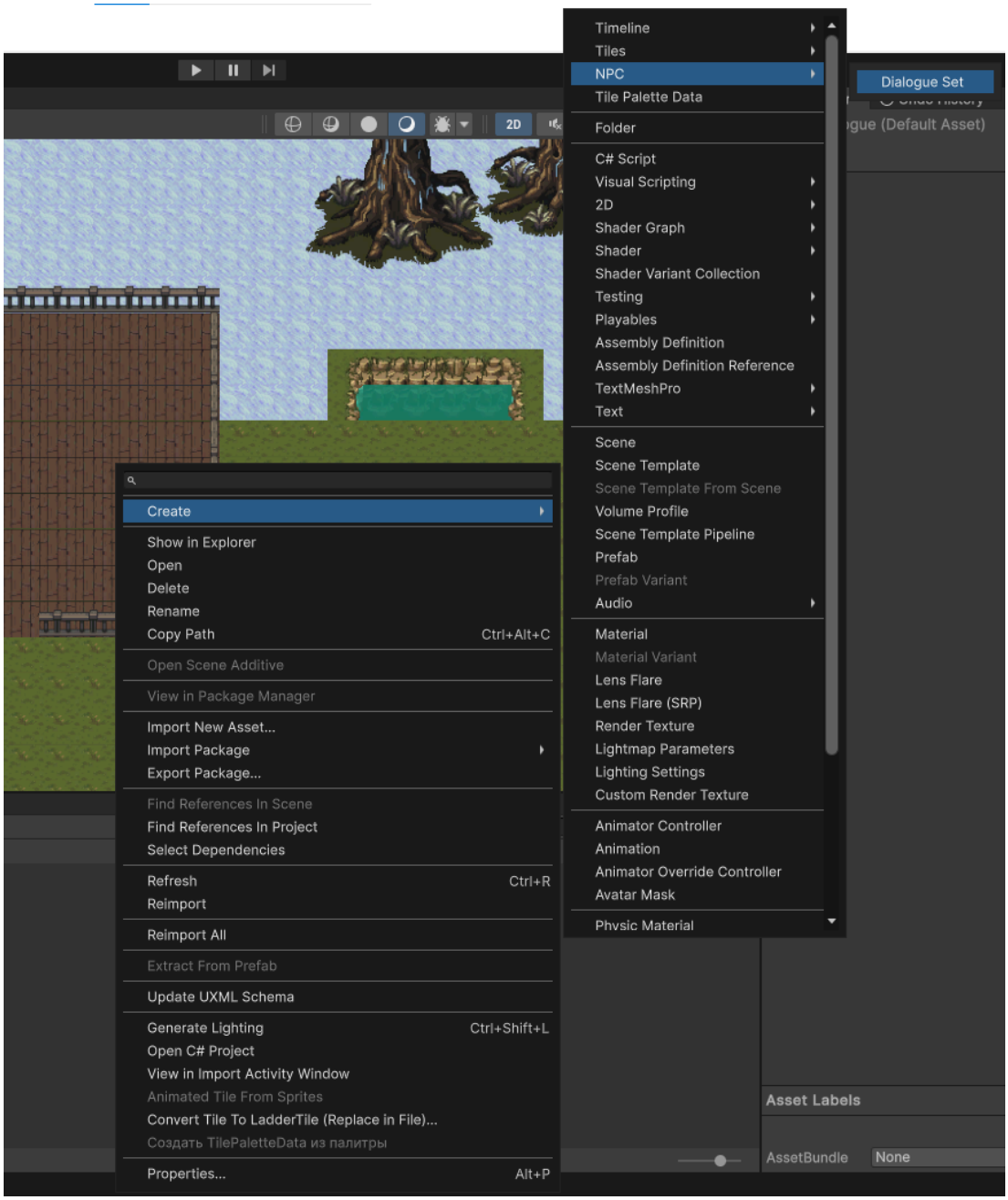
Team - not used in any way

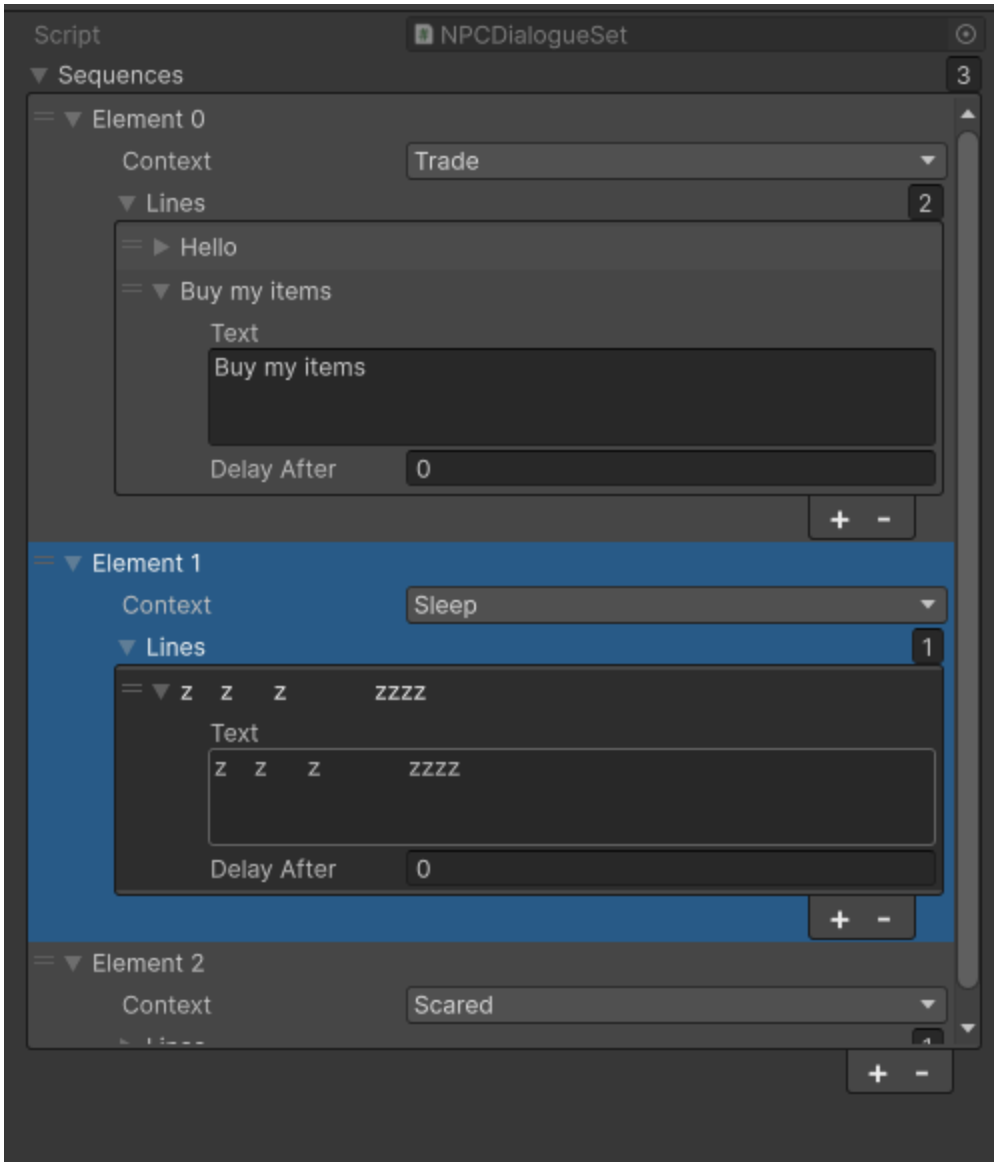
Stats - fields for unit characteristics

Abilities - abilities that a unit possesses

Phrase Dialogue Set is a set of phrases that NPCs say at short intervals depending on their state. For example, if they are trading, they will display the phrase "come buy."

To create a phrase set:





Context is a field that determines the state in which phrases will be pronounced.
The list of states is similar to the states in the NPC schedule

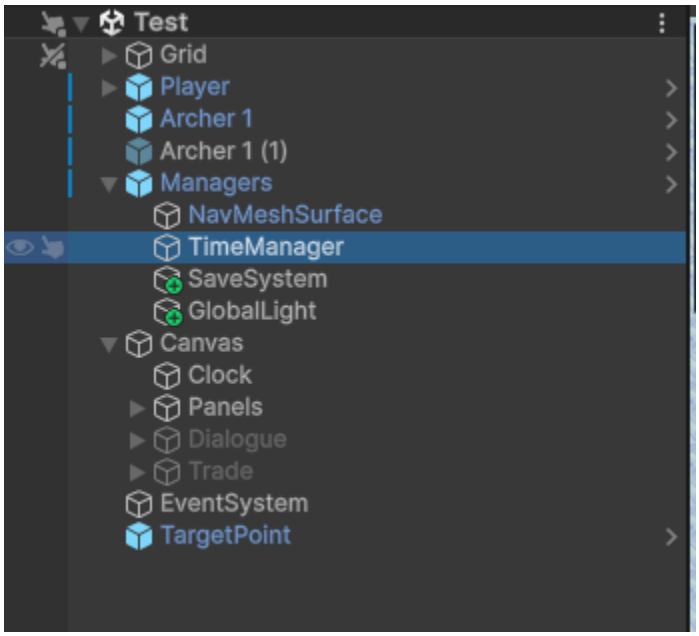
- Idle - also used for both Wander and Chill
- Work,
- Trade,
- Guard - also used for Patrol
- Sleep,
- Hunt,
- Scared

Lines - phrases that will be spoken

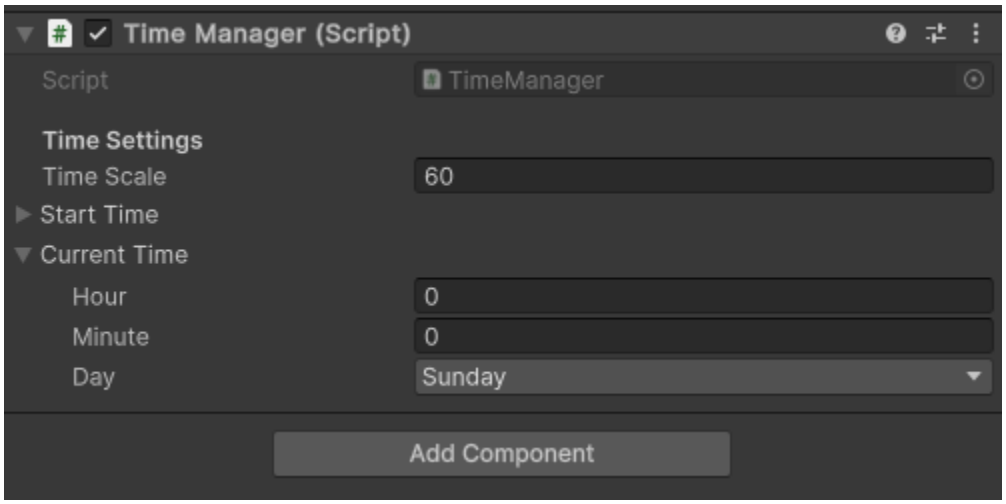
This parameter is optional, so you can either not create it for NPCs or, for example, use the same one for several

DialogueData - which dialogue with the player will be used (below in the Dialogues tab)

Global settings

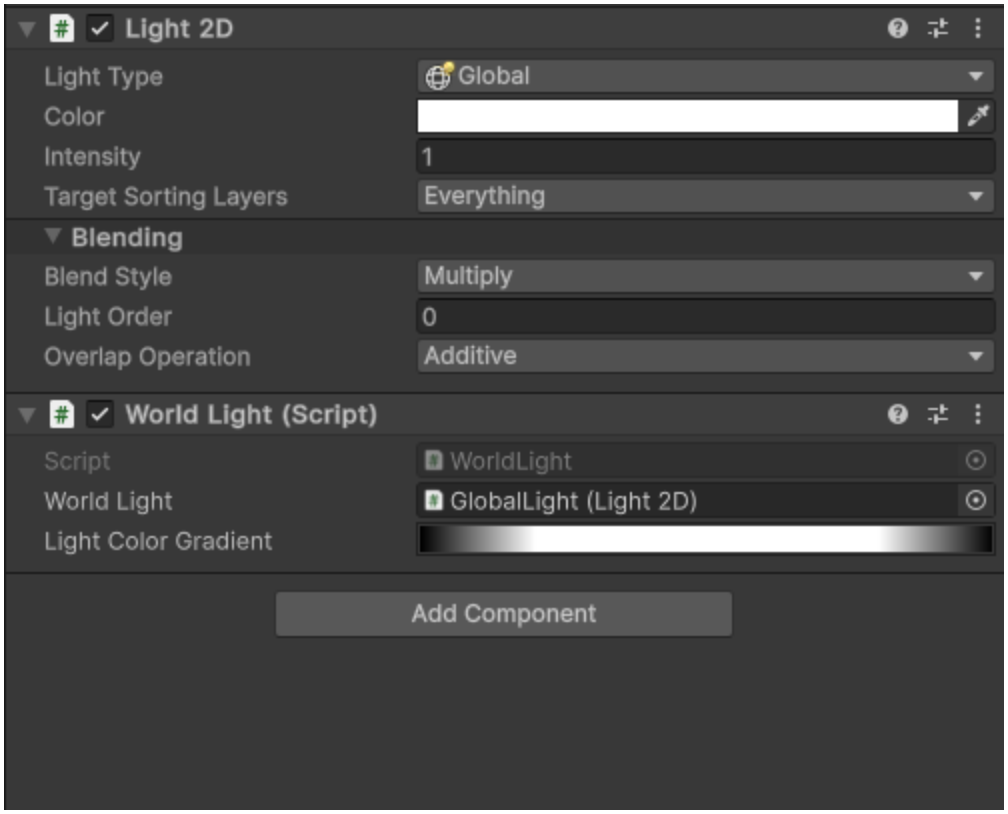


In the Managers section there is a TimerManager which is responsible for in-game time



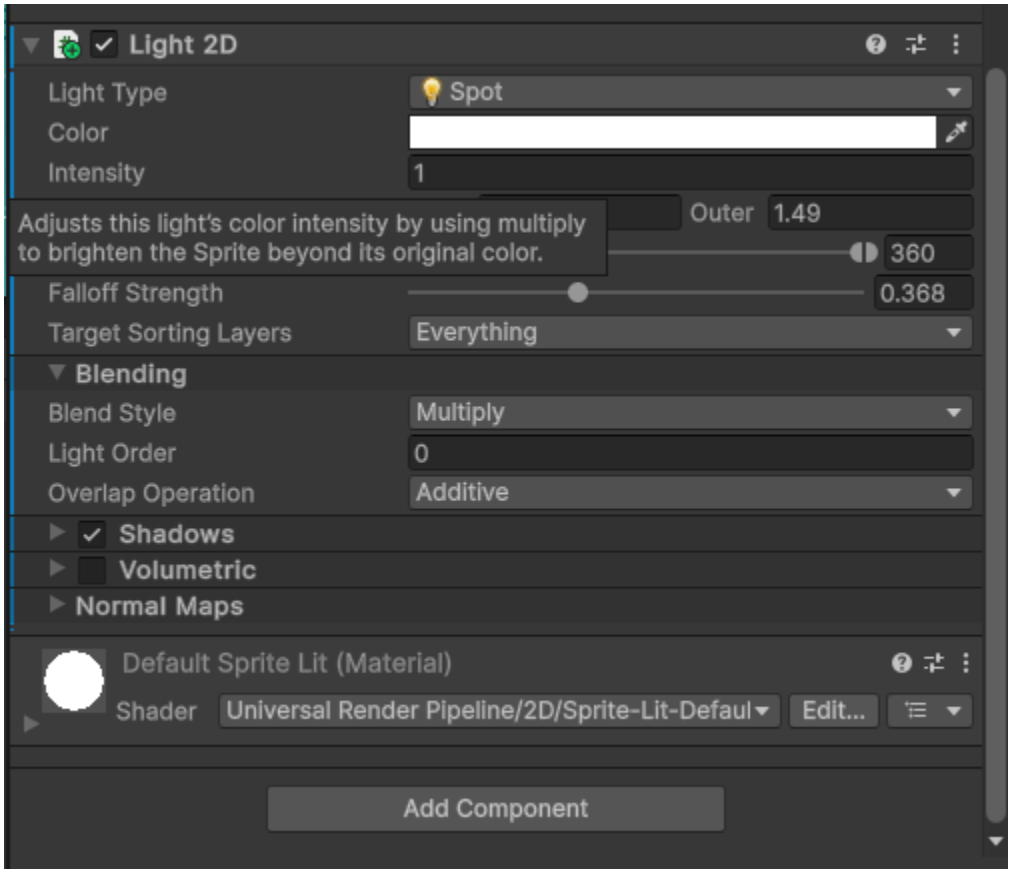
Time Scale is a parameter responsible for the speed of game time, i.e., how much faster it will be than real time, i.e., by what number a second is multiplied

There is also GlobalLight



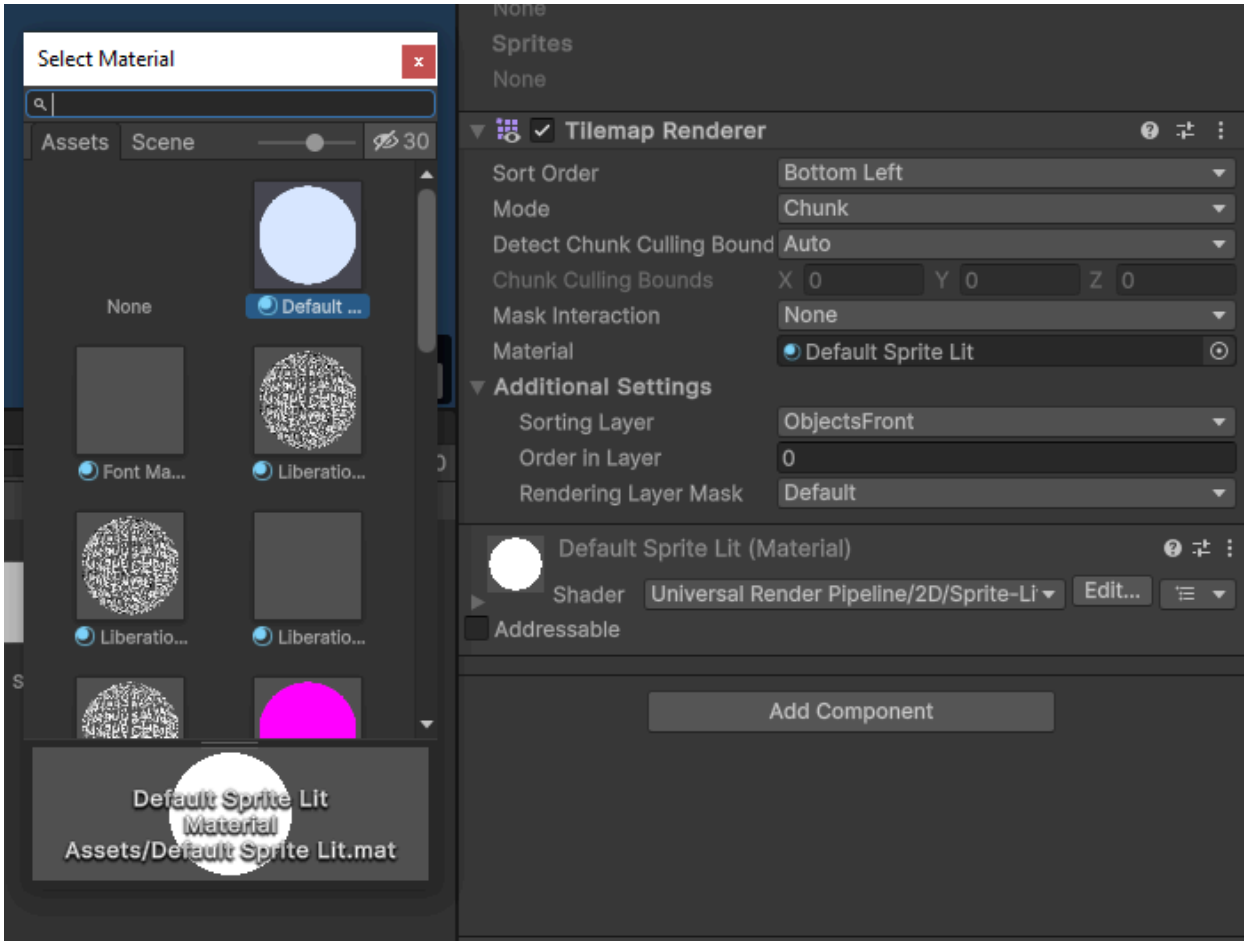
which has a Light Color Gradien parameter that can be adjusted for a more beautiful change in lighting, in it 0% = 00:00 50% = 12:00 100% = 24.00

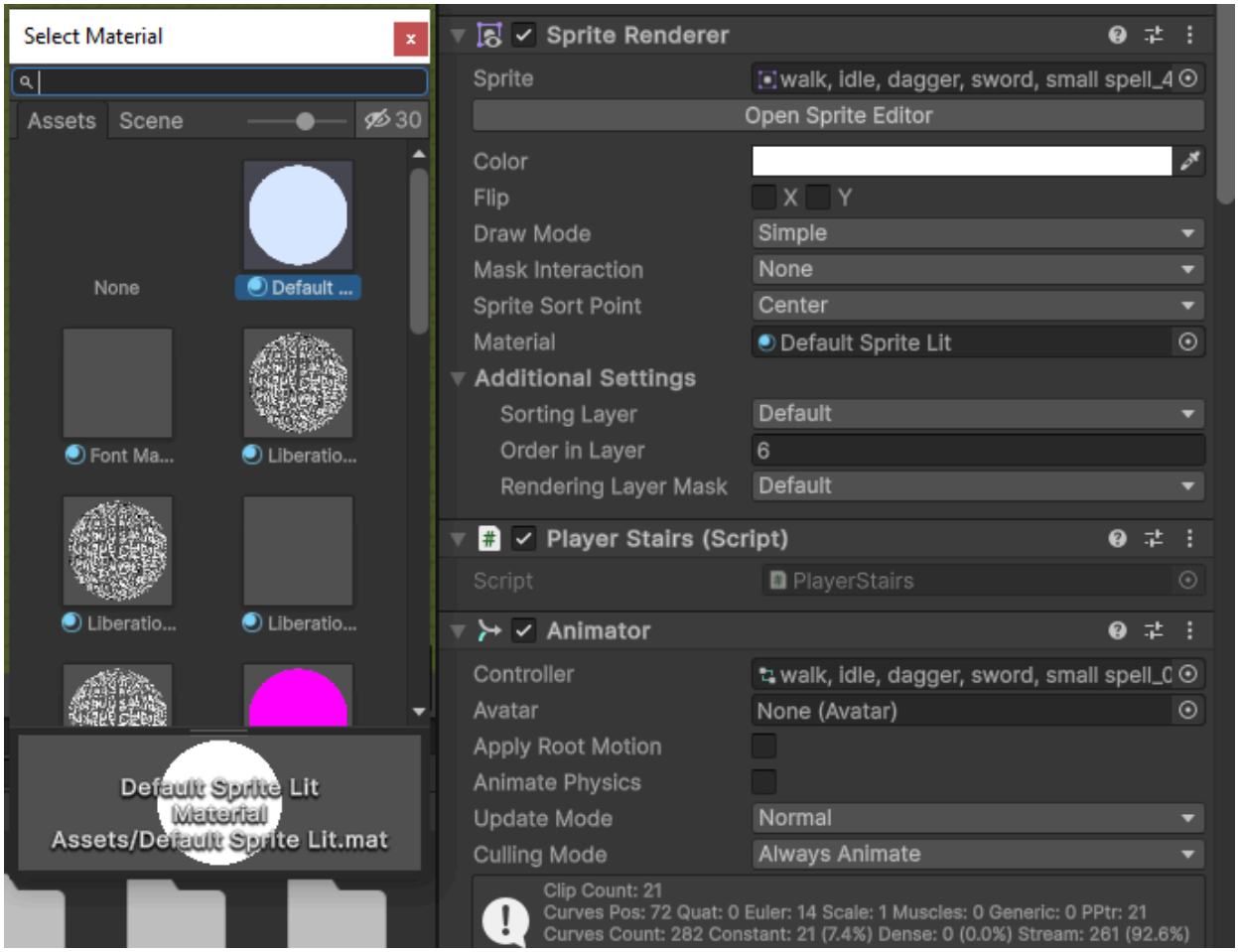
To configure lighting, namely the glow of lamps, etc., you need to add Light2D



and adjust the glow of the desired object

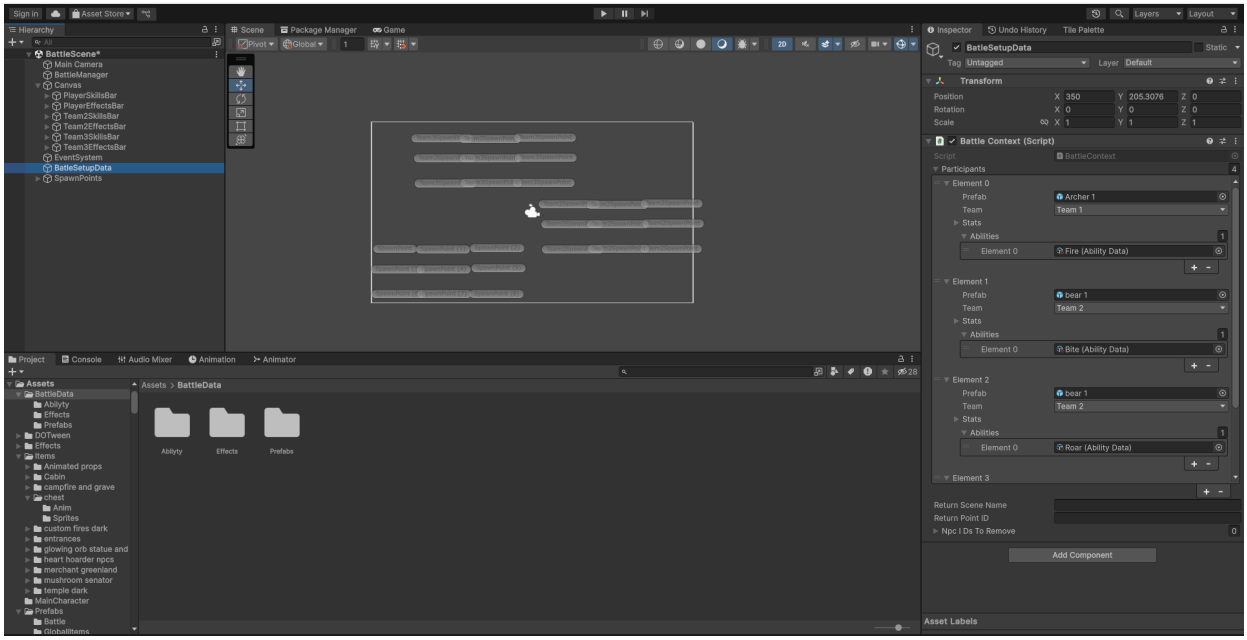
For an object to change its color depending on the lighting, the material of its rendering component must be Default Sprite Lit.


Select Material
Assets
Scene
30
None
Default ...
Font Ma...
Liberatio...
Liberatio...
Liberatio...
Default Sprite Lit
Material
Assets/Default Sprite Lit.mat
Sprites
None
Tilemap Renderer
Sort Order
Bottom Left
Mode
Chunk
Detect Chunk Culling Bound
Auto
Chunk Culling Bounds
X 0
Y 0
Z 0
Mask Interaction
None
Material
Default Sprite Lit
Additional Settings
Sorting Layer
ObjectsFront
Order in Layer
0
Rendering Layer Mask
Default
Default Sprite Lit (Material)
Shader
Universal Render Pipeline/2D/Sprite-Li
Edit...
Addressable
Add Component

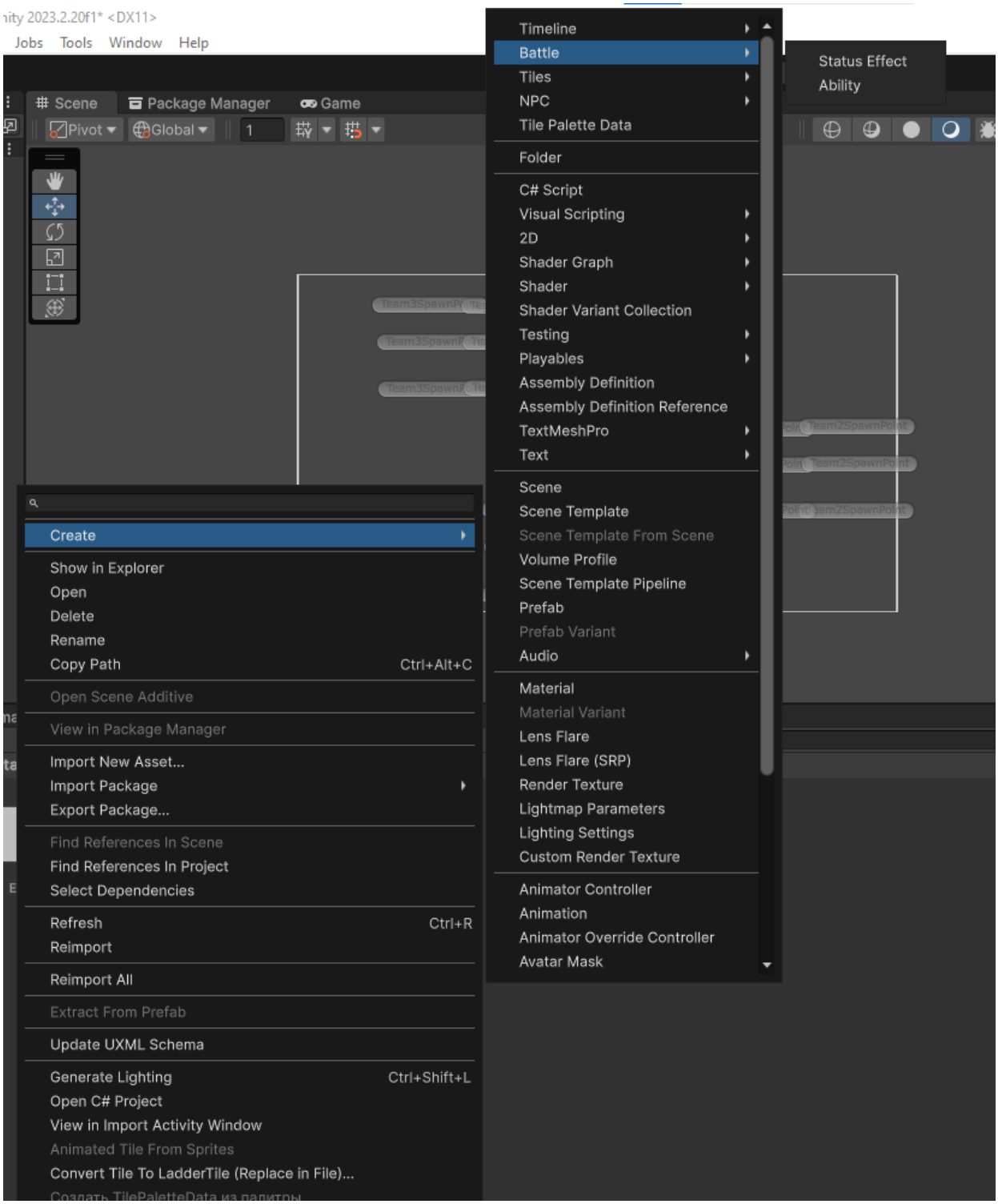


Battle System

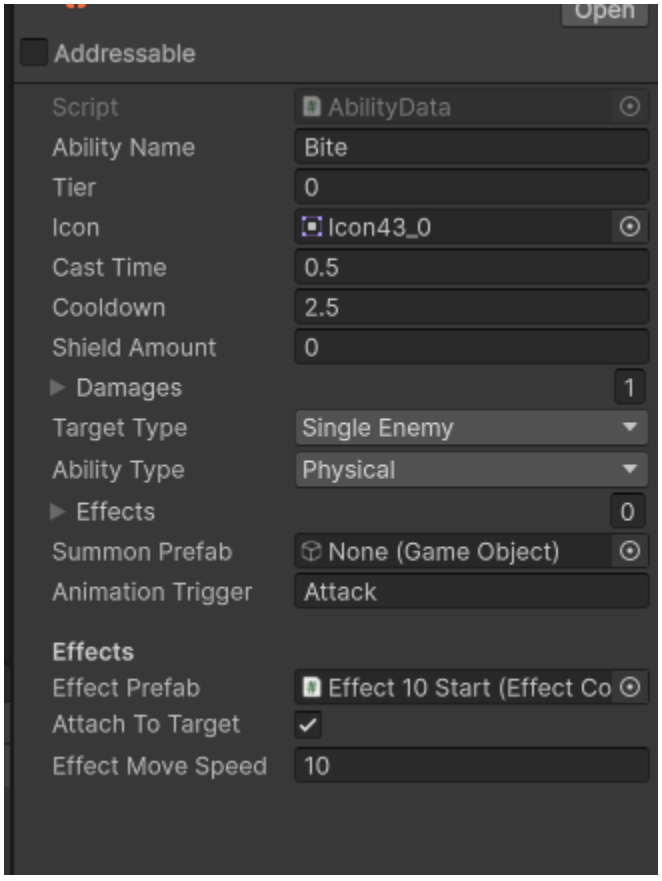
To prepare the battlefield, simply select the character that will be created at the beginning of the battle from the BattleData/Prefabs folder, choose which team he will be on and add the abilities he will have.



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

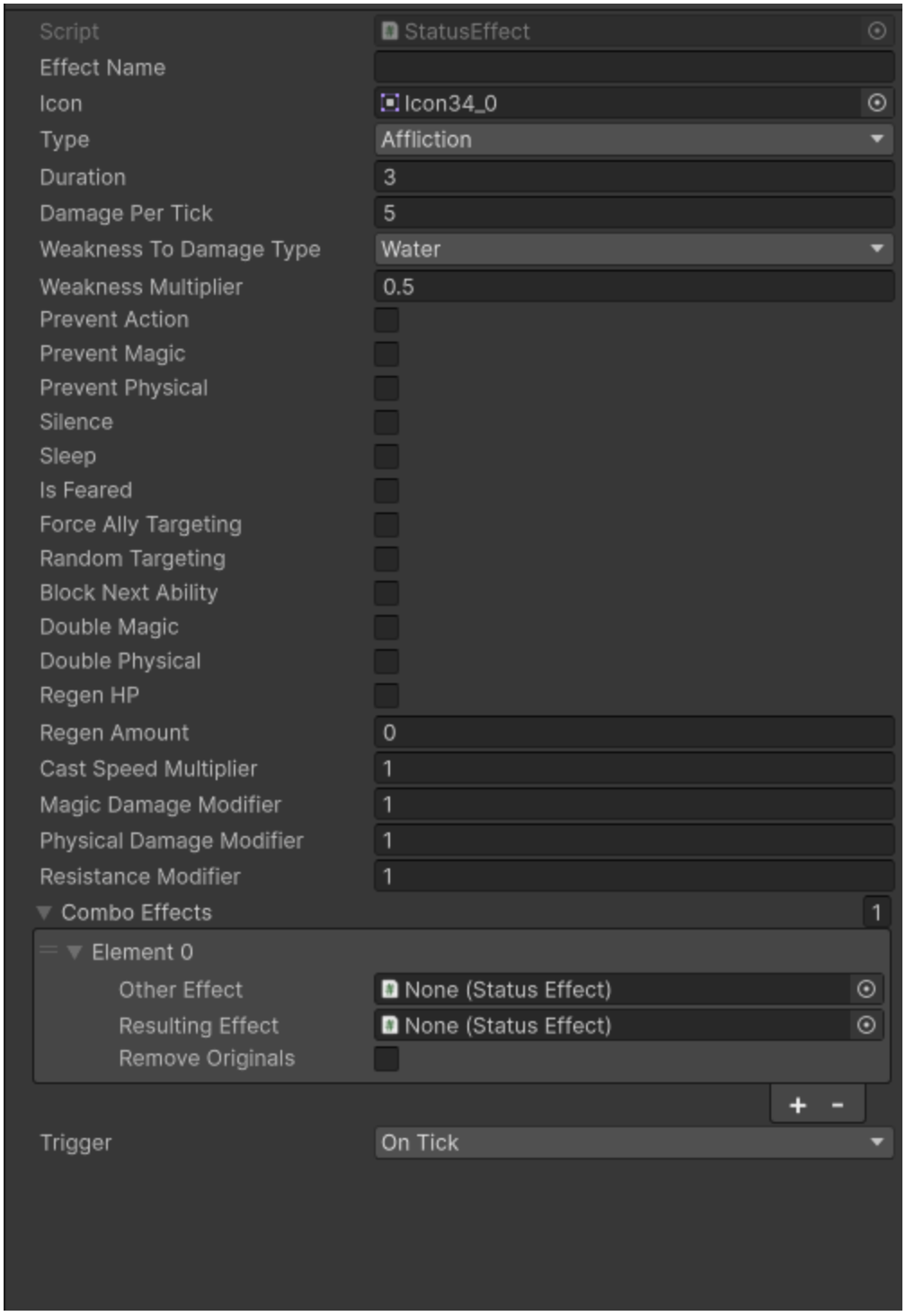

ity 2023.2.20f1* <DX11>
Jobs Tools Window Help
Scene
Package Manager
Game
Pivot
Global
Create
Show in Explorer
Open
Delete
Rename
Copy Path Ctrl+Alt+C
Open Scene Additive
View in Package Manager
Import New Asset...
Import Package
Export Package...
Find References In Scene
Find References In Project
Select Dependencies
Refresh Ctrl+R
Reimport
Reimport All
Extract From Prefab
Update UXML Schema
Generate Lighting Ctrl+Shift+L
Open C# Project
View in Import Activity Window
Animated Tile From Sprites
Convert Tile To LadderTile (Replace in File)...
Timeline
Battle
Tiles
NPC
Tile Palette Data
Folder
C# Script
Visual Scripting
2D
Shader Graph
Shader
Shader Variant Collection
Testing
Playables
Assembly Definition
Assembly Definition Reference
TextMeshPro
Text
Scene
Scene Template
Scene Template From Scene
Volume Profile
Scene Template Pipeline
Prefab
Prefab Variant
Audio
Material
Material Variant
Lens Flare
Lens Flare (SRP)
Render Texture
Lightmap Parameters
Lighting Settings
Custom Render Texture
Animator Controller
Animation
Animator Override Controller
Avatar Mask
Status Effect
Ability

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 applied by this ability.
If you specify a character from the BattleData/Prefabs folder in the Summon Prefab field of an ability, then when you use the ability, that character will be summoned (if there is free space for it)
Animation Trigger: If a character's ability has a special animation, the animation name must be specified differently (see below for how to set up animations correctly).

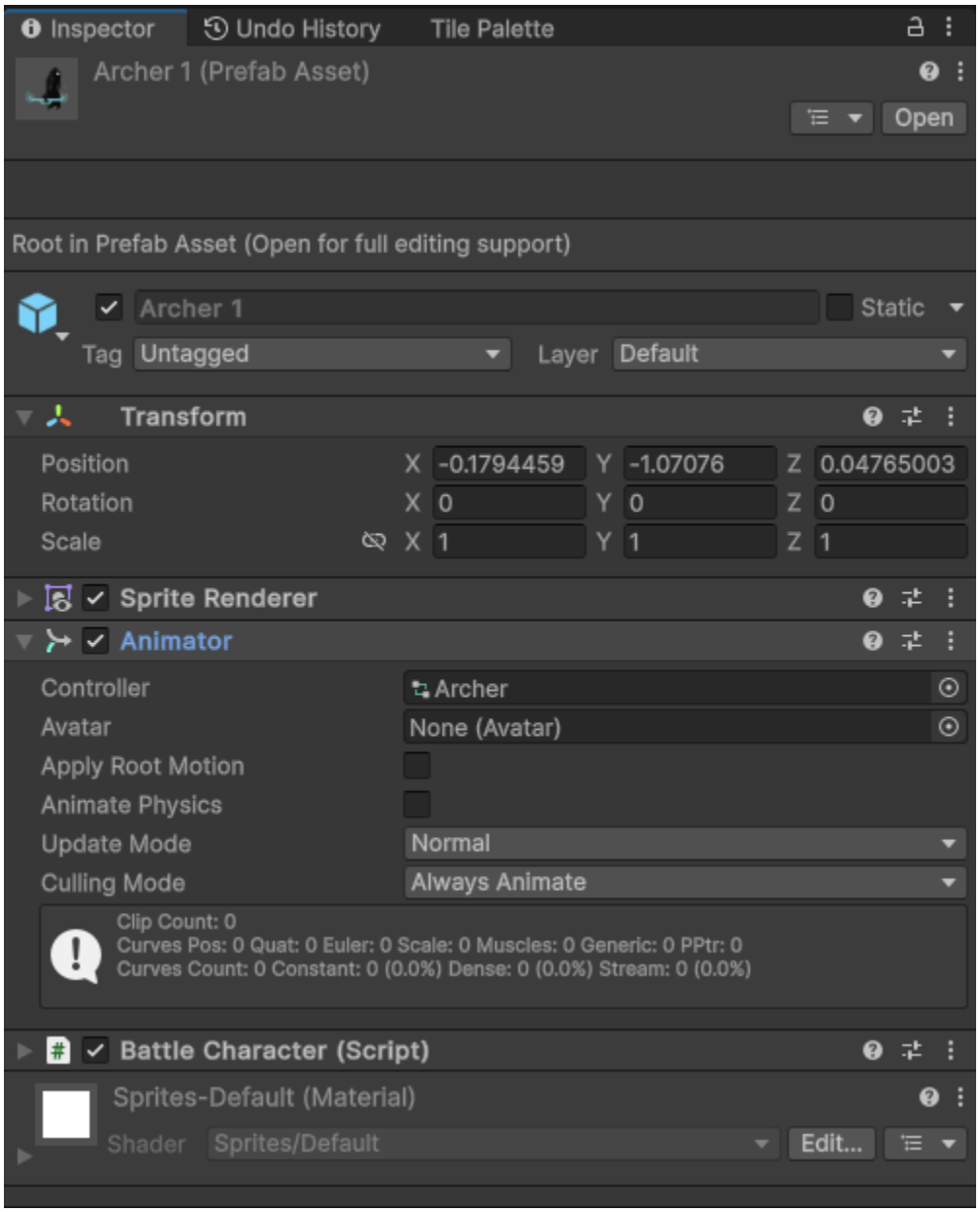
effectPrefab - a field that specifies the visual effect of the ability (more about prefab below)
attachToTarget - whether the effect will fly to the target
effectMoveSpeed is the speed at which it will fly.



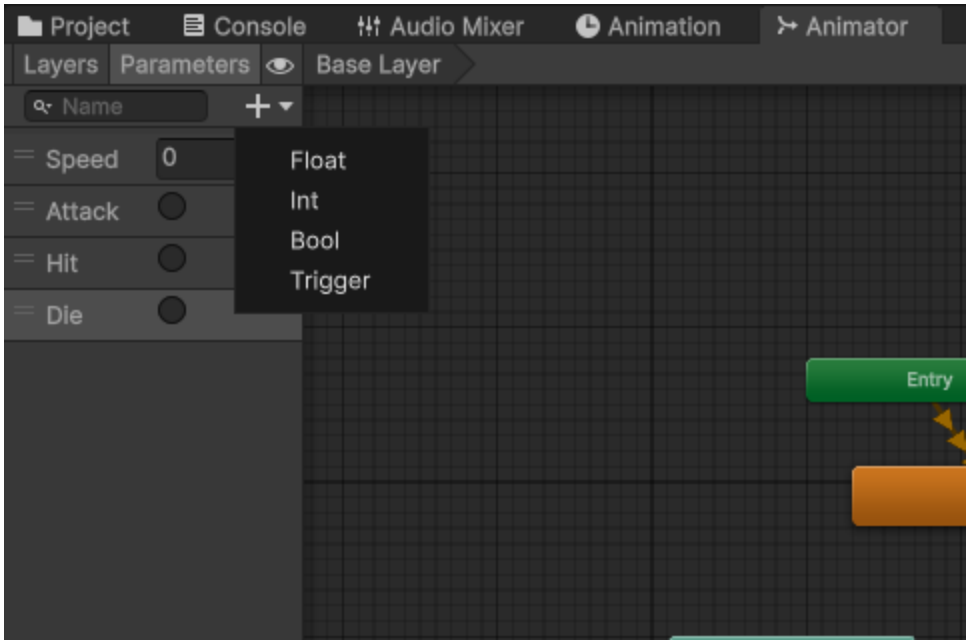
Combo effects are used for combinations of effects. The Other field indicates which effect is combined with it, and the Resulting field indicates the effect that we get at the output.
Trigger - remote field

Animations

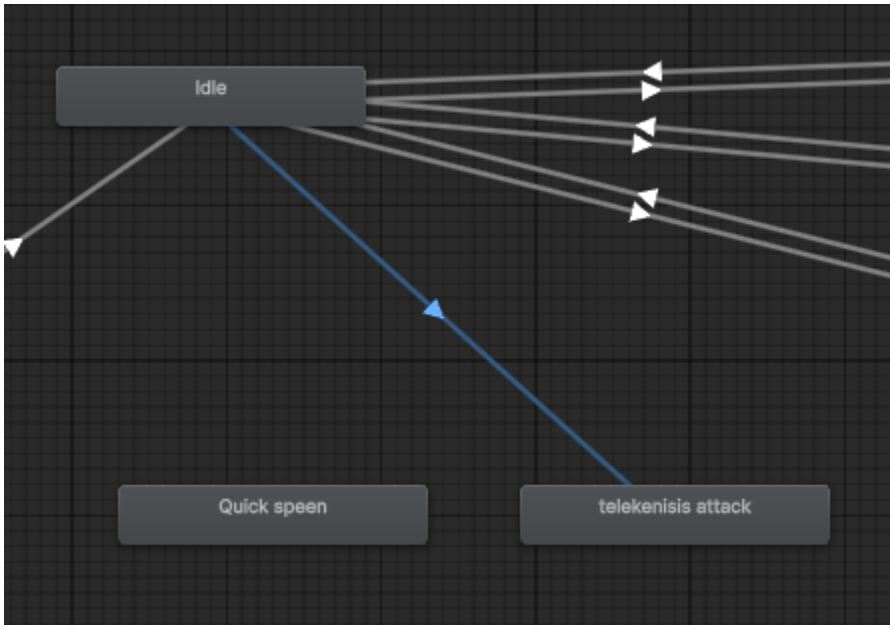
each prefab has an Animator



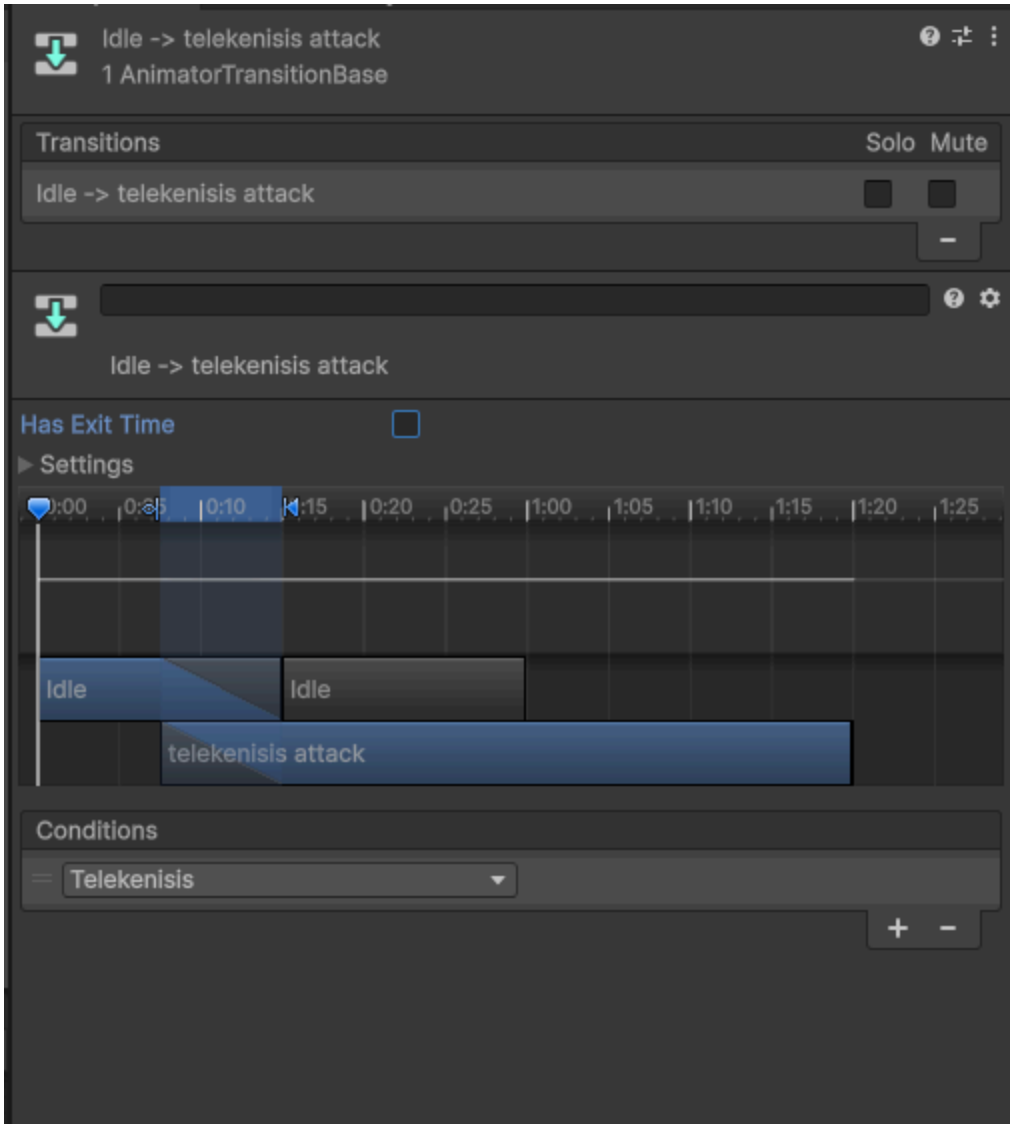
Simply go to the Animator window or double-click in the Controller field.

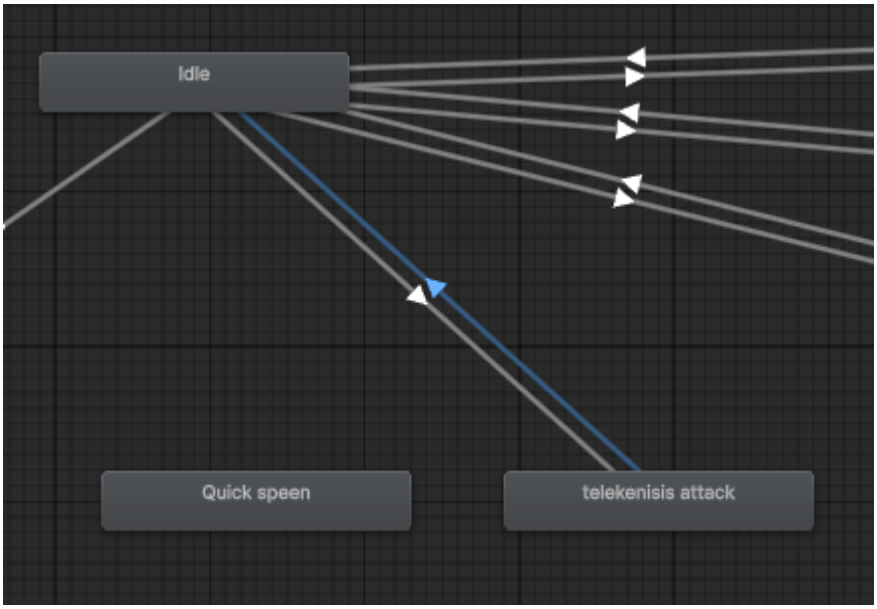


Create a new trigger and name it whatever you want the transition to animation to be called, for example, “Telekenisis”

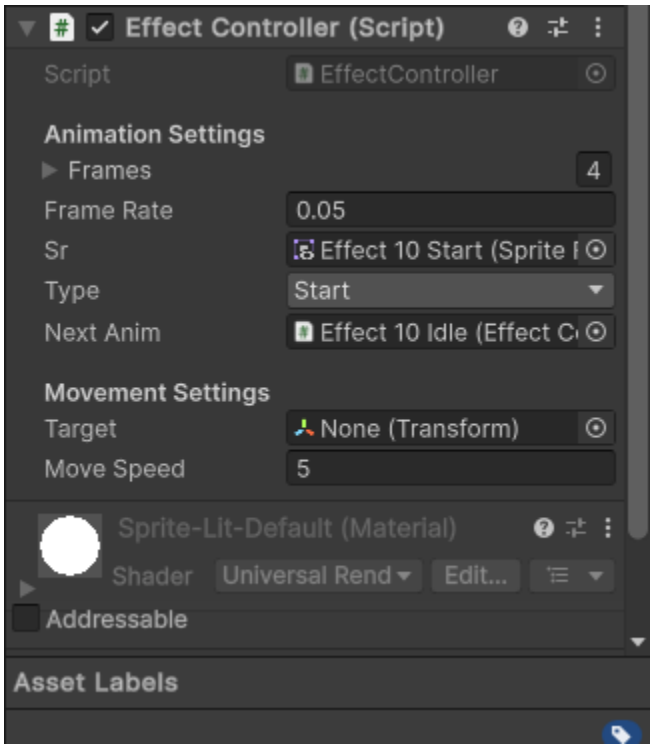


we create a transition from the idle animation to the desired animation,
Click on the arrow to remove Has Exit Time.
and in Conditions we specify the name of the newly created trigger


Idle -> telekenisis attack
1 AnimatorTransitionBase
Transitions
Solo
Mute
Idle -> telekenisis attack
Idle -> telekenisis attack
Has Exit Time
Settings
0:00
0:05
0:10
0:15
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 reverse arrow, but you don't need to do anything with it 

effects

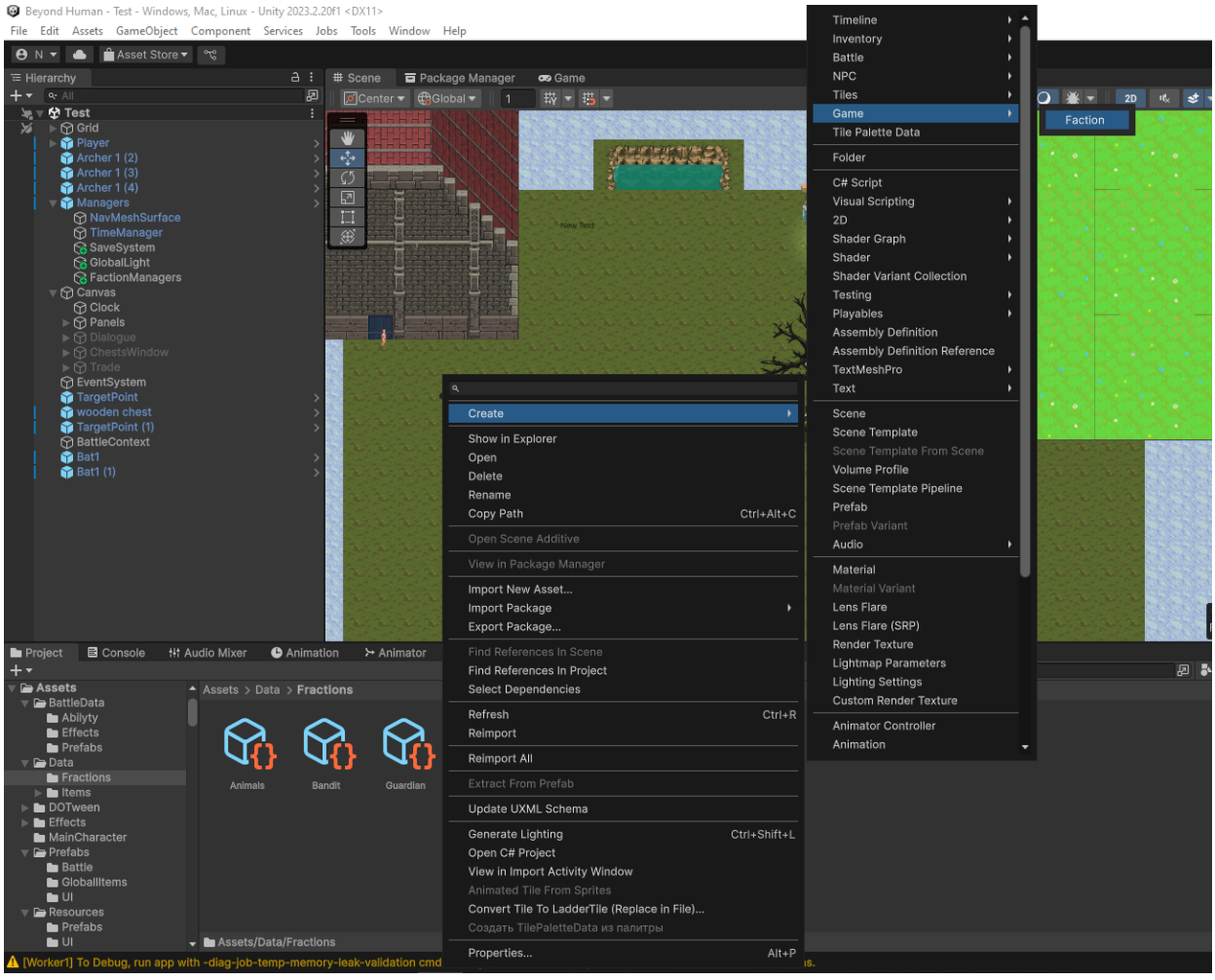


For abilities, you can create effects that will fly out from the character who casts the spell to the character to whom the spell is applied
All available effects are stored in the Effects/Name/Prefabs folder.

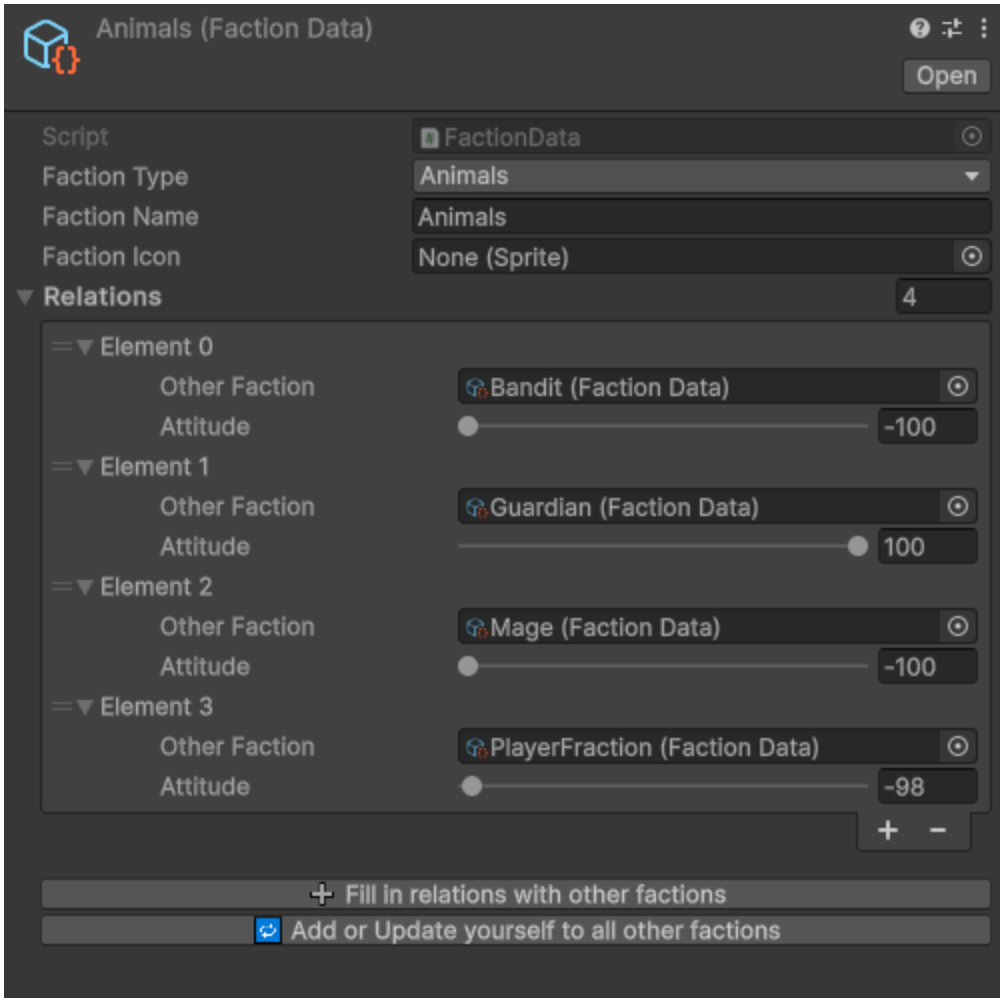
Each field effect has:
frames - effect sprites
frameRate time between sprites (animation speed)
EffectType - effect type
start - will play once, after which the animation from the nextAnim parameter will be played
Idle - the normal animation will play until the target is reached.
End - will be played once after idle if the animation has reached its target (if it is in the nextAnim parameter in the Idle animation)
nextAnim - needed to create a chain of different effects, can be left empty if it is not needed

Factions and battle start

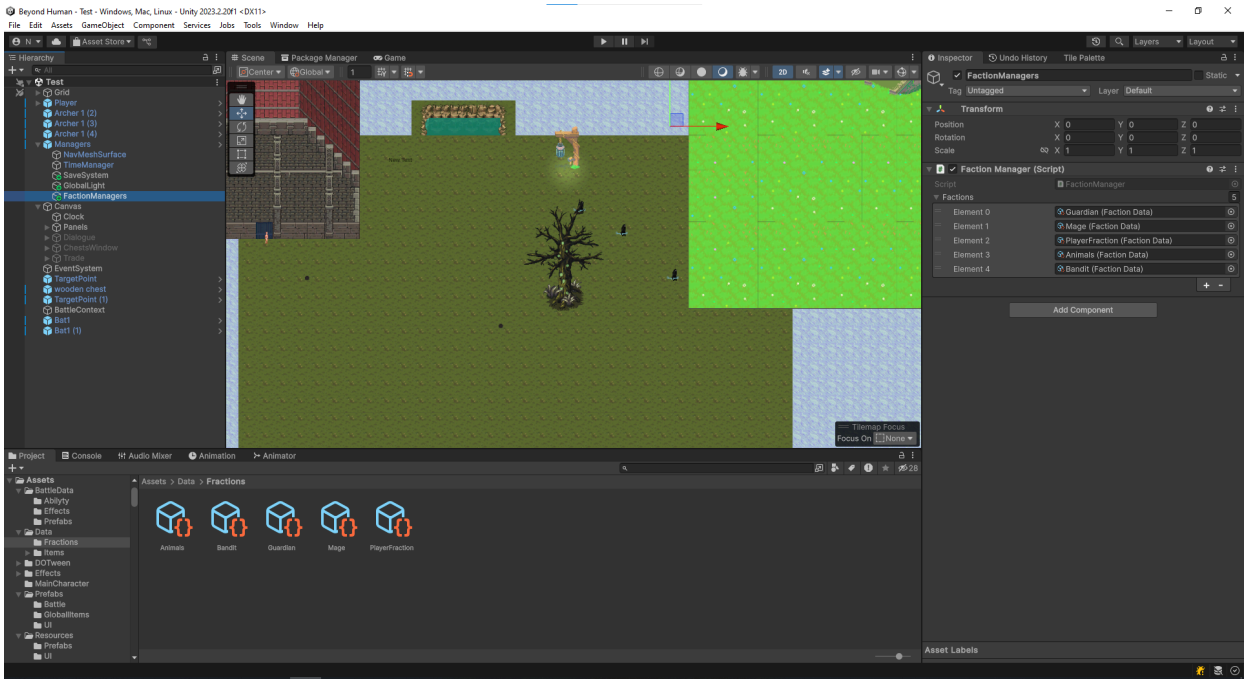
Create Factions



To create a faction, I recommend doing this in the Assets/Data/Fractions folder.
you need to select Create → Game → Faction

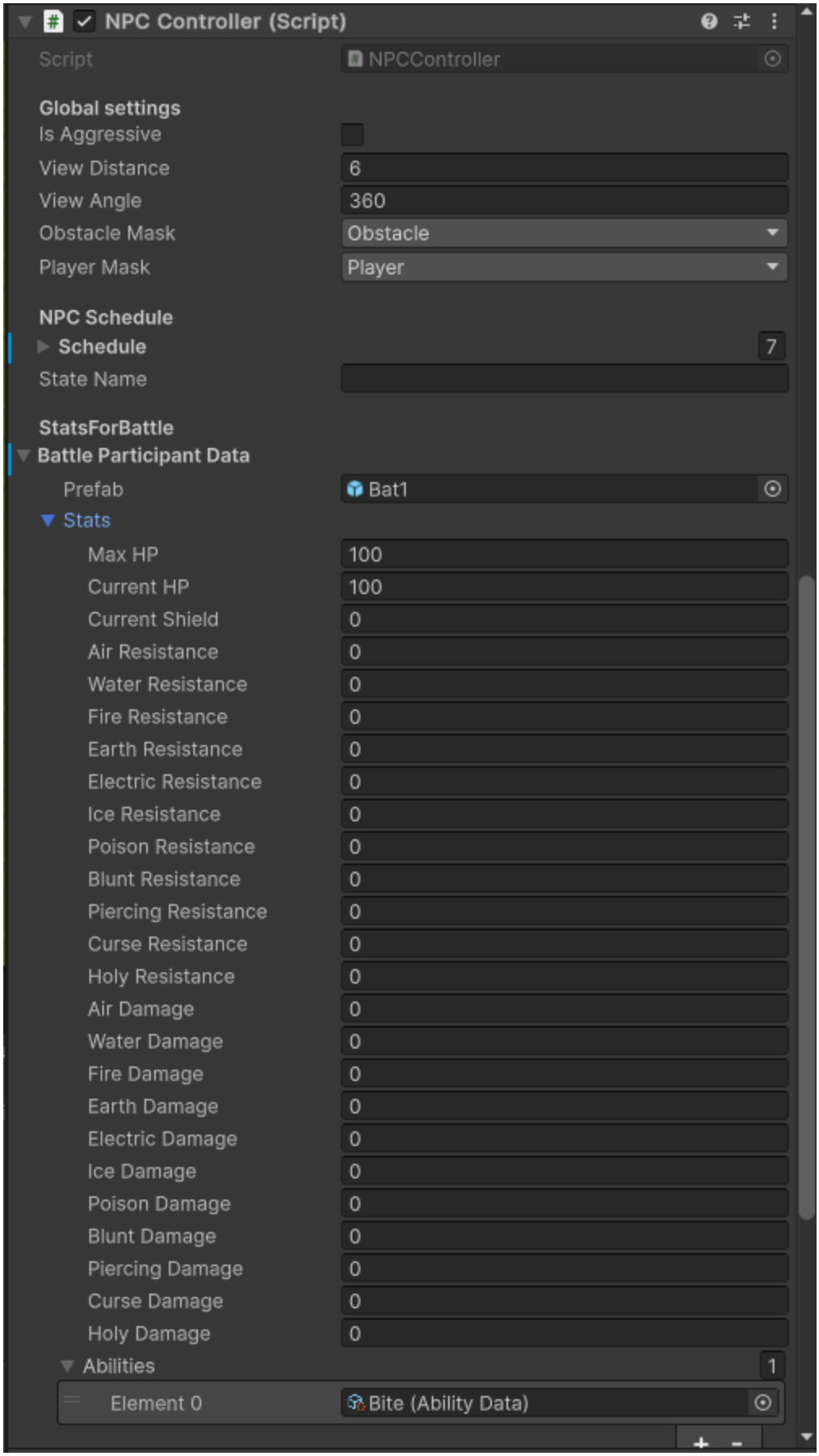


This is what the faction creation window looks like, where we specify the type, name, icon, and relationship with other factions. Each faction must be dragged manually, and their relationship must be specified. It is IMPORTANT that the second faction match the values. For ease of editing and adding, I added two buttons. The first button will cycle through all factions and assign the values specified in the others to the selected one. The second button, conversely, adds itself to all other factions.

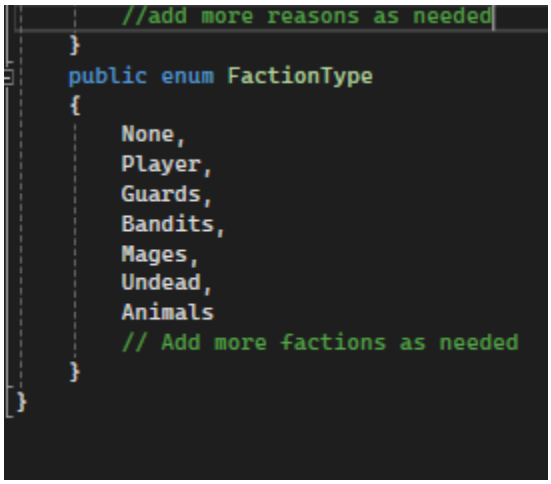


For all factions to work, they must be added to the “FactionManager”

settings to prepare for battle

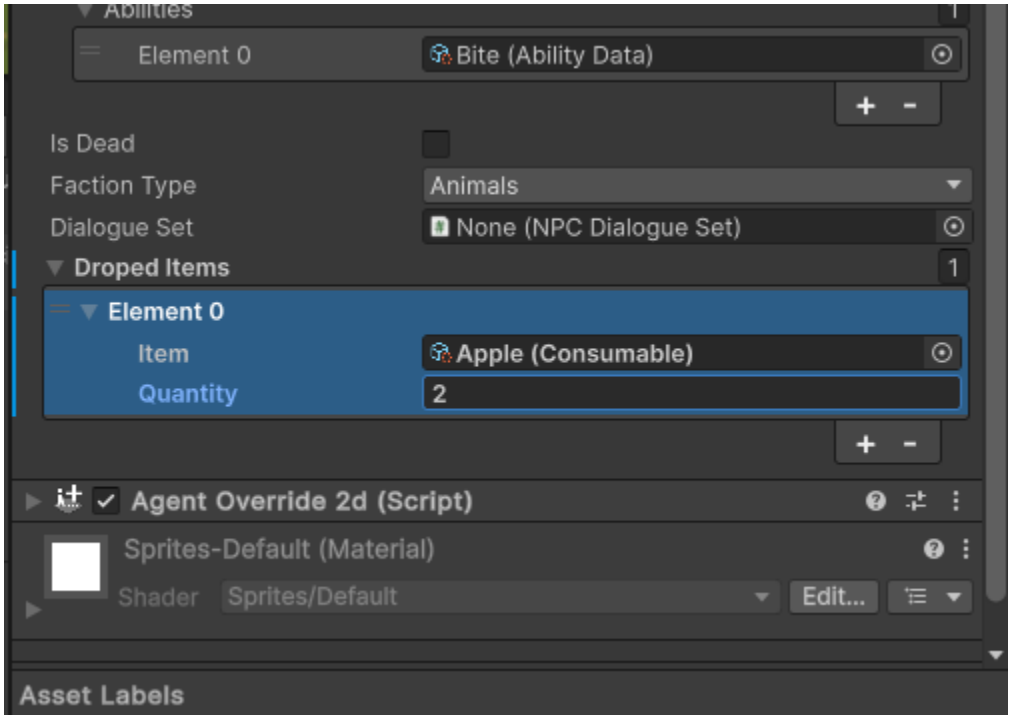

NPC Controller (Script)
Script
NPCController
Global settings
Is Aggressive
View Distance
6
View Angle
360
Obstacle Mask
Obstacle
Player Mask
Player
NPC Schedule
Schedule
7
State Name
StatsForBattle
Battle Participant Data
Prefab
Bat1
Stats
Max HP
100
Current HP
100
Current Shield
0
Air Resistance
0
Water 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)

This is what the updated NPCController looks like now.
where prefab is a link to the prefab used in the battle scene
Stats is how much health you have at the start of the battle.
and Abilities is a list of abilities that this NPC will have
The IsDead checkbox is used to store information about whether an NPC is alive, and if he is dead, he will not be on the scene the next time he loads (I do not recommend touching this)
Fraction Type is needed to assign an NPC to his faction.
NPCID is a unique identifier for each NPC used to save and load each NPC and is automatically assigned when a new NPC is created.



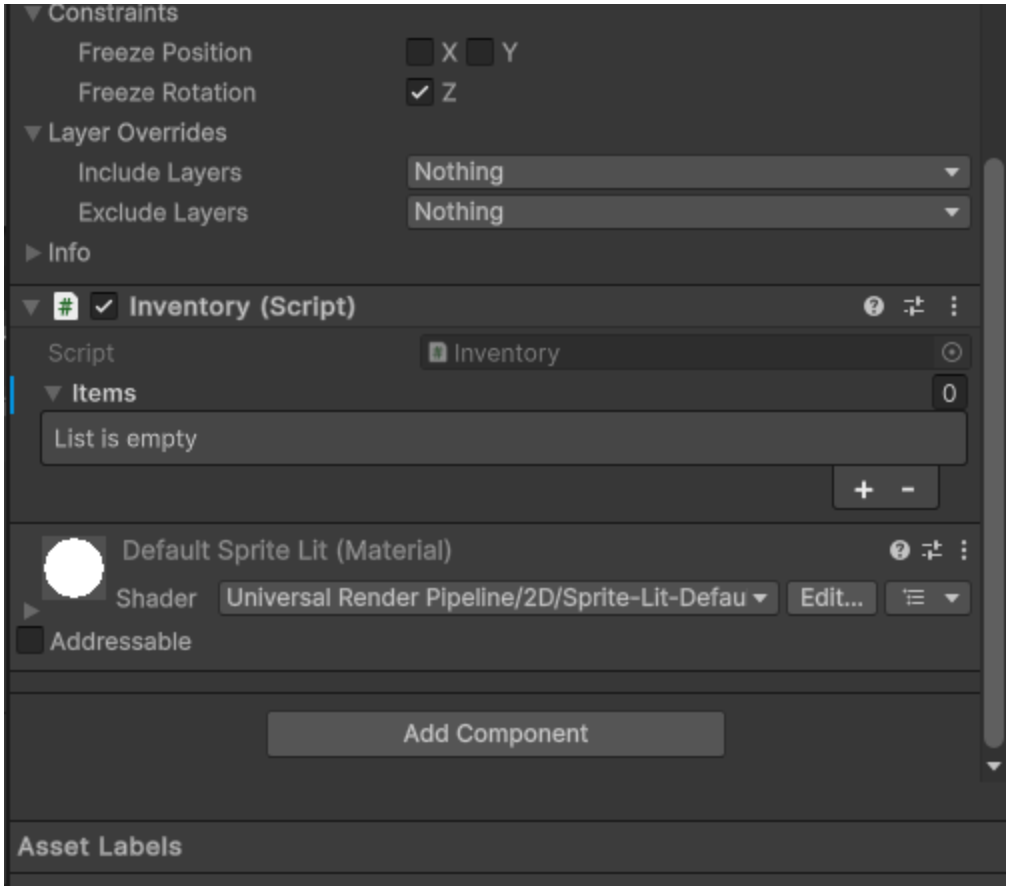
To expand or change the list of factions, you need to open the NPCEnums script in the Assets/Scripts/Abstracts folder.
Find FactionType there and add new faction names to this list

Party window and inventory



Dropped items - items that will drop from an NPC when he dies in battle

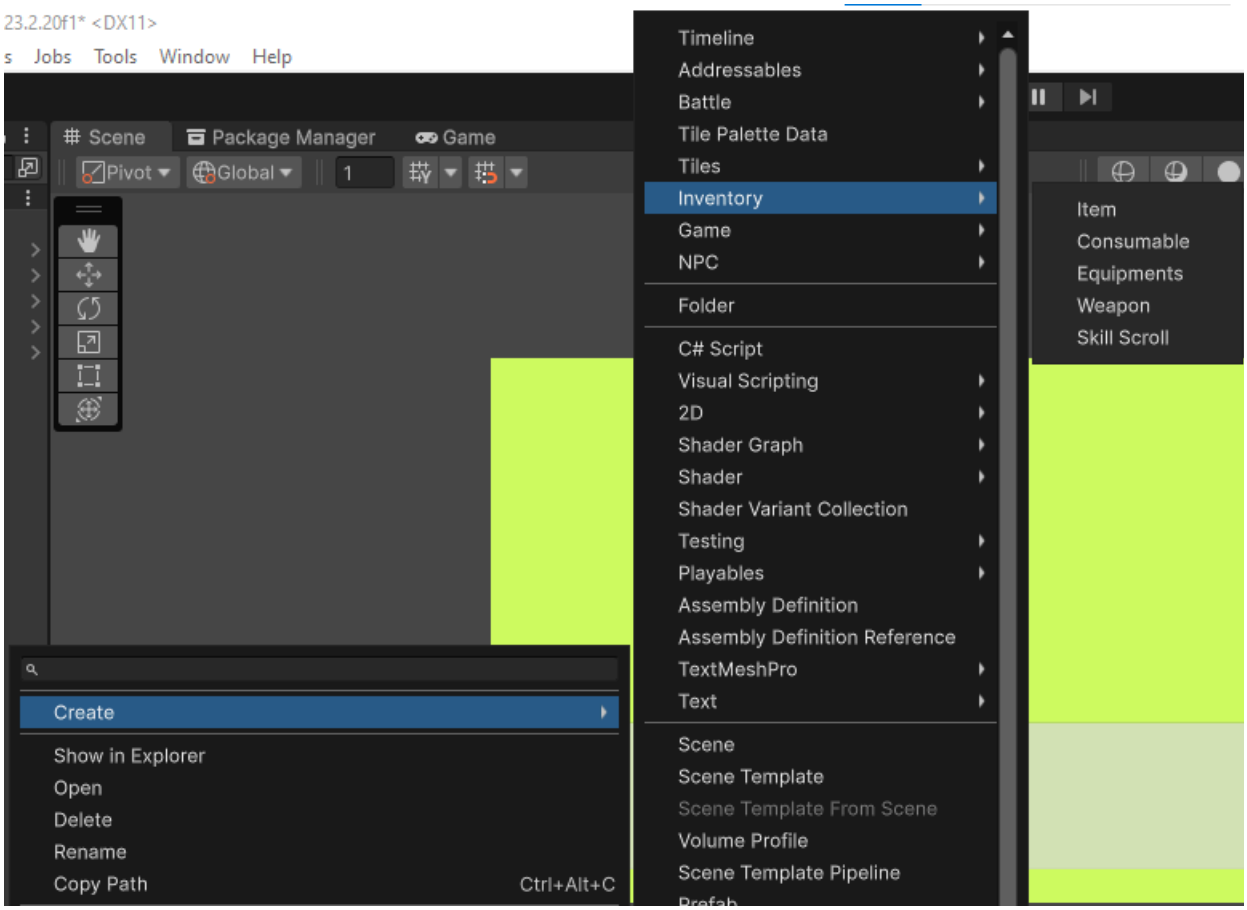
- Item drop
- Quantity - the number of items that will drop



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

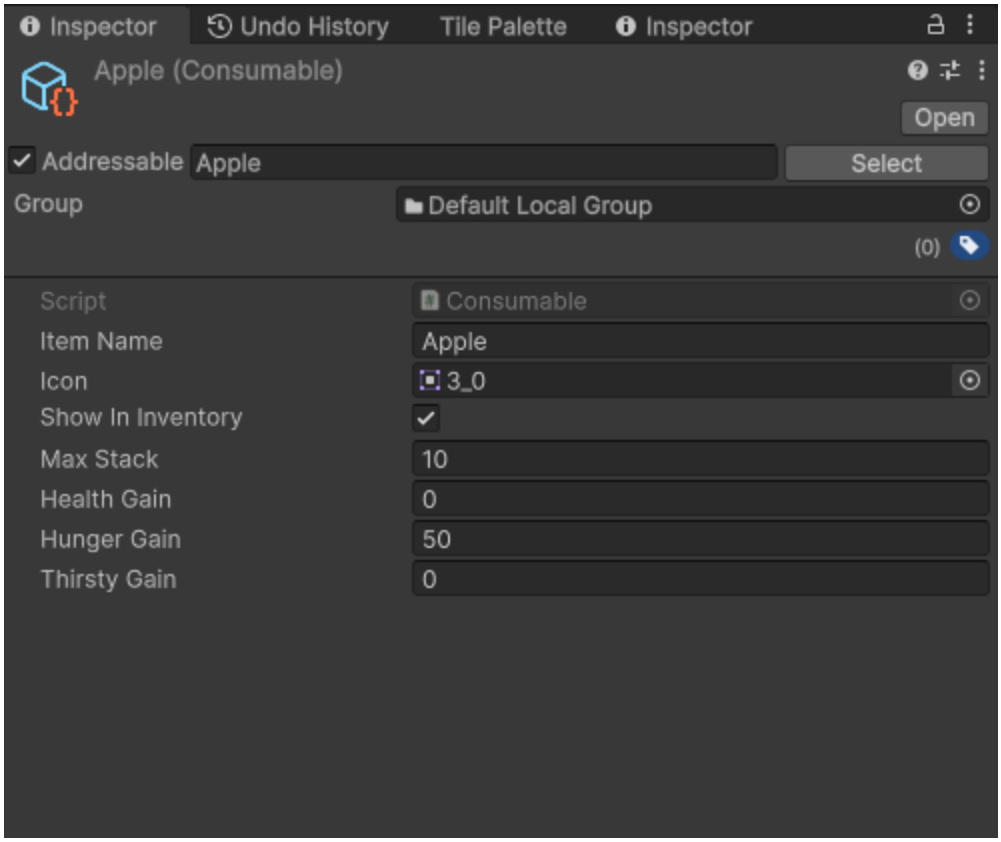
Crafting items

Folders for storing items: Assets/Data/Items

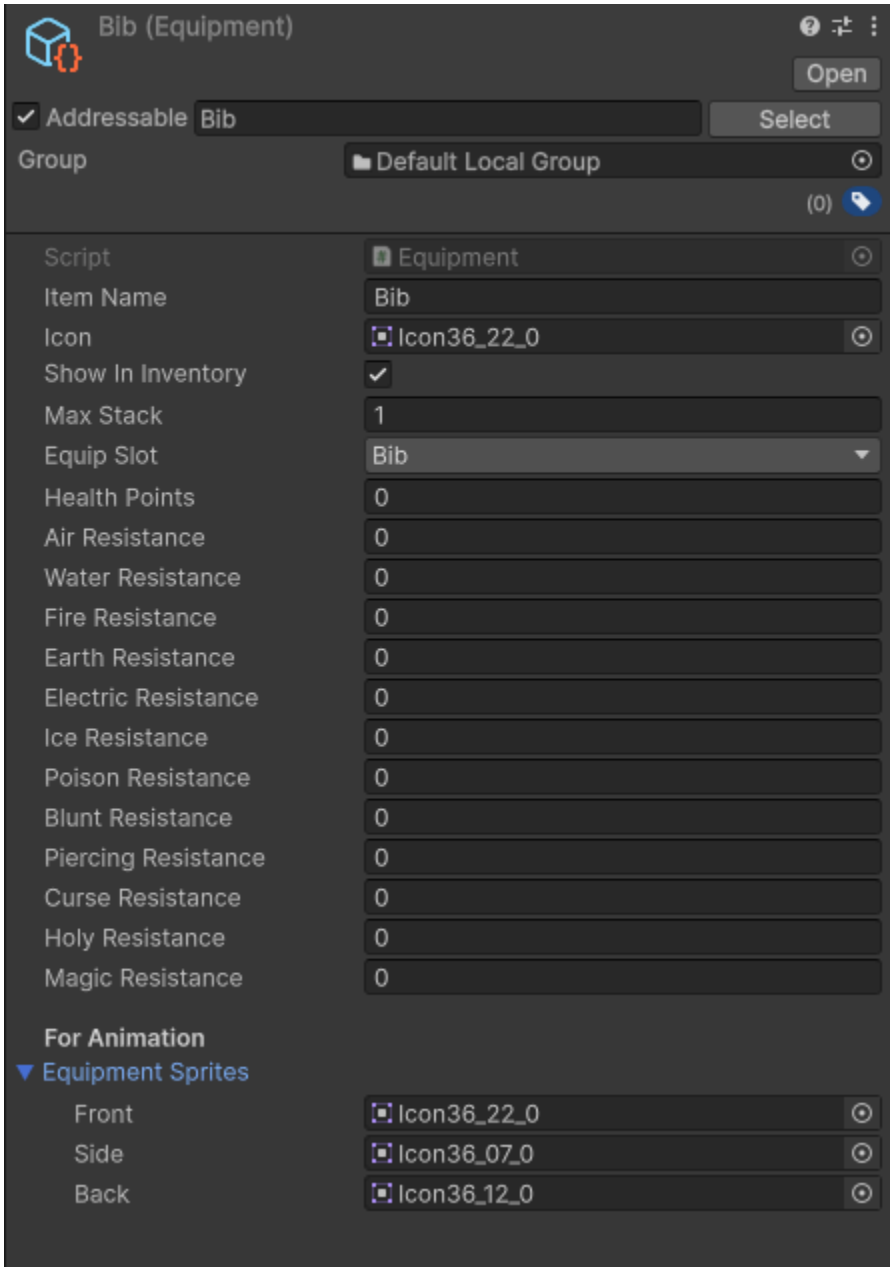


Items that can be created are located in Create->Inventory

Consumables:

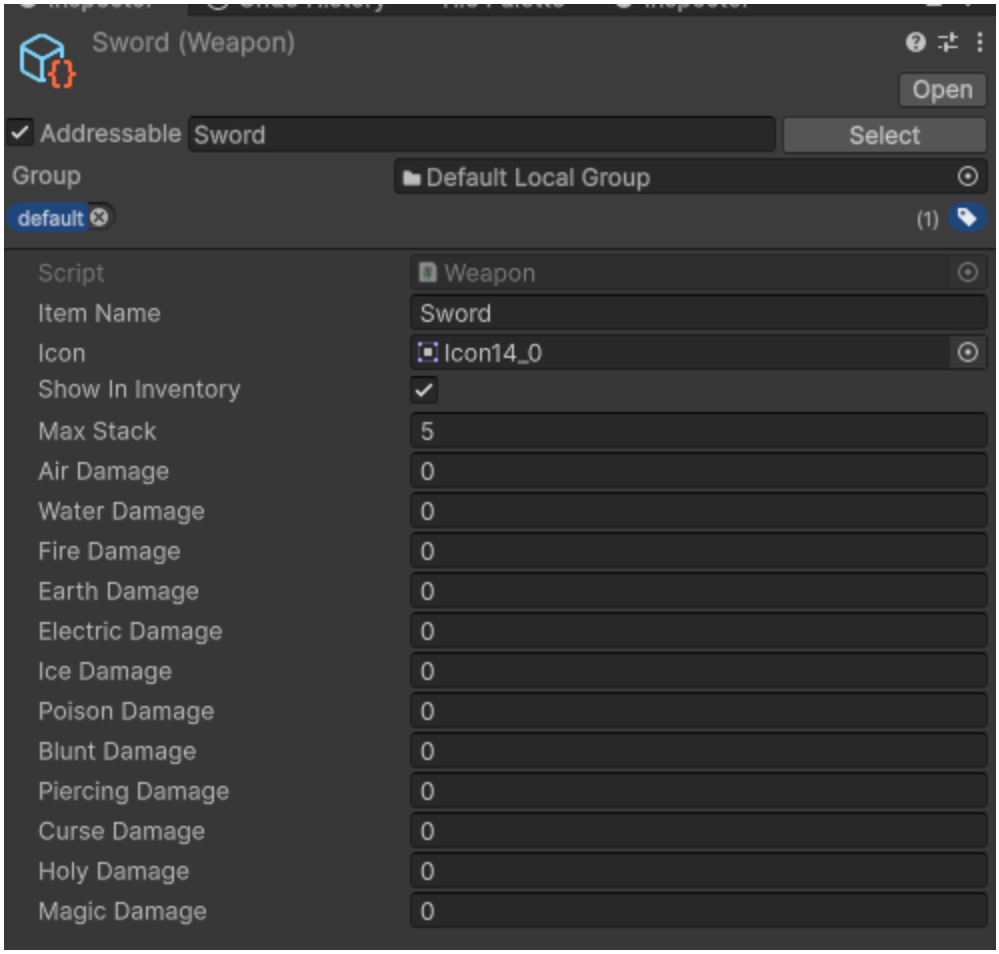

Inspector
Undo History
Tile Palette
Inspector
Apple (Consumable)
Open
Addressable
Apple
Select
Group
Default Local Group
(0)
Script
Consumable
Item Name
Apple
Icon
3_0
Show In Inventory
Max Stack
10
Health Gain
0
Hunger Gain
50
Thirsty Gain
0

Equipment:

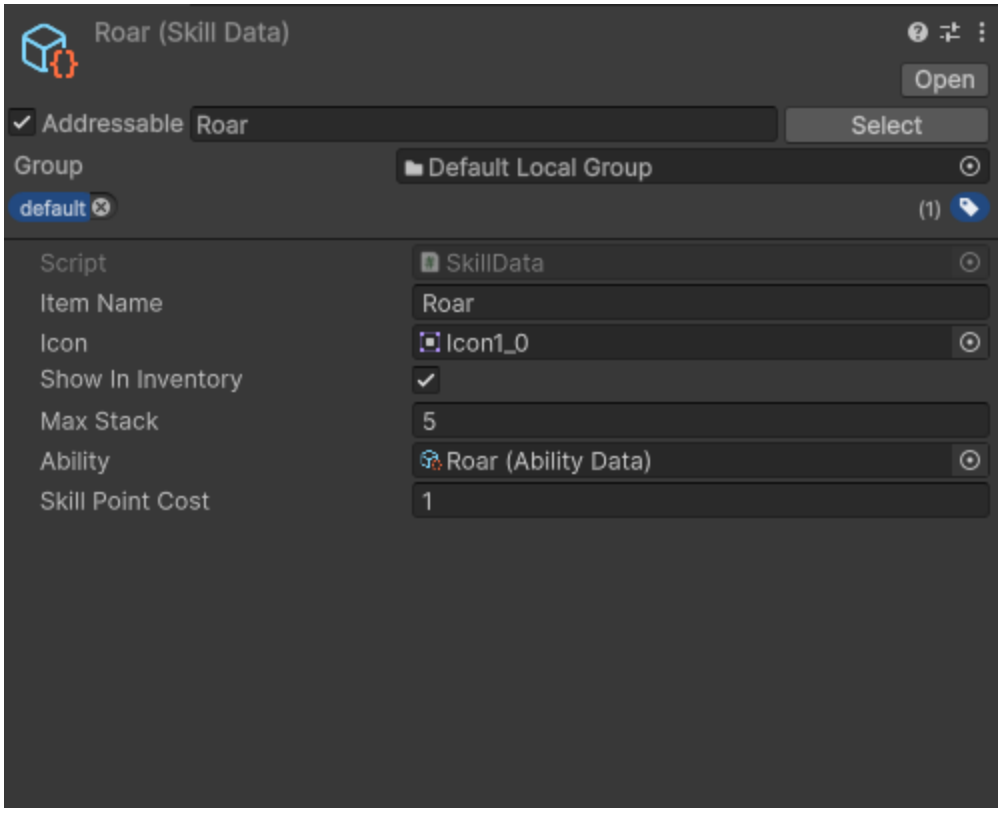


equipSlot - which slot this item belongs to
Equipment sprites are needed to assign different directions for animation on a character.

Weapon:



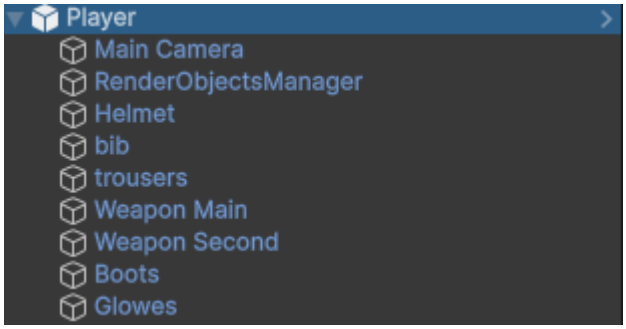
Capabilities:



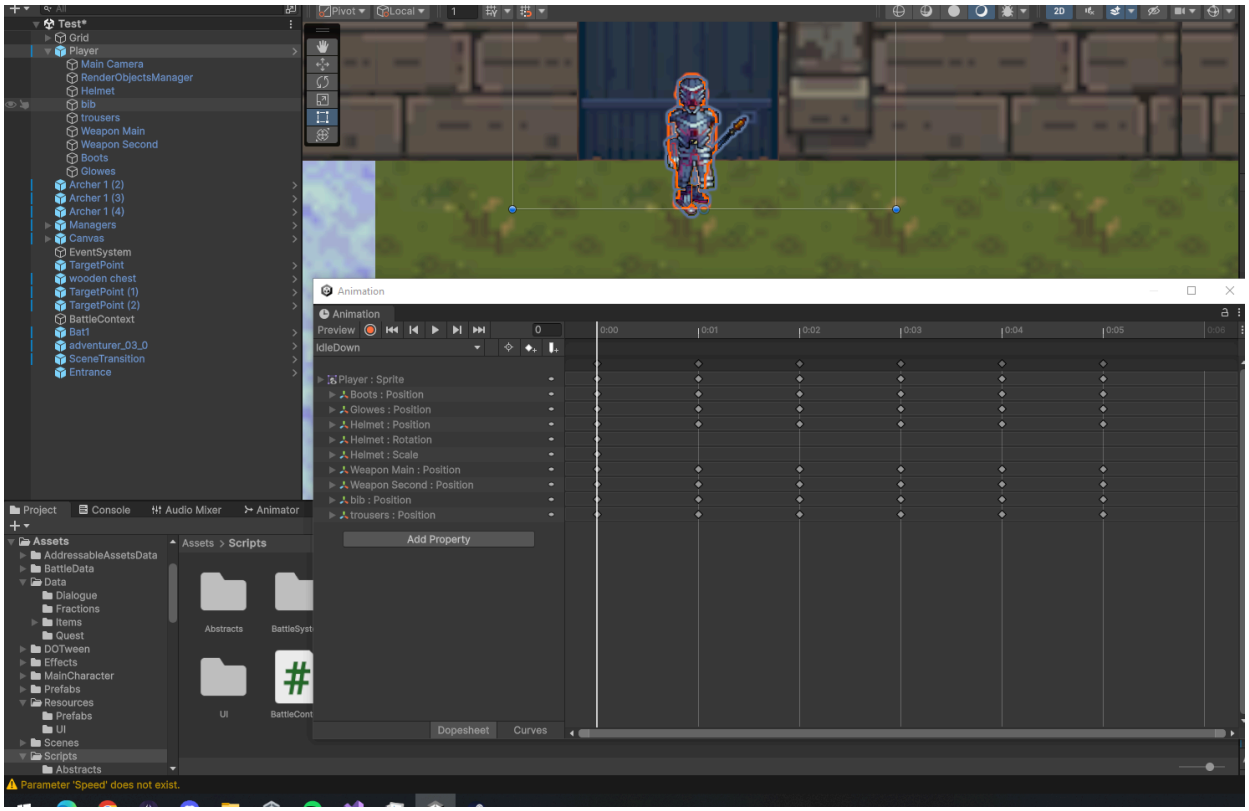
Ability - an indication of the ability from Assets/BattleData/Ability that is used in the battle scene

Player Animation

The player contains child elements for each piece of equipment



To set up the player's animation correctly, you need to select the desired animation in the animation window.

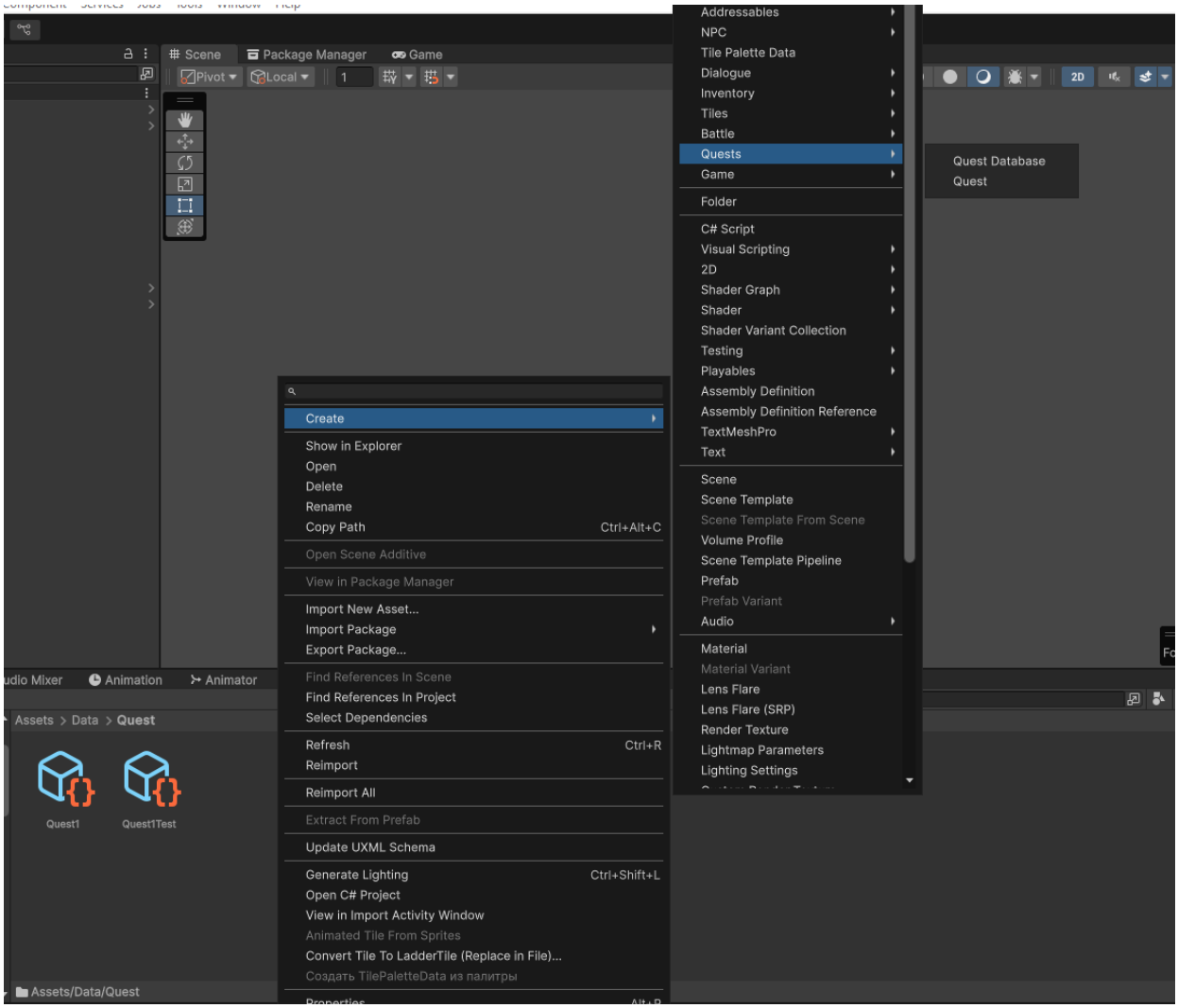


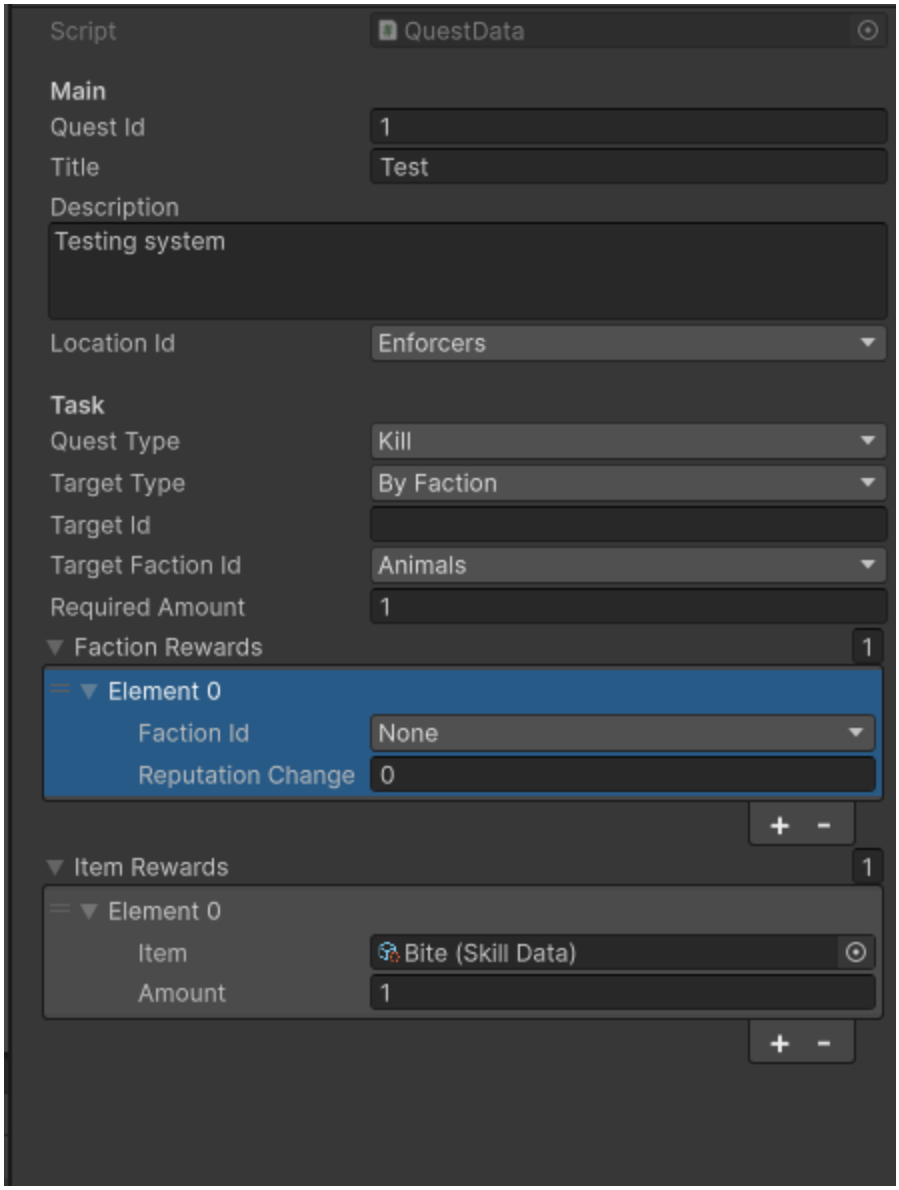
and for each item, add the corresponding keys in the place where they should be located. This can be done either by simply assigning keys or through the record button.

More details about creating animations can be found here: Unity - Creating an Animation Clip

Quests

To create quests in a folder, right-click and create a new quest along the specified path.





The following fields are available for quest customization:
ID for the quest to be able to influence the environment, each quest must have this parameter unique
Title - the name of the quest
LocationId is the location to which the quest belongs.
QuestType, what type of quest is it?
TargetType is required to filter by faction or name (only used for assassination type quests)
TargetId - depending on what is selected in QuestType

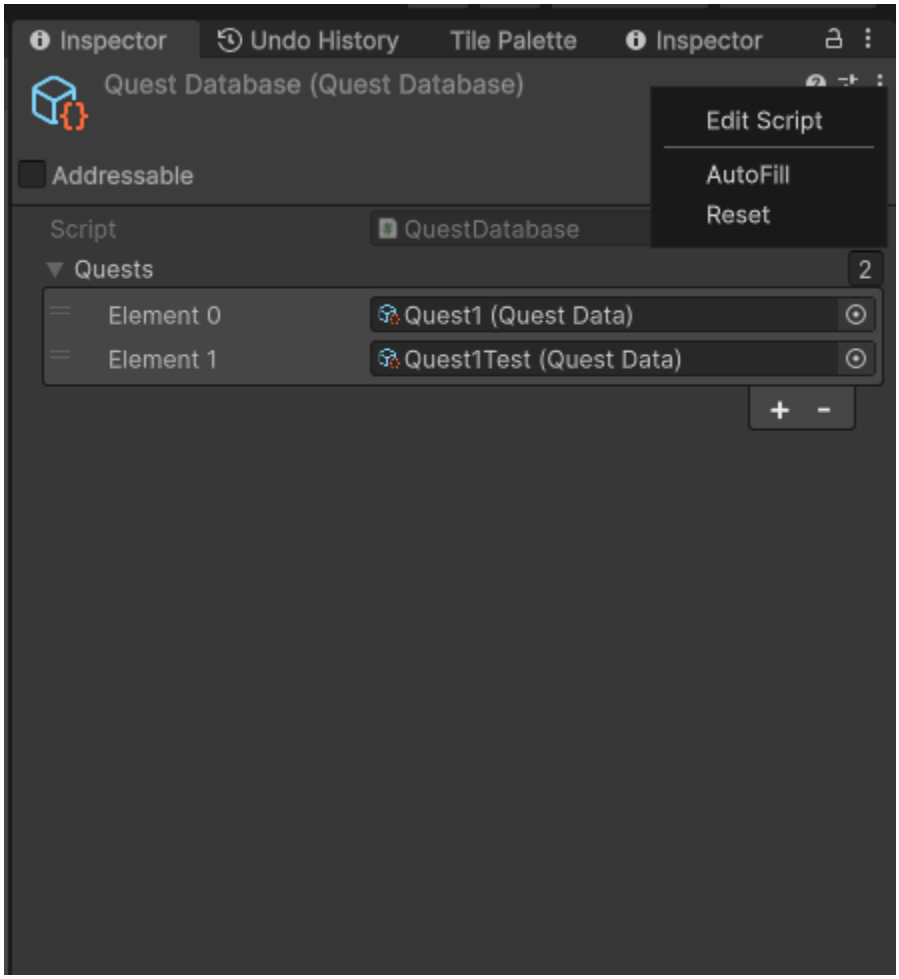
Kill - the name of the creature to be killed (Coincides with the CharacterName field of the NPCController)

Collect - The name of the item to be collected (matches the ItemName field in any of the items)

Talk - the name of the character with whom you want to start a dialogue (Coincides with the CharacterName field of the NPCController)

Explore - you need to visit a specific location (must match QuestExploreTriger ZoneID)

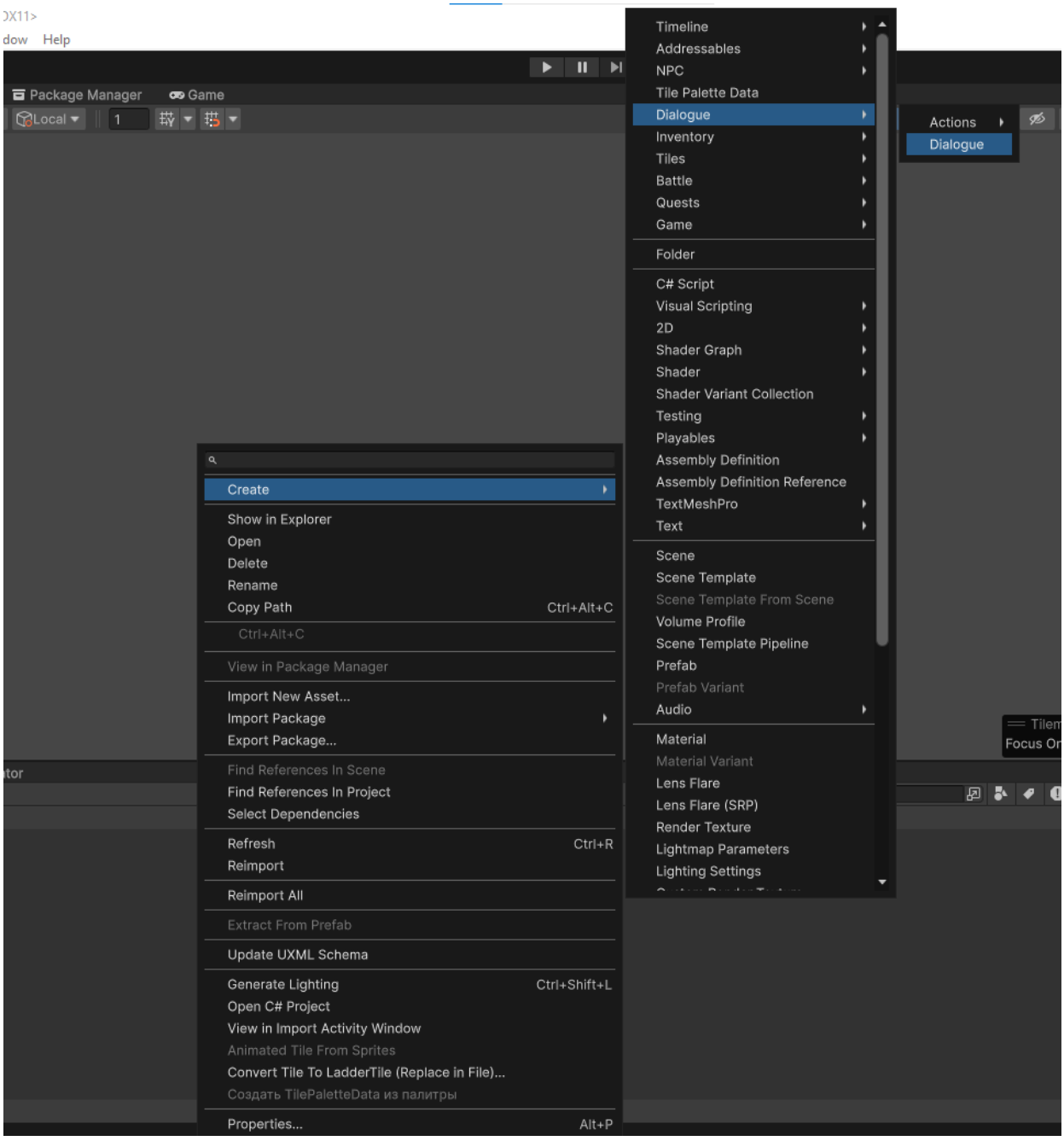
All quests should be stored in the QuestDatabase, which is located in the Data folder.



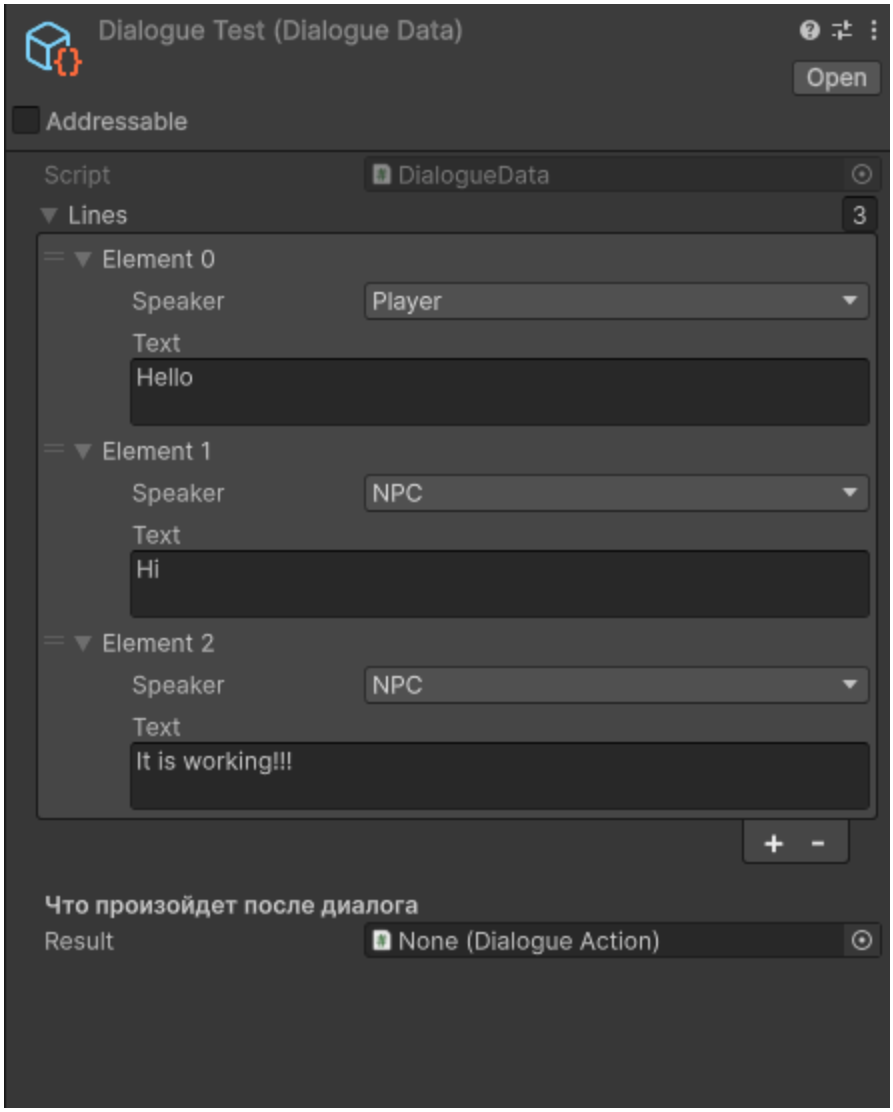
If you click on the three dots in the upper right corner, you can find the AutoFill button, which will attempt to add all the quests in the project.

Dialogues

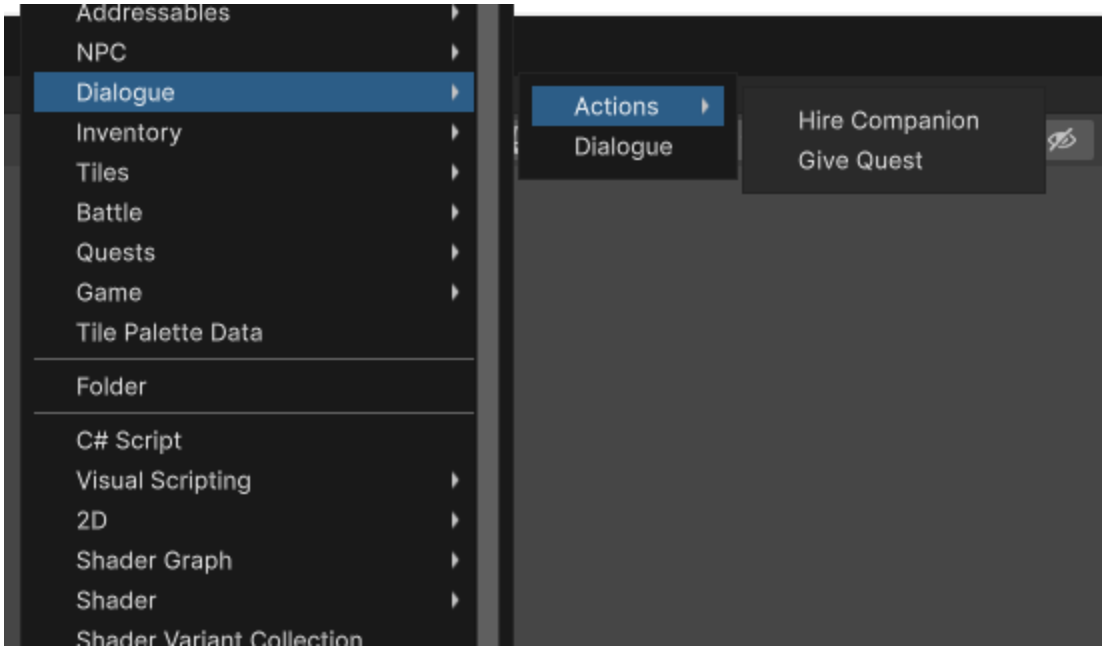
The path to create a dialog tree:



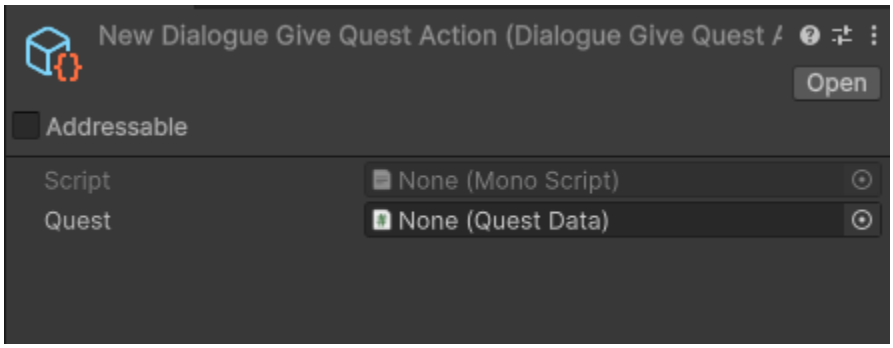
The dialog has lines where the sequence of displayed sentences goes



At the end of the dialogue, you can assign different events to trigger

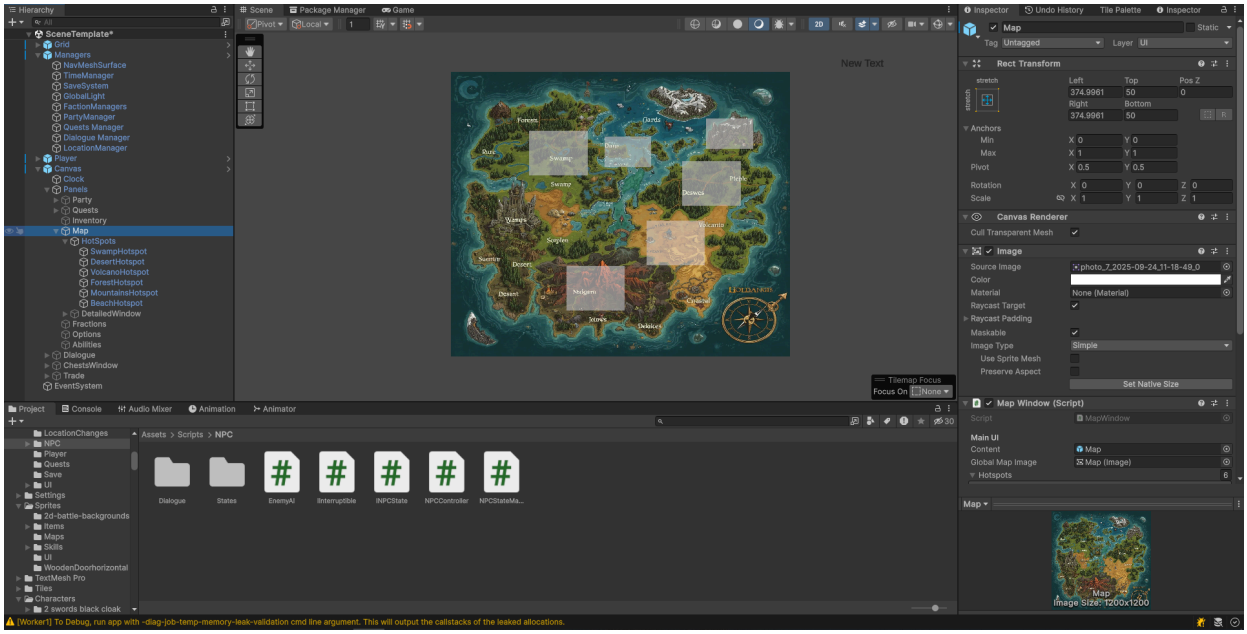


There are two events available,
Where Hire Companion is responsible for hiring an NPC with whom a dialogue is conducted (you can create one instance and assign it to all dialogues that require such a need)
and Give Quest, responsible for issuing quests

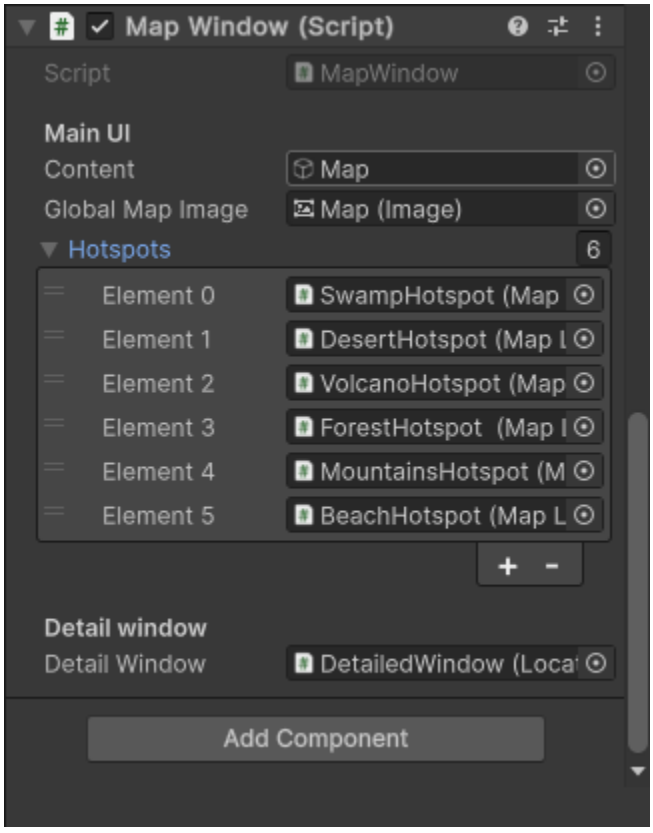


in which it is enough to indicate the quest that will be assigned

World Map

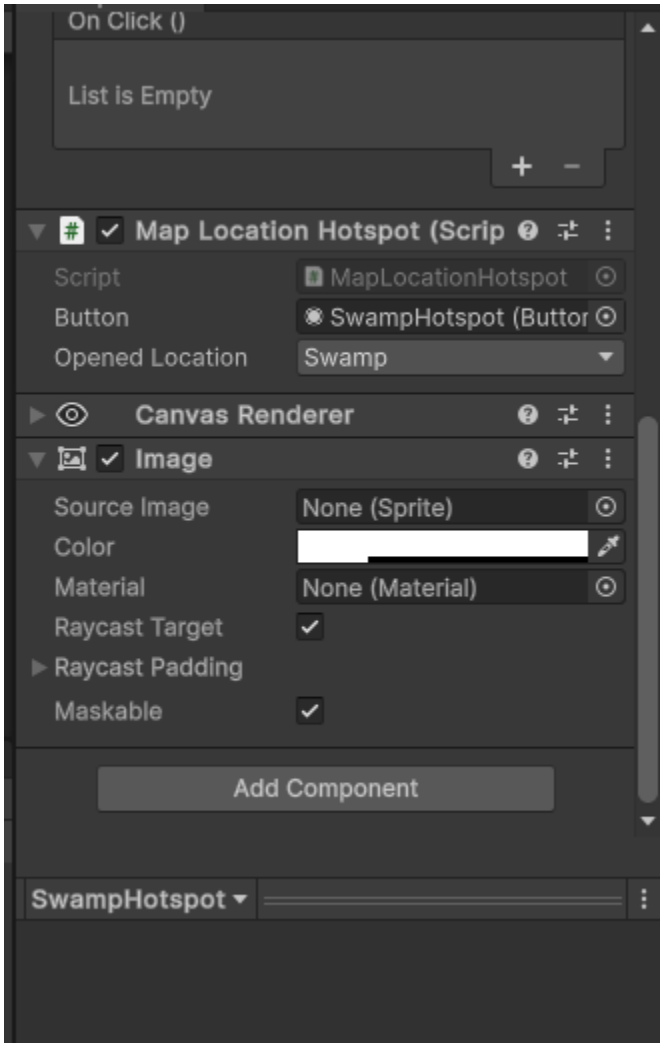


The “Map” component has an Image component in which you need to specify the world map sprite. It can also be specified in the WorldDataBase component in the WorldMap parameter.



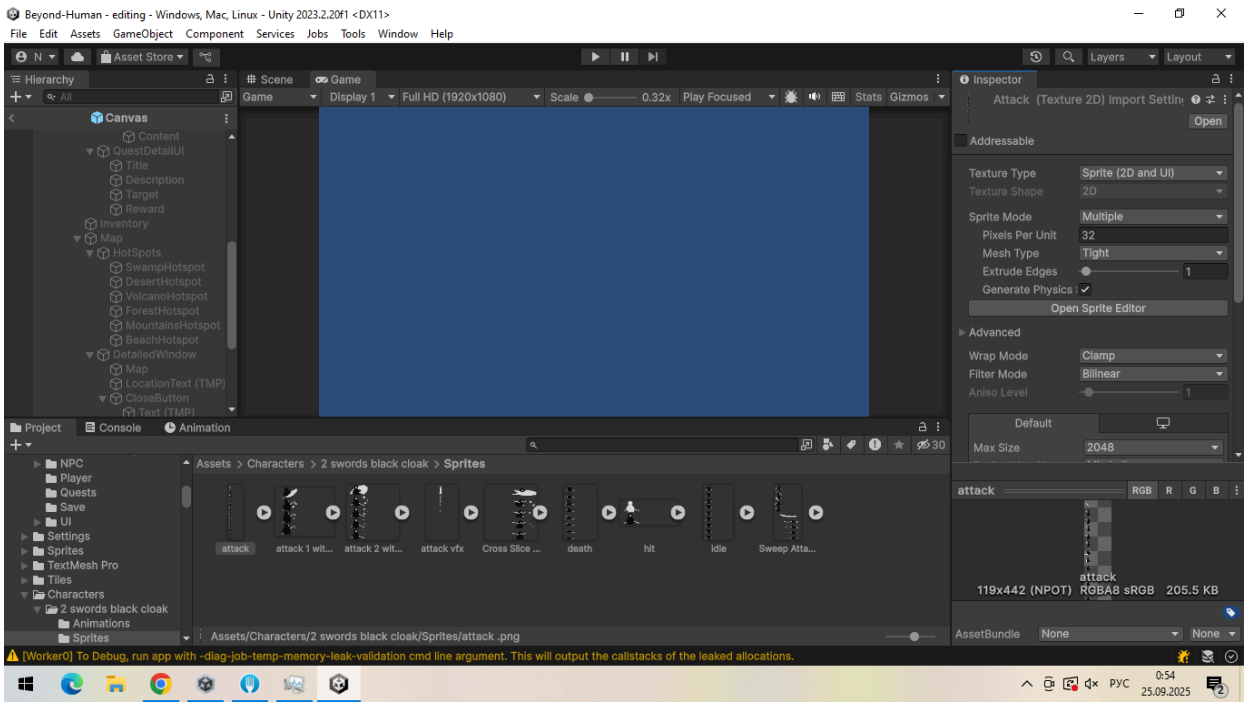
In parameters Hotspots

You need to specify the transition buttons that need to be added to the Map/Hotspots child object
which serve to deploy a more detailed location

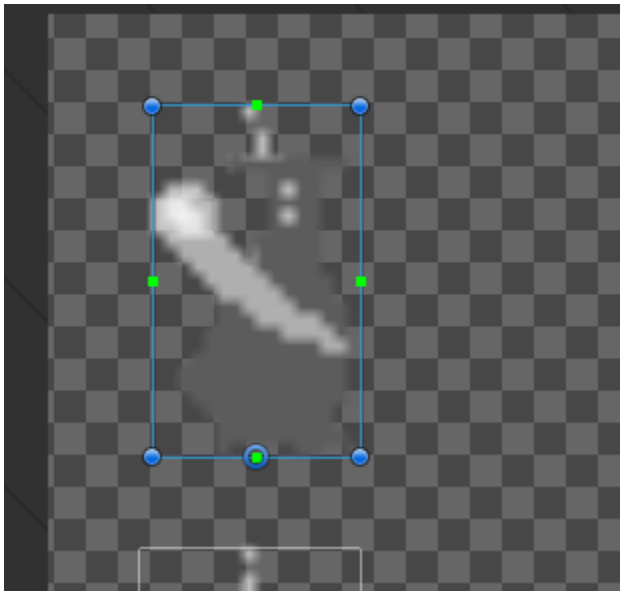


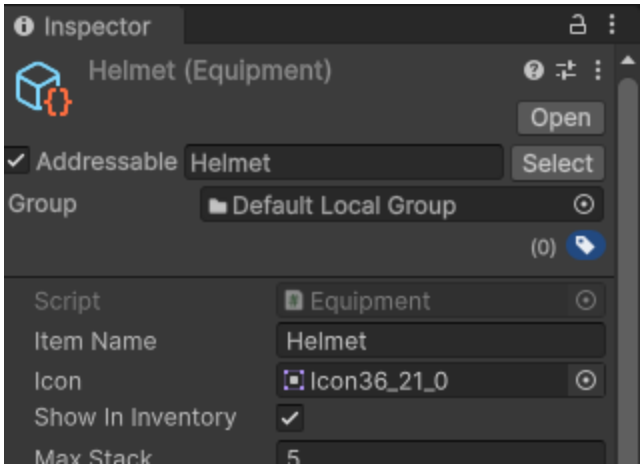
Hotspot has a LocationId parameter that determines which map will open when you click the button (it takes the map sprite from the WorldDataBase, which is specified in each location in the Detailed Map).
You can freely change the location and size of the hotspot

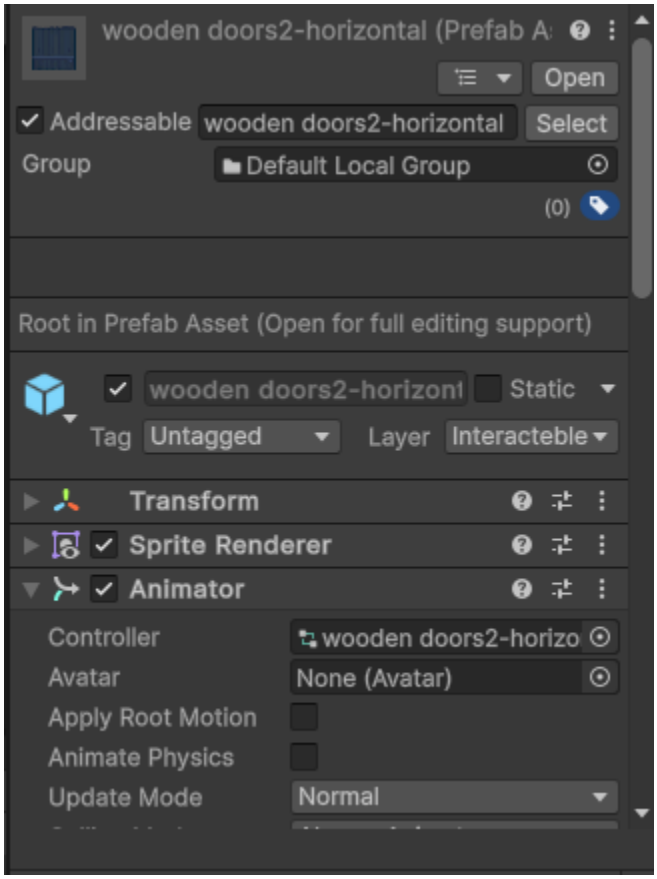
Additional information



All units and objects in the project are displayed depending on their pivot; whose pivot(Blue circle) is higher will be displayed behind. To configure the pivot for each NPC object or just an object on the scene, you need to select the desired sprite, select the Sprite Editor in it, and move it to the desired location.



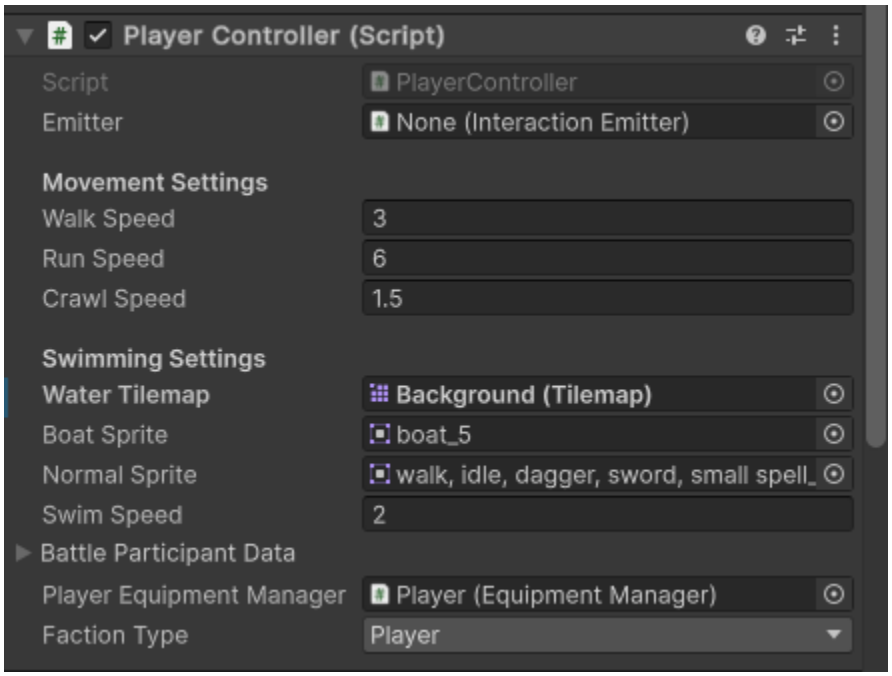




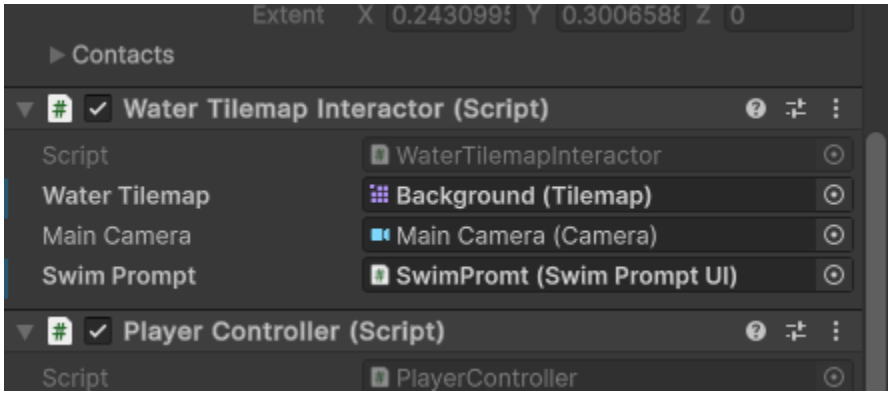
All units and items have an Addressable field, which will be filled in automatically (but you need to be careful because it may not always save correctly)
The name of the item (in the itemName/characterName fields) must exactly match the name in the Addressable field; this is necessary for the save and load system to work.

Player Control:

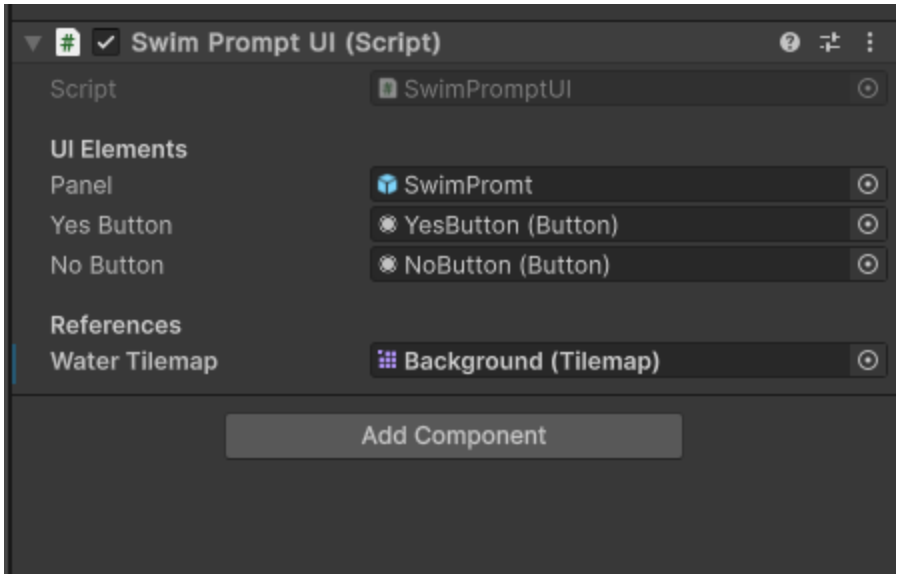
The player has a Player Controller component, which is entirely responsible for character behavior, specifically running/walking, and climbing stairs.



WaterTileMap - points to the card that is responsible for water



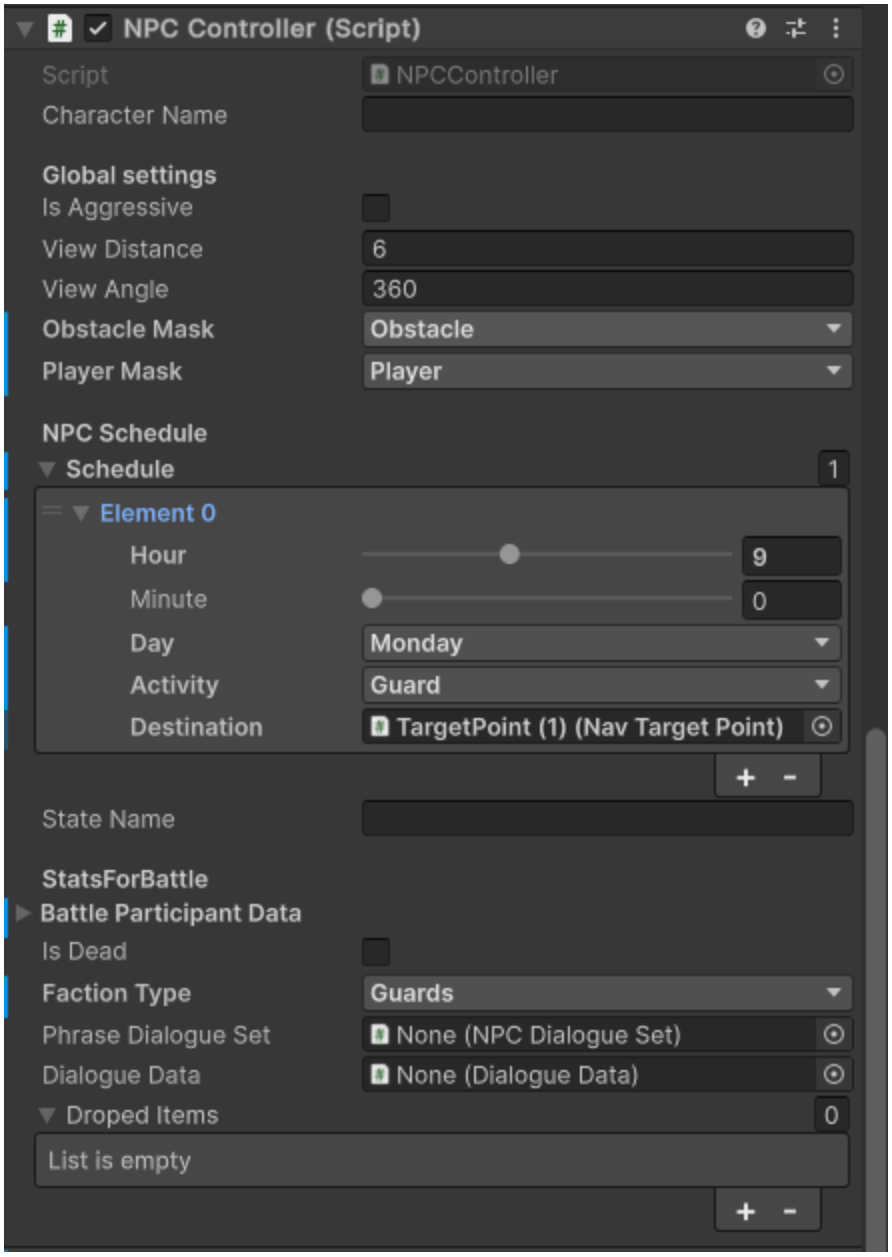
WaterTileMap - points to the card that is responsible for water
mainCamera - indicates the main camera (which follows the player)
swimPromt - A panel in Canvas that pops up to invite the player to swim.



WaterTileMap - points to the card that is responsible for water

NPC Behavior

Each NPC has a component responsible for the character's behavior logic.



The following dictionary is responsible for the set of NPC behaviors:

```
}

private void InitializeStateFactory()
{
    _activityStateFactory = new Dictionary<NPCActivityType, Func<ScheduleEntry, INPCState>>
    {
        { NPCActivityType.Idle, _ => new IdleState(this) },
        { NPCActivityType.Guard, entry => CreateGoTo(entry, () => new GuardState(this, entry.destination.transform)) },
        { NPCActivityType.Patrol, entry => CreateGoTo(entry, () => new GuardState(this))},
        { NPCActivityType.Work, entry => CreateGoTo(entry, () => new WorkState(this)) },
        { NPCActivityType.Sleep, entry => CreateGoTo(entry, () => new SleepState(this)) },
        { NPCActivityType.Trade, entry => CreateGoTo(entry, () => new TradeState(this)) },
        { NPCActivityType.Wander, _ => new RoamState(this) },
        { NPCActivityType.Hide, entry => CreateGoTo(entry, () => new HiddenState(this)) },
        { NPCActivityType.Chill, entry => CreateGoTo(entry, () => new ChillState(this, GetDestination(entry))) },
        { NPCActivityType.Hunt, _ => new HuntState(this) },
    };
}
```

There is also NPCActivityType, which contains a list of states for NPCs that are displayed in the inspector when constructing behavior logic.

```
public enum NPCActivityType
{
    Sleep,
    Work,
    Idle,
    Wander,
    Trade,
    Guard,
    //Eat,
    //FightClub,
    //Fishing,
    //Travel,
    Hide,
    //Camp,
    Hunt,
    Chill,
    Patrol

}
```

There are also the NPC behavior states themselves, all located in the Script/NPC/States folder:

```
namespace Assets.Scripts.NPC.States
{
    public class IdleState : INPCState
    {
        private NPCController npc;
        private float idleTimer = 2f;

        public IdleState(NPCController npc)
        {
            this.npc = npc;
        }

        public void Enter()
        {
            npc.Agent.ResetPath();
            idleTimer = Random.Range(1.5f, 3f);

            npc.StartContextDialogue(DialogueContext.Idle);
        }

        public void Exit() { }

        public void Update()
        {
            if (npc.CanSeePlayer(out var player) && npc.isAggressive)
            {
                npc.target = player;
                npc.StateMachine.ChangeState(new ChaseState(npc));
                return;
            }

            idleTimer -= Time.deltaTime;
            if (idleTimer <= 0)
            {
                npc.StateMachine.ChangeState(new RoamState(npc));
            }
        }
    }
}
```

```
namespace Assets.Scripts.NPC.States
{
    public class SleepState : INPCState, IInterruptible
    {
        private NPCController npc;

        public SleepState(NPCController npc)
        {
            this.npc = npc;
        }

        public void Enter()
        {
            npc.Agent.ResetPath();
            npc.Animator.SetBool("IsSleeping", true);

            npc.StartContextDialogue(DialogueContext.Sleep);
        }

        public void Update()
        {
            // NPC спит, ничего не делает. Может проснуться по времени или если его потревожили
            if (npc.CanSeePlayer(out var player) && npc.isAggressive)
            {
                npc.target = player;
                npc.StateMachine.ChangeState(new ChaseState(npc));
            }
        }

        public void Exit()
        {
            npc.Animator.SetBool("IsSleeping", false);
        }


        public void Interrupt(NPCController source, InterruptReason reason)
        {
            if (reason == InterruptReason.PlayerRunning || reason == InterruptReason.PlayerWalking)
            {
                npc.StateMachine.ChangeState(new IdleState(npc));
                UIFloatingText.Create(npc.transform.position + Vector3.up, "Who are you?");
            }
        }
    }
```

All states have a common rule: they must inherit from INPCState.

```
public class SleepState : INPCState, IInterruptible
{
    private NPCController npc;
```

This inheritance is responsible for what the NPC does in a given state (walking, jumping, flying, standing, etc.). There are also two other interfaces: IInterruptible and IInteractableState.

```
namespace Assets.Scripts.NPC.States
{
    public class GuardState : INPCState, IInterruptible, IInteractableState
    {
        private NPCController npc;
        private Vector3 guardPoint;
```

These interfaces are responsible for how NPCs react to other NPCs, where IInterruptible is responsible for how the NPC reacts to the sound system associated with Emmiter.

And IInteractableState is responsible for how The NPC will react to player-NPC interactions (initiate dialogue, give out an item, etc.).

The "sound system" works as follows: for any action (not just the player's), the emitter.Activate(InterruptReason, Radius, Duration) string is called;

PlayerController.cs

Assembly-CSharp — PlayerController

```
184     }
185
186
187     void FixedUpdate()
188     {
189         rb.MovePosition(rb.position + movement.normalized * currentSpeed * Time.fixedDeltaTime);
190         if (movement.sqrMagnitude > 0.1f)
191         {
192             emitter.Activate(InterruptReason.PlayerWalking, currentSpeed / 2);
193         }
194     }
195
196     void Flip()
197     {
```

```
public class ScaredState : INPCState
{
    private NPCController npc;
    private Transform threat;
    private float fleeDistance = 10f;

    public ScaredState(NPCController npc, Transform threat)
    {
        this.npc = npc;
        this.threat = threat;
    }

    public void Enter()
    {
        Vector2 fleeDirection = (npc.transform.position - threat.position).normalized;
        Vector2 fleeTarget = npc.transform.position + (Vector3)(fleeDirection * fleeDistance);
        npc.Agent.SetDestination(fleeTarget);

        npc.emitter.Activate(InterruptReason.ScreamHelp, 4f, 1f);
    }

    public void Update()
    {
        // Можно добавить проверку "достаточно ли далеко"
    }

    public void Exit() { }
}
```

```
public class ChaseIntruderState : INPCState
{
    private readonly NPCController npc;
    private Transform target;

    public ChaseIntruderState(NPCController npc)
    {
        this.npc = npc;
        this.target = GameObject.FindGameObjectWithTag("Player").transform;
    }

    public void Enter()
    {
        npc.target = target;
    }

    public void Exit()
    {
        npc.Agent.ResetPath();
    }

    public void Update()
    {
        if (npc.target == null)
        {
            npc.StateMachine.ChangeState(new IdleState(npc));
            return;
        }

        npc.Agent.SetDestination(npc.target.position);
        npc.emitter.Activate(InterruptReason.ChaseAlert);

        float distance = Vector3.Distance(npc.transform.position, npc.target.position);
        if (distance <= 1.5f)
        {
            Debug.Log("Поймал игрока!");
            npc.Agent.ResetPath();
        }

        if (!npc.CanSeePlayer(out var seenPlayer))
        {
            npc.StateMachine.ChangeState(new IdleState(npc));
        }
    }
```

which triggers the sound source and "plays the sound." For greater accuracy, the system requires a "sound reason," also known as InterruptReason, which determines why the sound was played. You can also pass the "sound's" radius and its duration; by default, the radius is 1 game unit, and the duration is 0.1 seconds.

```
[System.Flags]
public enum InterruptReason
{
    None = 0,
    PlayerWalking = 1 << 0,
    PlayerRunning = 1 << 1,
    Lockpicking = 1 << 2,
    ScreamHelp = 1 << 3,
    ChaseAlert = 1 << 4,
    Hunting = 1 << 5,
    CombatJoin = 1 << 6,
    //add more reasons as needed
}
public enum FactionType
```

You can request the GPT Chat by passing it the above information to create new states.