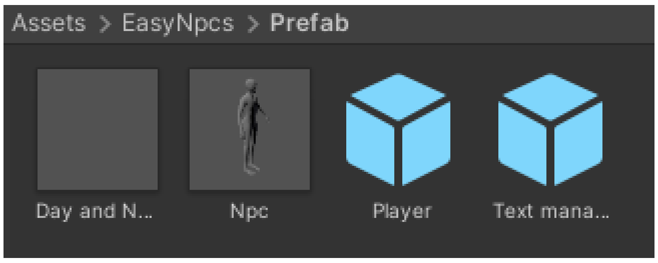
Easy NPC documents

An example project is already set up in EasyNPCs/Scenes/ForStore
This document will provide you with information on using 'Easy Npcs' for your project. The main feature of the asset is the 'Npc' prefab stored inside EasyNpcs/Prefab/Npc

1. Setting up Easy npcs

1.1 Setting up

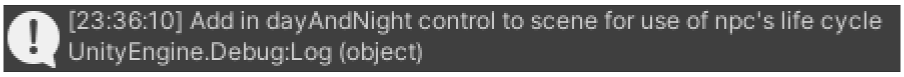
Before we get started we need a plane for the npcs to walk on and a navmesh configured on the plane so that the npcs can walk around.
After that go to EasyNpcs/Prefab and find the 'Npc' prefab.



Click and drag the 'Npc' prefab into the scene.

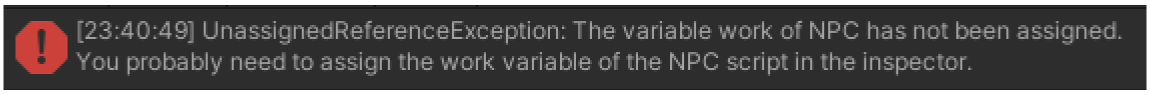
1.2 Creating day and night cycle of npcs

If you play the game you will get a debug.Log inside the console indicating that you need a day and night controller.



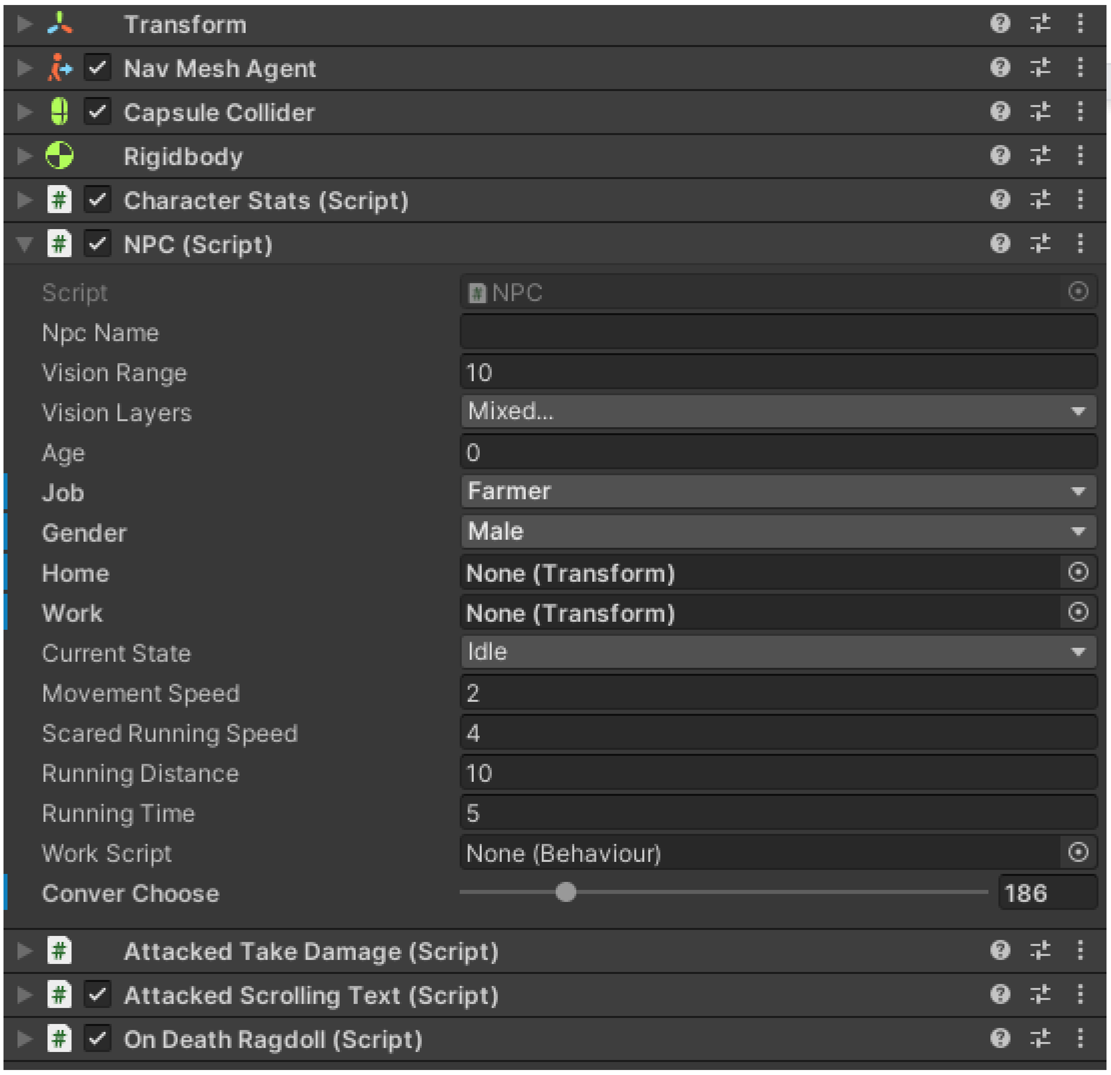
```
[23:36:10] Add in dayAndNight control to scene for use of npc's life cycle
UnityEngine.Debug:Log (object)
```

Inside EasyNpcs/Prefab and next to the 'Npc' prefab, you'll see the 'Day and Night Controller' prefab. Click and drag the prefab into the scene and play it again. This time you will get this error.



```
[23:40:49] UnassignedReferenceException: The variable work of NPC has not been assigned.
You probably need to assign the work variable of the NPC script in the inspector.
```

This means that the work position where the npc goes for the day is not assigned. Create a new game object named 'work' and place it on the plane. Click the npc we put in and go to the inspector.

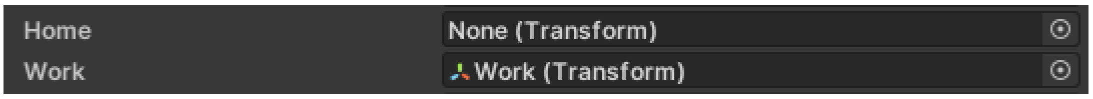


Transform
Nav Mesh Agent ✓
Capsule Collider ✓
Rigidbody
Character Stats (Script) ✓
NPC (Script) ✓

Script	NPC
Npc Name	
Vision Range	10
Vision Layers	Mixed...
Age	0
Job	Farmer
Gender	Male
Home	None (Transform)
Work	None (Transform)
Current State	Idle
Movement Speed	2
Scared Running Speed	4
Running Distance	10
Running Time	5
Work Script	None (Behaviour)
Conver Choose	186

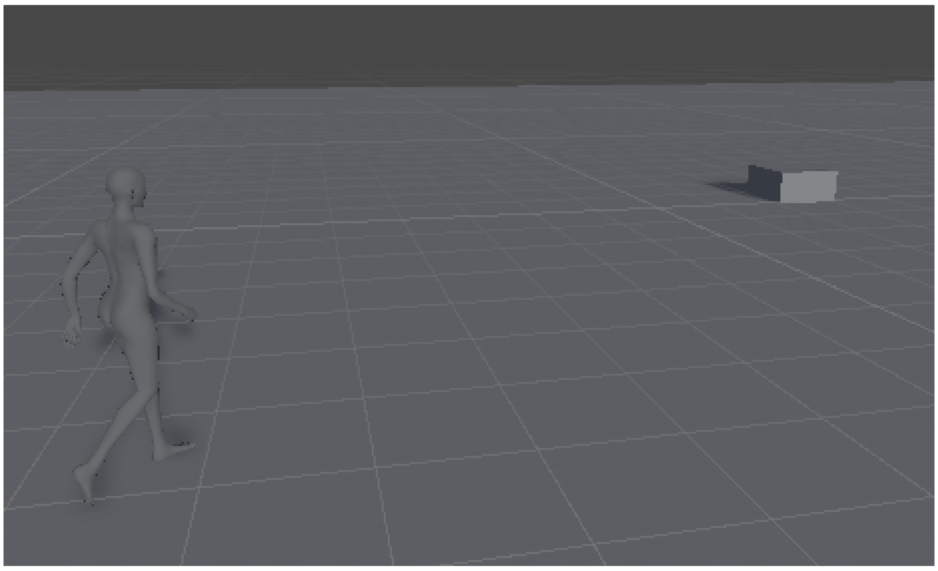
Attacked Take Damage (Script)
Attacked Scrolling Text (Script) ✓
On Death Ragdoll (Script) ✓

Find the ‘NPC’ component like in the picture. Find the variable ‘Work’ and assign the ‘work’ game object’s transform to it.

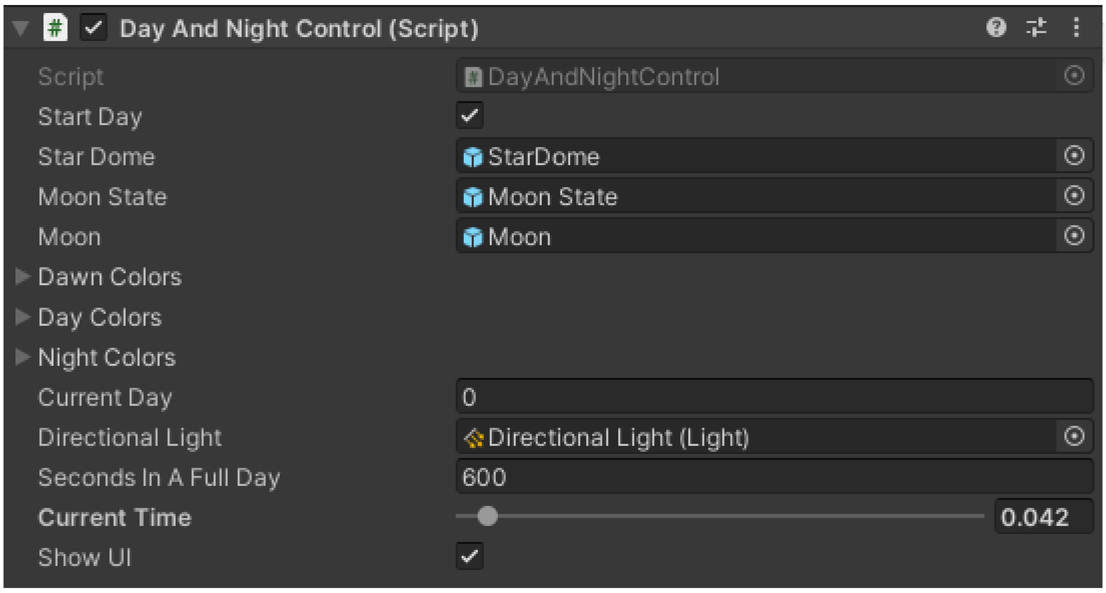


Home	None (Transform)
Work	Work (Transform)

Now play the game again and the npc will go to the designated spot.



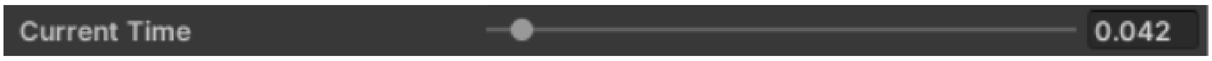
You will see that the npc heads over to the position of the ‘work’ game object.
Since the npc goes to work for the day, now we want it to go home for the night. Click on the day and night controller inside the scene and head to the inspector. You will find the ‘Day and Night Control’ component attached.



Day And Night Control (Script)

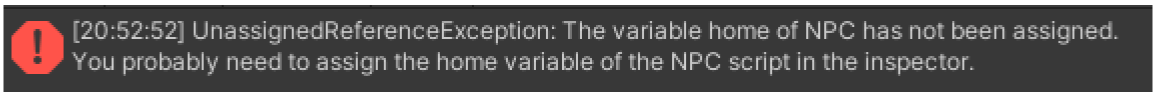
Script	DayAndNightControl
Start Day	✓
Star Dome	StarDome
Moon State	Moon State
Moon	Moon
Dawn Colors	
Day Colors	
Night Colors	
Current Day	0
Directional Light	Directional Light (Light)
Seconds In A Full Day	600
Current Time	0.042
Show UI	✓

Notice the ‘Current Time’ variable.



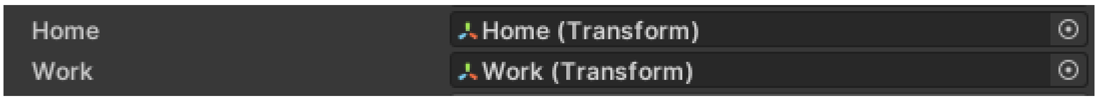
Current Time 0.042

This variable indicates what time of the day it is. 0.3 ~ 0.8 is considered as morning and daytime. 0.8 ~ 1, 0 ~ 0.3 is considered as night time. When the variable reaches 1 it will go back to 0.
Change the time of day to night and check the console. You will get this error.



```
[20:52:52] UnassignedReferenceException: The variable home of NPC has not been assigned.
You probably need to assign the home variable of the NPC script in the inspector.
```

This time create a game object named ‘home’ and place it on the plane away from the ‘work’ game object. And assign the transform of the ‘home’ game object to the ‘Home’ variable inside the ‘NPC’ component.

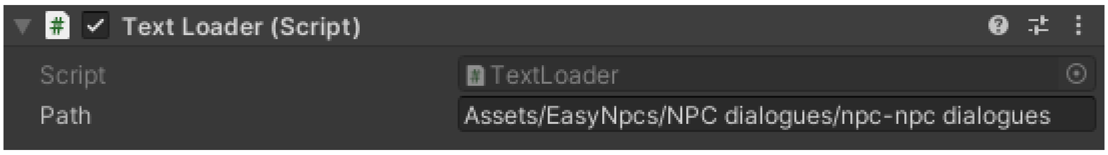

Home Home (Transform)
Work Work (Transform)

Now once you hit play and set the time to night, the npc will move the location of the ‘home’ game object.

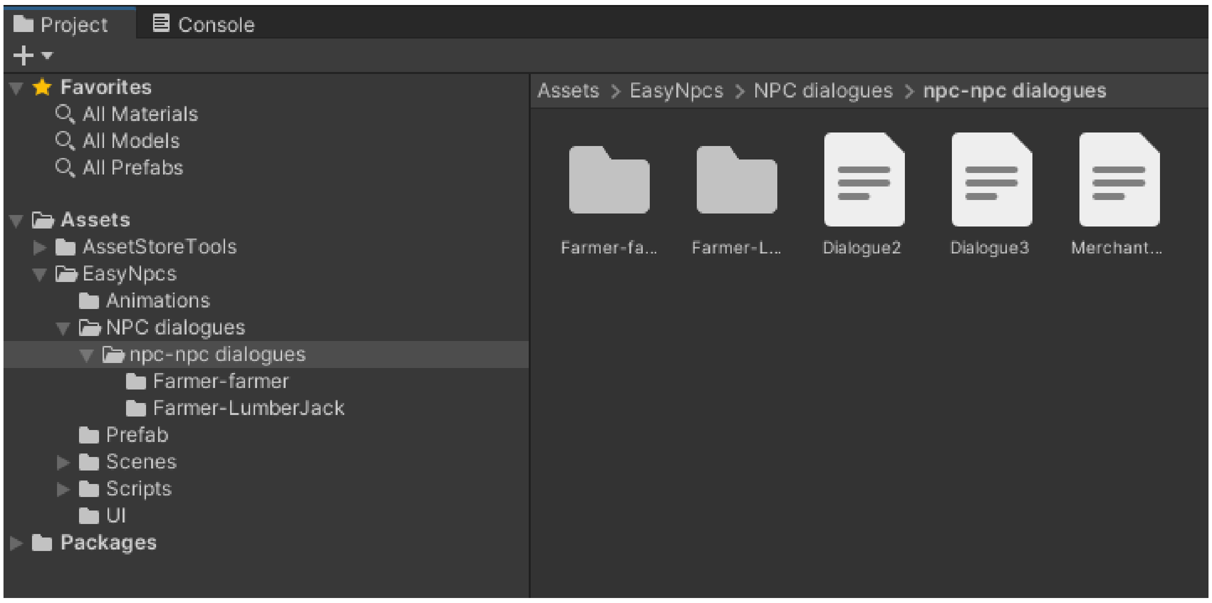
2. Creating conversations

2.1 Adding Text manager

Before we can have the npc talk we need a text manager that loads the conversation files. You can find the ‘text manager’ prefab in EasyNpcs/Prefab. Click and drag the prefab into the scene. If you see the text manager in the inspector you will see the ‘text loader’ component. With a string variable named path

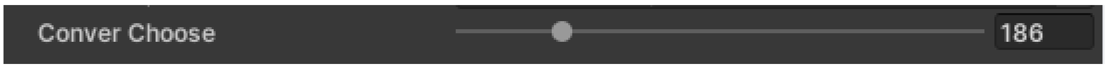

Text Loader (Script)
Script TextLoader
Path Assets/EasyNpcs/NPC dialogues/npc-npc dialogues

Path is the path to where all the conversation files are stored. By default, it is EasyNpcs/NPC dialogues/npc-npc dialogues. If you head over to the files you’ll see a list of conversations in ‘txt’ format.

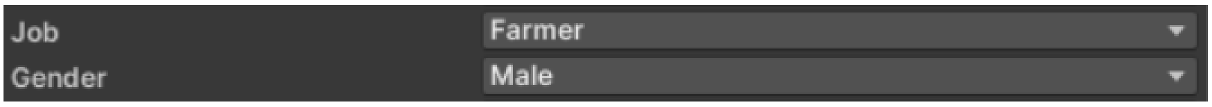


2.2 Trying out the conversations

Let’s try out the conversations. Go to the ‘NPC’ component of the npc we previously added in the scene to try out the day and night cycle. And from there find the ‘Conver Choose’ variable.

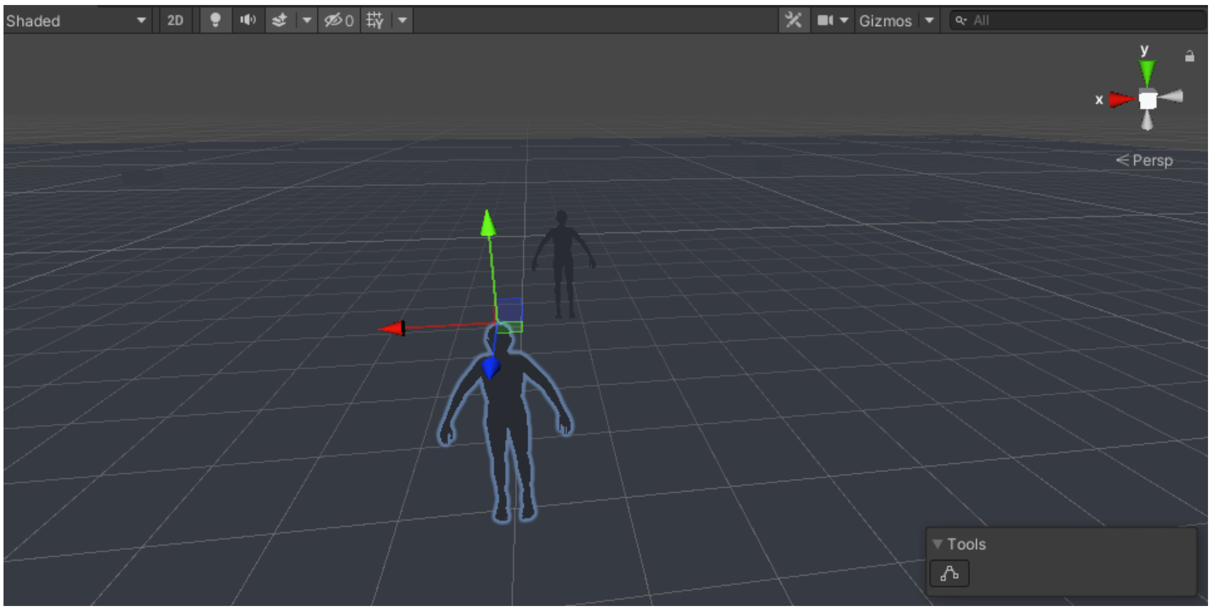

Conver Choose 186

The ‘Conver Choose’ variable chooses how likely the npc will have an interaction with another npc near it. It is calculated per frame. Set the variable to a number greater than 0. Then find the ‘Job’ and ‘Gender’ variables inside the ‘NPC’ component. Set each of the variables to ‘Farmer’ and ‘Male’.

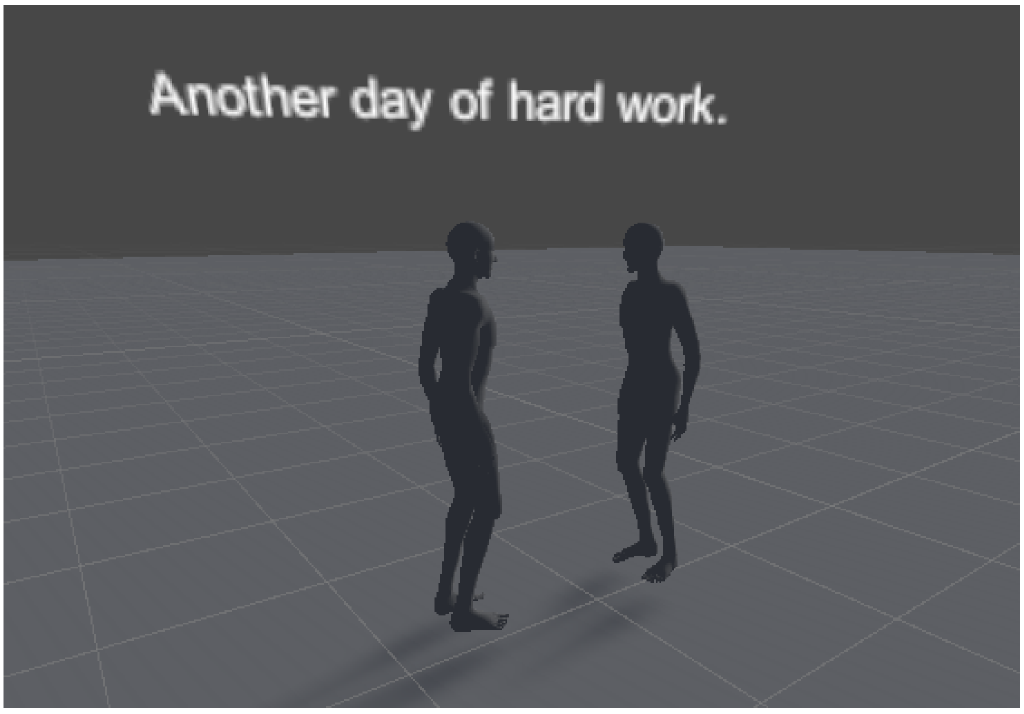


Job	Farmer
Gender	Male

Add another npc to the scene. And just as before, set the ‘Conver Choose’ variable to a number bigger than 0. And set the ‘Job’ and ‘Gender’ variables to ‘Farmer’ and ‘Male’.

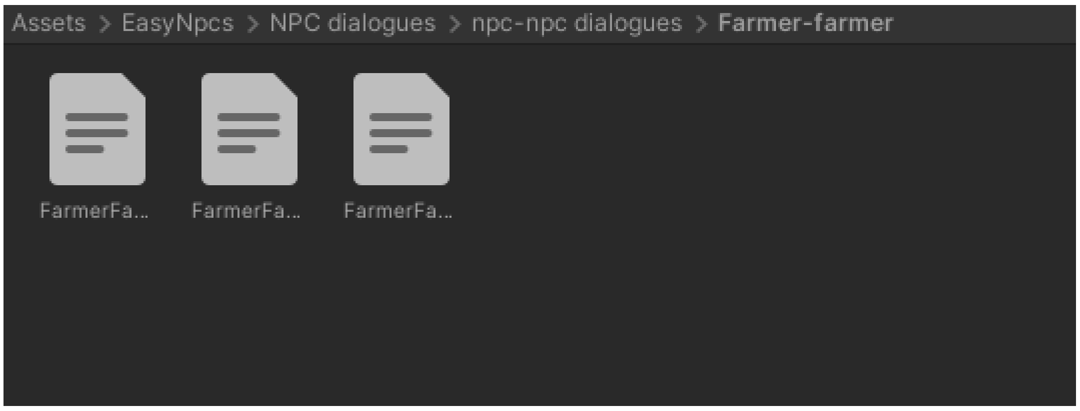


Now run the game and the npcs will have conversations.

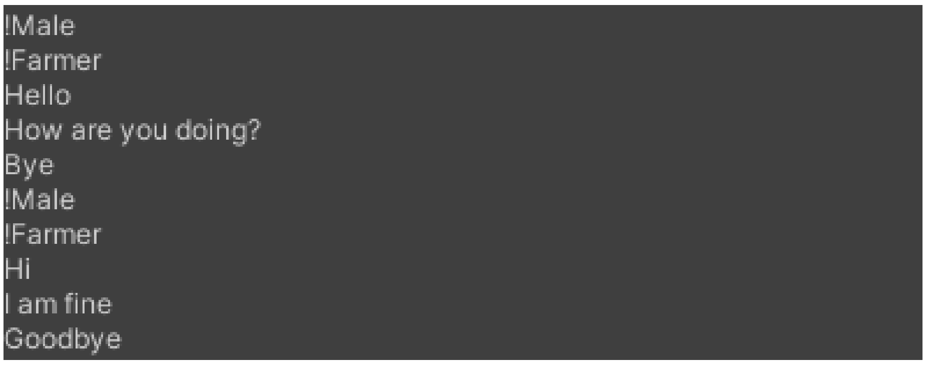


Another day of hard work.

Go over to EasyNpcs/NPC dialogues/npc-npc dialogues/Farmer-farmer. The ‘txt’ files inside here are only for conversations between 2 male farmers. If you keep on running the conversations between the 2 npcs inside the scene you will notice that only the files inside ‘Farmer-farmer’ are chosen and executed.


Assets > EasyNpcs > NPC dialogues > npc-npc dialogues > Farmer-farmer
FarmerFa...
FarmerFa...
FarmerFa...

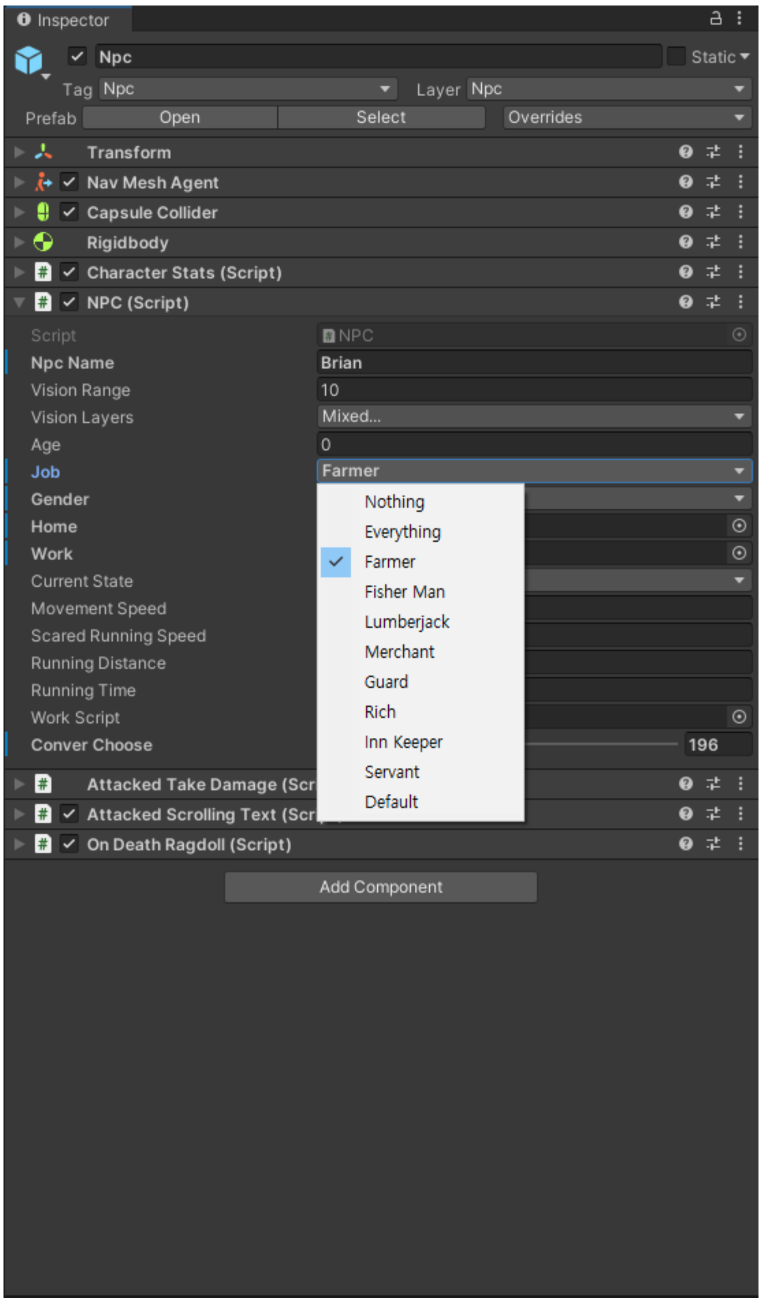
Open the first txt file from the left.



```
!Male
!Farmer
Hello
How are you doing?
Bye
!Male
!Farmer
Hi
I am fine
Goodbye
```

‘!Male’ and ‘!Farmer’ are indicators that this section is to be said by an npc whose gender is a male and his job is being a farmer. The section ends before the indicator for the 2nd npc is written. So in this case, the section ends with the sentence ‘Bye’. The second indicator and section are for the 2nd npc. So when this conversation is executed it will go by this:
Farmer1: Hello
Farmer2: Hi
Farmer1: How are you doing?
Farmer2: I am fine
Farmer1: Bye
Farmer2: Goodbye

2.3 Creating conversations
Let’s write our own conversations. First, we need to see what types of jobs and genders are there. Go to the npc component by the inspector and click on the Job and Gender variables.



Npc

Tag Npc Layer Npc

Prefab Open Select Overrides

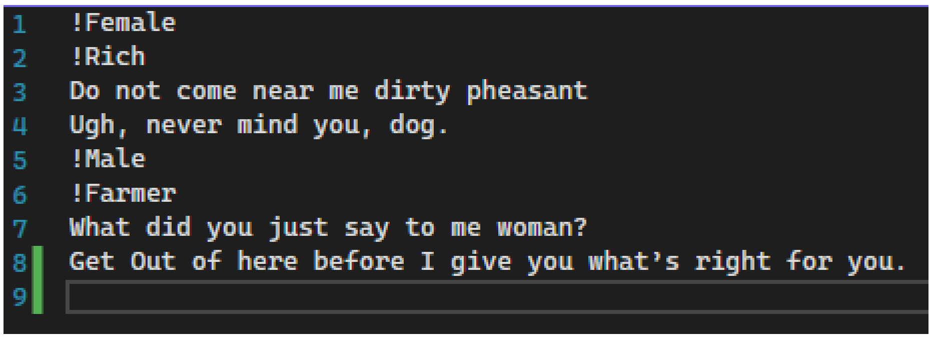
Script NPC
Npc Name Brian
Vision Range 10
Vision Layers Mixed...
Age 0
Job Farmer
Gender
Home
Work
Current State
Movement Speed
Scared Running Speed
Running Distance
Running Time
Work Script
Conver Choose 196

You’ll be able to see all the available jobs and genders.
Let’s write a conversation between a rich female npc and a male farmer npc. The conversation will go as the following
RichFemale: Do not come near me dirty pheasant.
MaleFarmer: What did you just say to me woman?
RichFemale: Ugh, nevermind you, dog.
MaleFarmer: Get out of here before I give you what’s right for you.

Make a ‘txt’ file and write like the following
!Female
!Rich
Do not come near me dirty pheasant
Ugh, never mind you, dog.

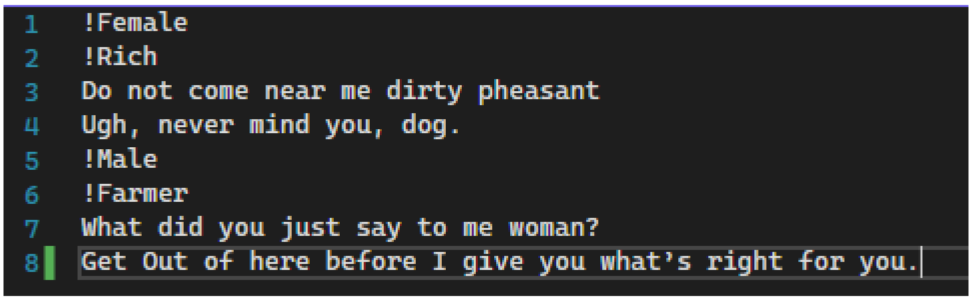
!Male
!Farmer
What did you just say to me woman?
Get Out of here before I give you what’s right for you.

Save it and put it inside EasyNpcs>NPC dialogues>npc-npc dialogues.
Warning! Do not leave an extra space of a line at the end. This is an example of a wrong ‘txt’ file.



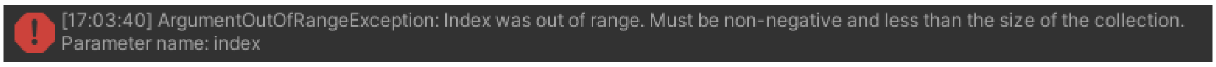
```
!Female
!Rich
Do not come near me dirty pheasant
Ugh, never mind you, dog.
!Male
!Farmer
What did you just say to me woman?
Get Out of here before I give you what’s right for you.

```

The asset will also take the blank space as a string which will result in an error. This is the right way to write the ‘txt’ file.



```
!Female
!Rich
Do not come near me dirty pheasant
Ugh, never mind you, dog.
!Male
!Farmer
What did you just say to me woman?
Get Out of here before I give you what’s right for you.
```

Change one of the npc’s jobs and gender to rich and female. And you will be able to see the npcs speaking the conversation we’ve written above.
But instead of having a successful conversation, the npcs might not have one and you might get this error.



```
[17:03:40] ArgumentOutOfRangeException: Index was out of range. Must be non-negative and less than the size of the collection.
Parameter name: index
```

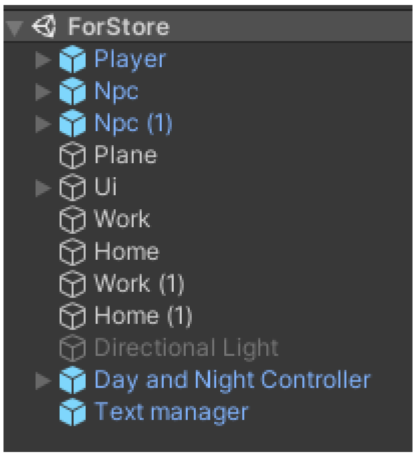
This is because each npc has its own specific Id. Depending on the Id, npcs can start a conversation with another npc which lets it start the conversation first. Sometimes because of the Id, an npc is not allowed to start a conversation with a specific npc. And is only allowed to receive a conversation. If you change the jobs and genders of the 2 npc to the opposite the npcs will be able to have a conversation.

3. Damaging npcs

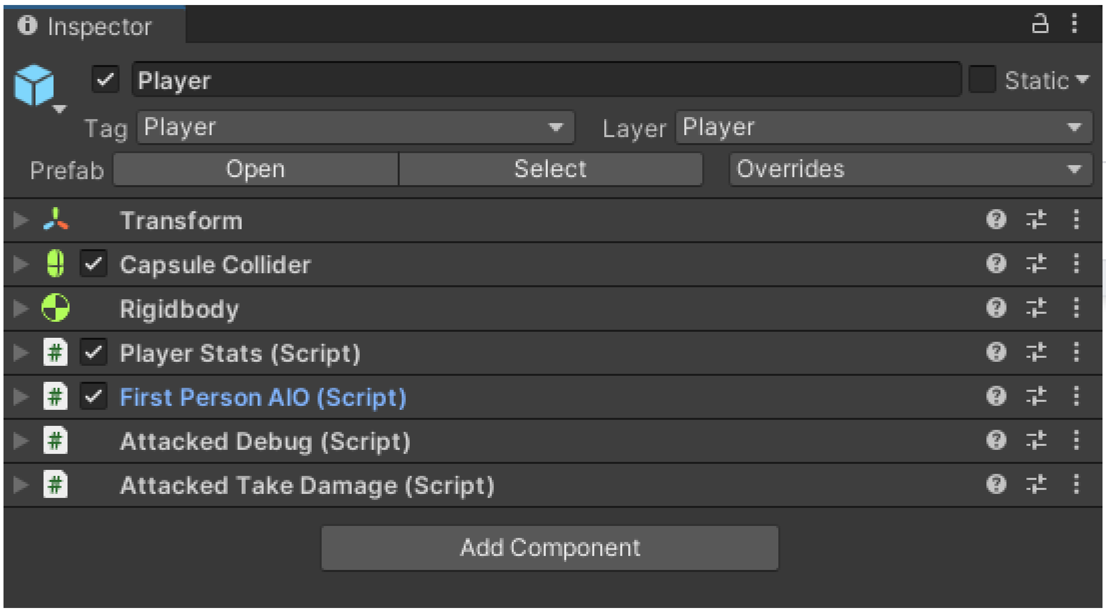
3.1 How to give damage to npcs

We will go through how the attacks are executed in Easy NPCs and how you can add them to your character.

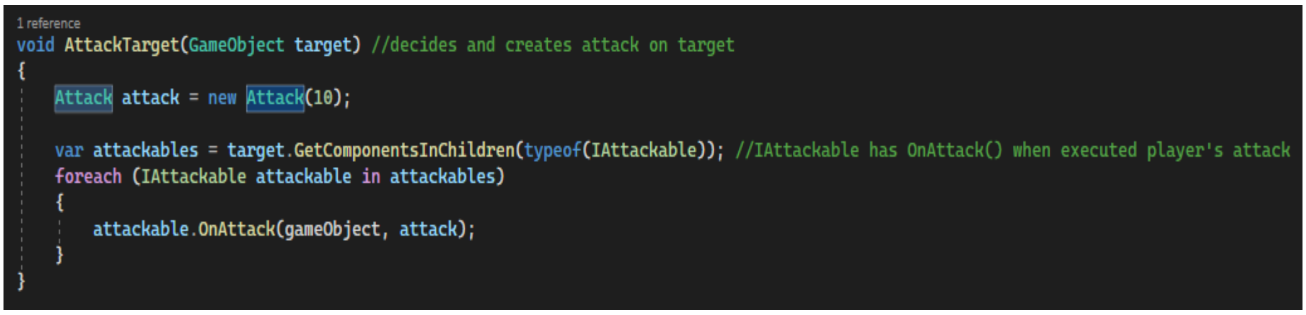
If you head over to the sample scene(named 'For Store') we've already made you will be able to find the 'Player' game object.



Click the 'Player' object and check it inside the inspector. Find the 'First Person AIO' component.



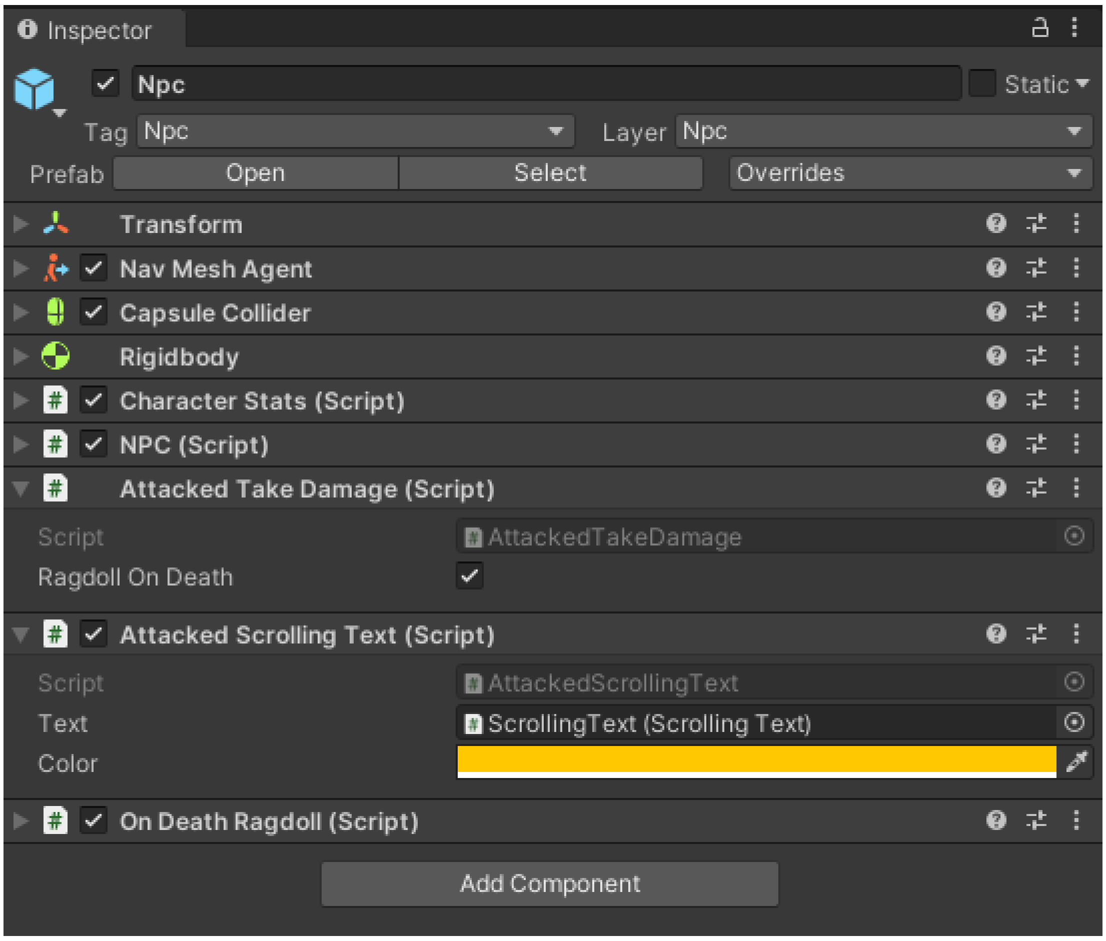
Open the 'First Person AIO' component in a scripting API. Find the method AttackTarget(GameObject target, bool bashAttack = false)



```
1 reference
void AttackTarget(GameObject target) //decides and creates attack on target
{
    Attack attack = new Attack(10);

    var attackables = target.GetComponentsInChildren(typeof(IAttackable)); //IAttackable has OnAttack() when executed player's attack
    foreach (IAttackable attackable in attackables)
    {
        attackable.OnAttack(gameObject, attack);
    }
}
```

This method is executed when the player presses mouse1 to attack.
'GameObject target' is the npc that the player is attacking.

Then we create an ‘attack’. ‘Attack’ is a class that can be returned as an int when dealing damage. Right now it’s written ‘Attack attack = new Attack(10)’. This means we’ve created an ‘attack’ that will give 10 damage to the npc when it is used.
The next line is written ‘var attackables = target.GetComponenetsInChildren(typeof(IAttackable)); Which means get all the components inside the target npc that is inherited from interface’IAttackable’. The npc has 2 components that inherit from ‘IAttackable’. Those are the ‘Attacked Take Damage’ and ‘Attacked Scrolling Text’ components. ‘Attacked Take Damage’ is the component that handles the damage while ‘Attacked Scrolling Text’ pops up a text that the npc has been attacked.

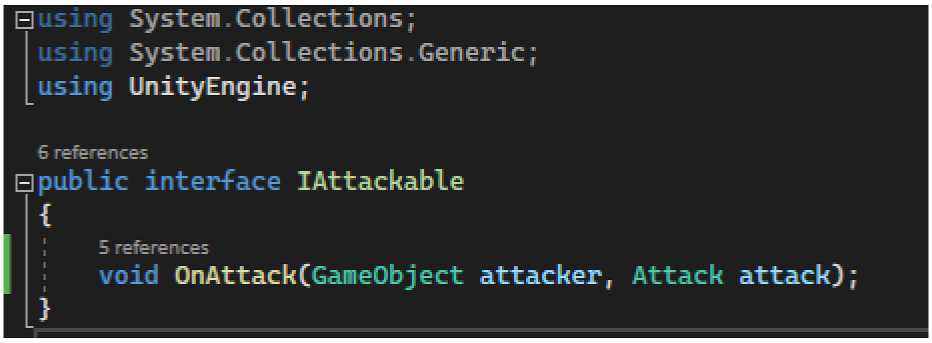

Inspector

Npc — Static
Tag: Npc — Layer: Npc
Prefab: Open | Select | Overrides

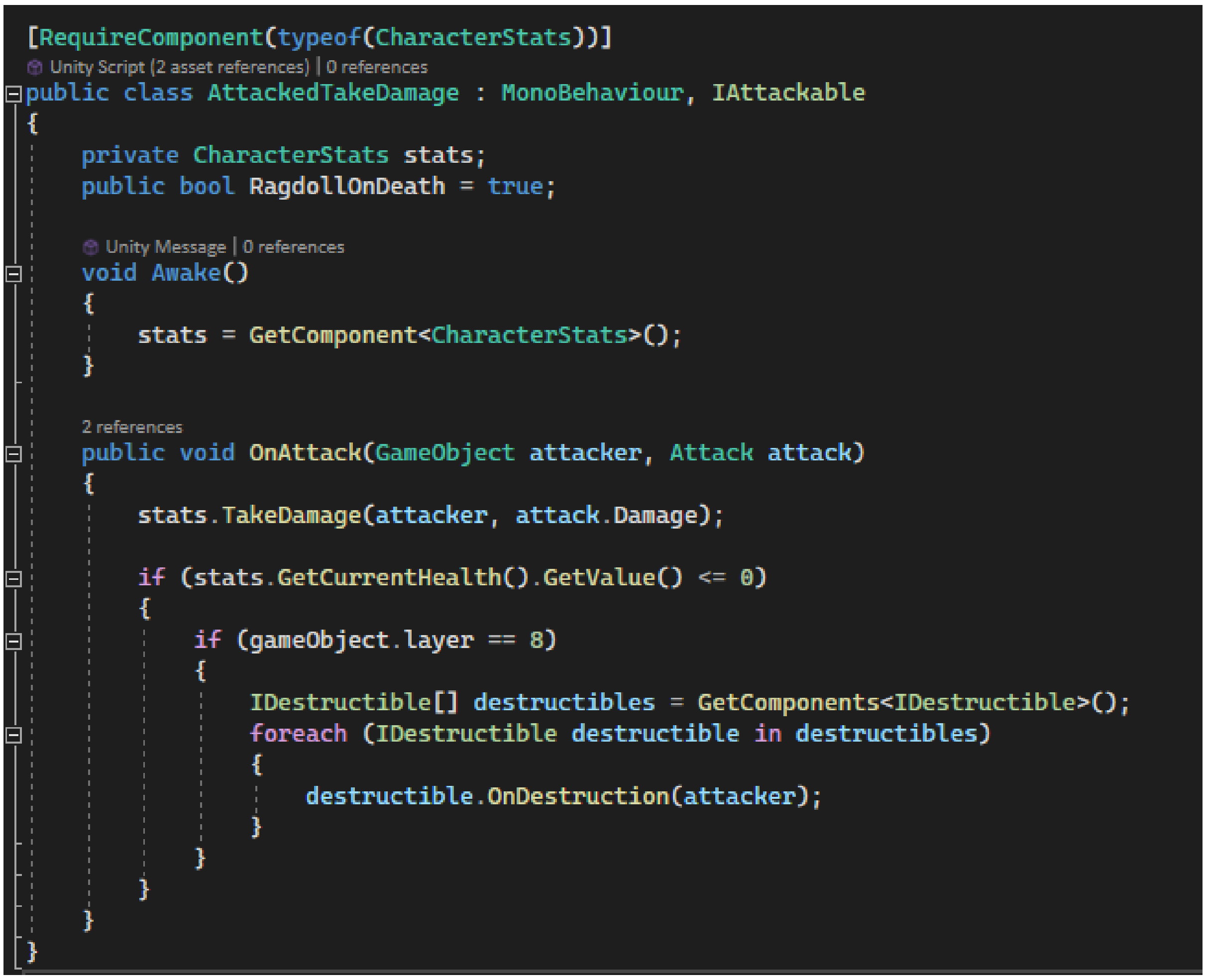
- Transform
- ✓ Nav Mesh Agent
- ✓ Capsule Collider
- Rigidbody
- ✓ Character Stats (Script)
- ✓ NPC (Script)
- Attacked Take Damage (Script)
  - Script: AttackedTakeDamage
  - Ragdoll On Death: ✓
- ✓ Attacked Scrolling Text (Script)
  - Script: AttackedScrollingText
  - Text: ScrollingText (Scrolling Text)
  - Color: 🟨
- ✓ On Death Ragdoll (Script)

Add Component

The interface ‘IAttackable’ has a method named OnAttack().

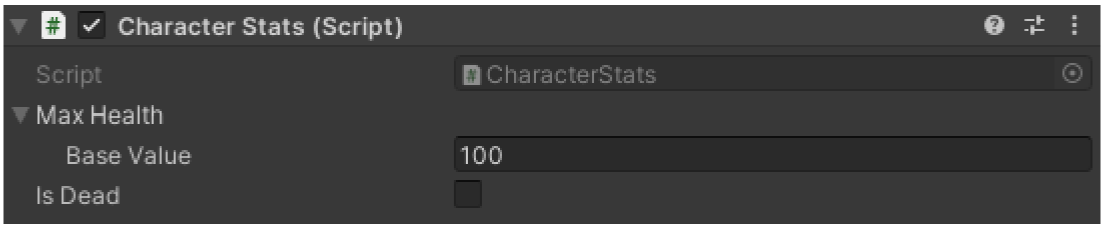

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

6 references
public interface IAttackable
{
    5 references
    void OnAttack(GameObject attacker, Attack attack);
}
```

The components ‘Attacked Take Damge’ and ‘Attacked Scrolling Text’ override the OnAttack() method for each of their purposes. Let’s take a look at how ‘Attacked Take Damage’ works.



```
[RequireComponent(typeof(CharacterStats))]
Unity Script (2 asset references) | 0 references
public class AttackedTakeDamage : MonoBehaviour, IAttackable
{
    private CharacterStats stats;
    public bool RagdollOnDeath = true;

    Unity Message | 0 references
    void Awake()
    {
        stats = GetComponent<CharacterStats>();
    }

    2 references
    public void OnAttack(GameObject attacker, Attack attack)
    {
        stats.TakeDamage(attacker, attack.Damage);

        if (stats.GetCurrentHealth().GetValue() <= 0)
        {
            if (gameObject.layer == 8)
            {
                IDestructible[] destructibles = GetComponents<IDestructible>();
                foreach (IDestructible destructible in destructibles)
                {
                    destructible.OnDestruction(attacker);
                }
            }
        }
    }
}
```

First, it finds the component ‘Character Stats’ attached to the npc itself at ‘stats’. ‘CharacterStats’ is the component that stores the current health rate of the ‘Npc’. It is also responsible to show whether the npc is alive or dead.



Character Stats (Script)	
Script	CharacterStats
Max Health	
Base Value	100
Is Dead	☐

Max Health>Base value: Health of the npc on start.
Is Dead: Changes to true if npc's health has reached below 0. Otherwise false.

The OnAttack() method calls another method named TakeDamage(Gameobject attacker, float damage) from the ‘Character Stats’ component which subtracts the health variable in the ‘Character Stats’ component. We can see in the method that we return the attack value to int by Attack.Damage.

The rest of the code is for enabling the ragdoll effect.

If you wish to intergrade Easy Npcs with your game to the full extent you will need to go through the source code. You ask for support on our official website:
https://1stmanleader.wixsite.com/website
If you wish to contact us directly: 1stmanleader@gmail.com
Also, Easy Npcs will be constantly updated. For we believe long time support is as important as creating something new.