

Quick Start Guide

Contents

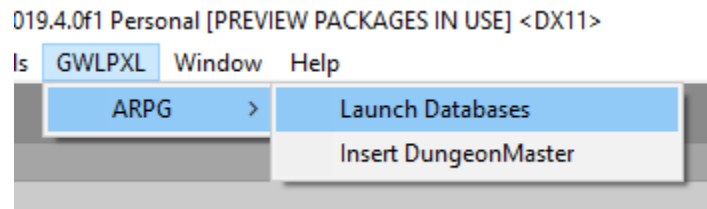
Quick Start Guide	1
Quick Creation of Database	2
Quick Creation of Actors	5
Quick Creation of Scene	10
Quick Unity NavMesh Setup	11
Quick Player Input Setup	13

This quick guide assumes you go in order:

- 1) Create new game databases,
- 2) Create a player actor,
- 3) Set up a Scene,
- 4) Set up player Inputs.

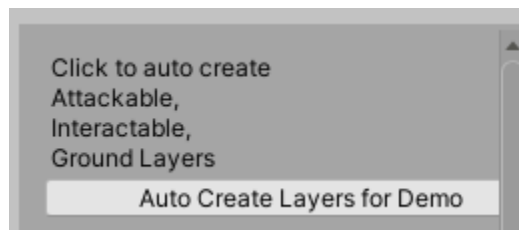
Quick Creation of Database

1. Launch the database launcher.



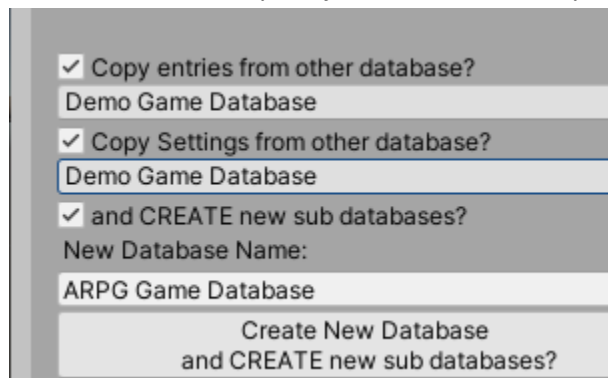
a.

2. Include necessary layers (if you don't already have them in the project). You can auto create or manually create.



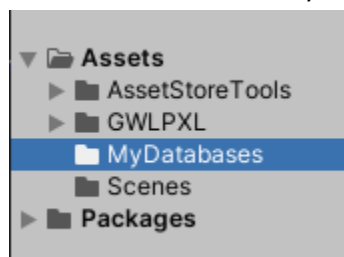
a.

3. Enable Copy Database and Copy Settings, select the Demo Database. (Disabling the toggle will still create the databases, they will just be left blank for you to create.)



a.

4. Click Create and Create a New Folder or Place it in your own game's unique folder. Select the new folder as the destination for your databases.

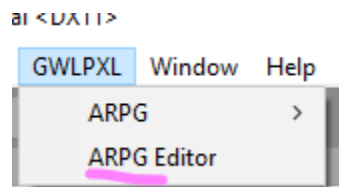


a.

- i. **DO NOT CHOOSE THE /Assets ROOT FOLDER OR THE GWLPXL FOLDER OR ANY OF ITS SUBFOLDERS. THIS WILL CAUSE CONFLICTS WITH UPDATES and databases.**

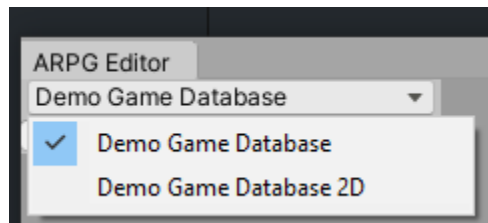
5. Allow the asset to create and copy over the necessary files.
6. Once complete, launch the new database editor.

Quick Start - Ultimate ARPG - Return to [Main Document](#)



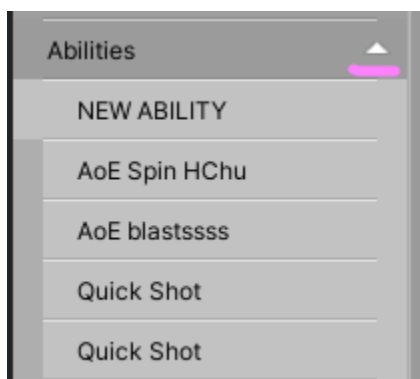
a.

7. Click on the newly created database (it will have the name you gave it from the previous step, which may differ from the name seen in the picture below).



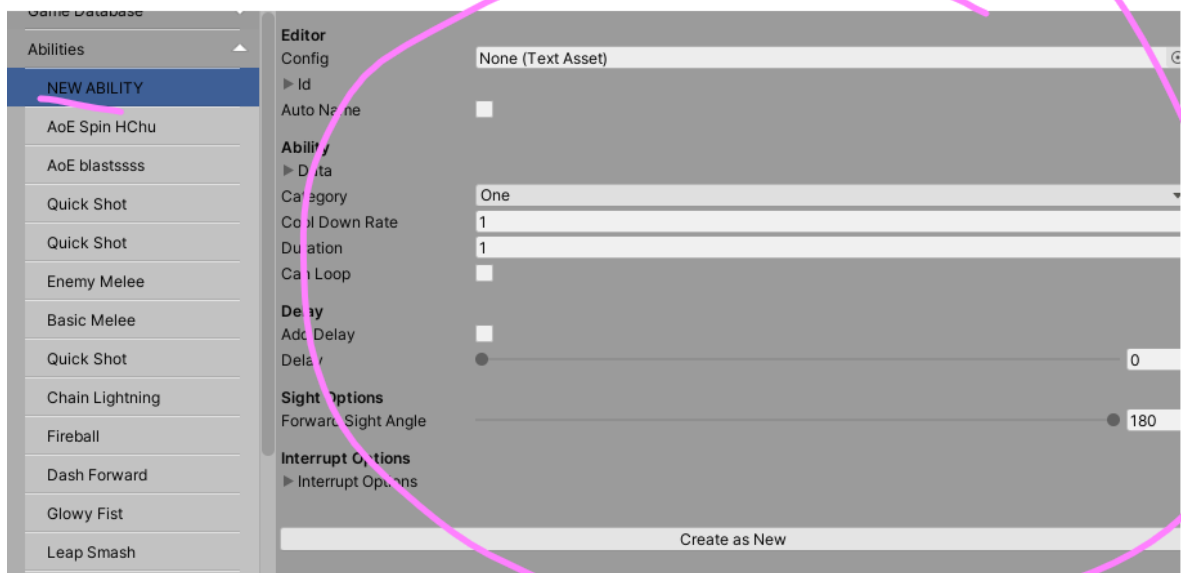
a.

8. Click the dropdown arrow on any of the selections to expand.



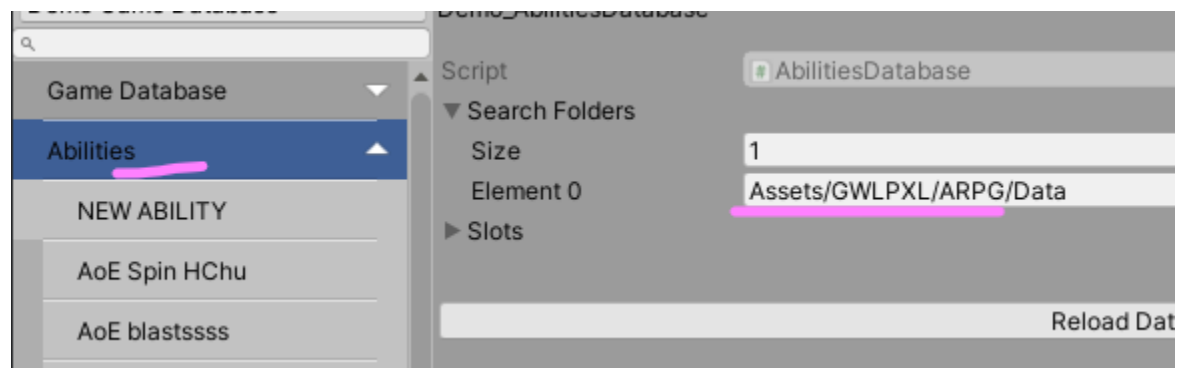
9.

10. Use the first entry to create any new elements.



11.

12. By creating elements through the game database, it will auto add them and assign unique IDs. This is preferred over duplication, which won't assign unique IDs until you reload the DB AND the duplicate item is found within your specified path



a.

13. Your path should differ if you created new databases.

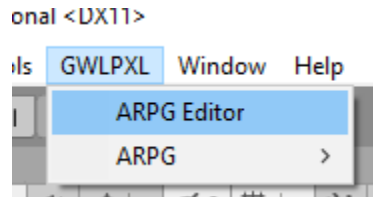
Quick Creation of Actors

Demo characters are found in

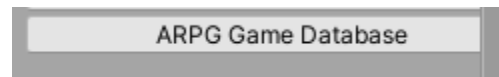
- Assets/GWLPXL/ARPG/Data/_Characters/_Player

You can drag in one of those prefabs, duplicate the one from the demo scene, or read on to learn how to create a new one.

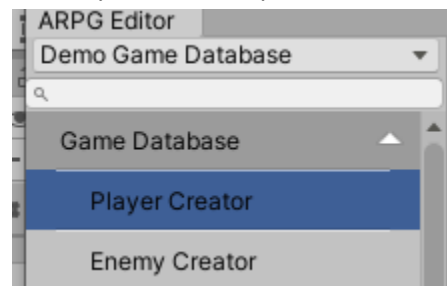
1. Open the Editor.



- a.
2. Select your database.



- a.
3. Choose the 'Player Creator' option under the "Game Database".



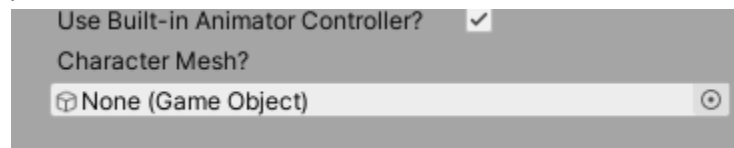
- a.
4. Assign the options (you can make custom ones later and these can be changed later, just assign some quick ones to see the process for now).

Quick Start - Ultimate ARPG - Return to [Main Document](#)



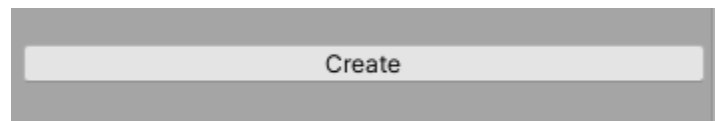
a.

5. Assign a mesh gameobject (the visualization of the character). This is the object the system will assume you want animated.



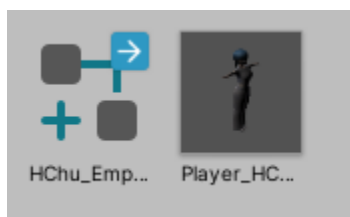
a.

6. Click Create and choose the location to save it.



a.

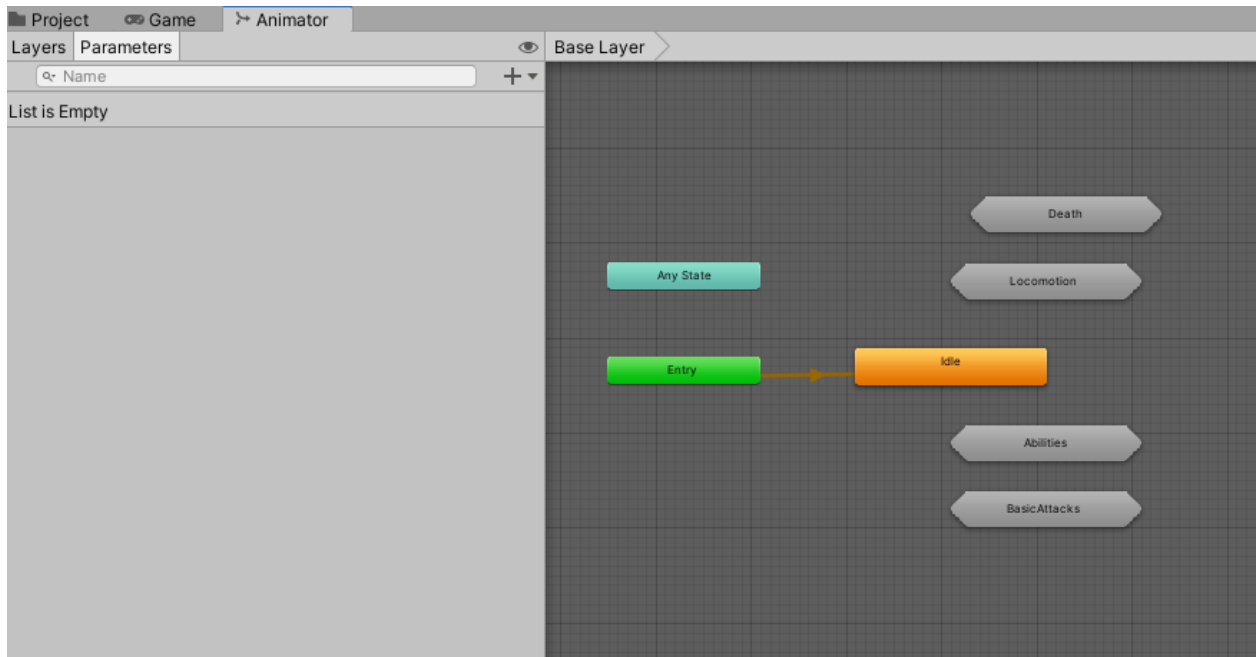
If you selected a **mesh** and are using the **ARPG Animation system**, the Player and Enemy actors will create two things: The Prefab and an Animator Controller. Example pictured below.



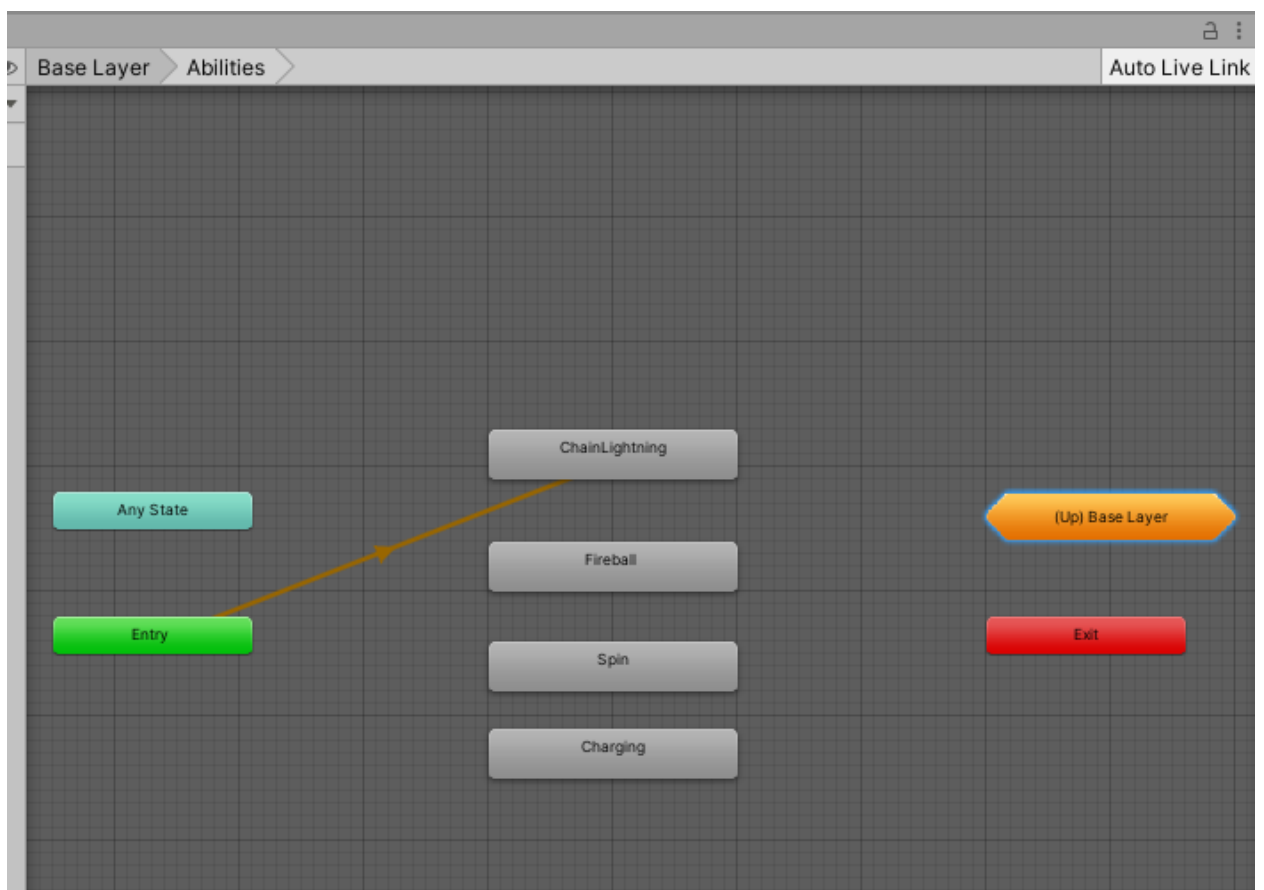
The prefab is the character you can drag into the scene, and the Animator Controller is where you'll set the animations if using the built-in animation.

An example the Hchu Controller

Quick Start - Ultimate ARPG - Return to [Main Document](#)

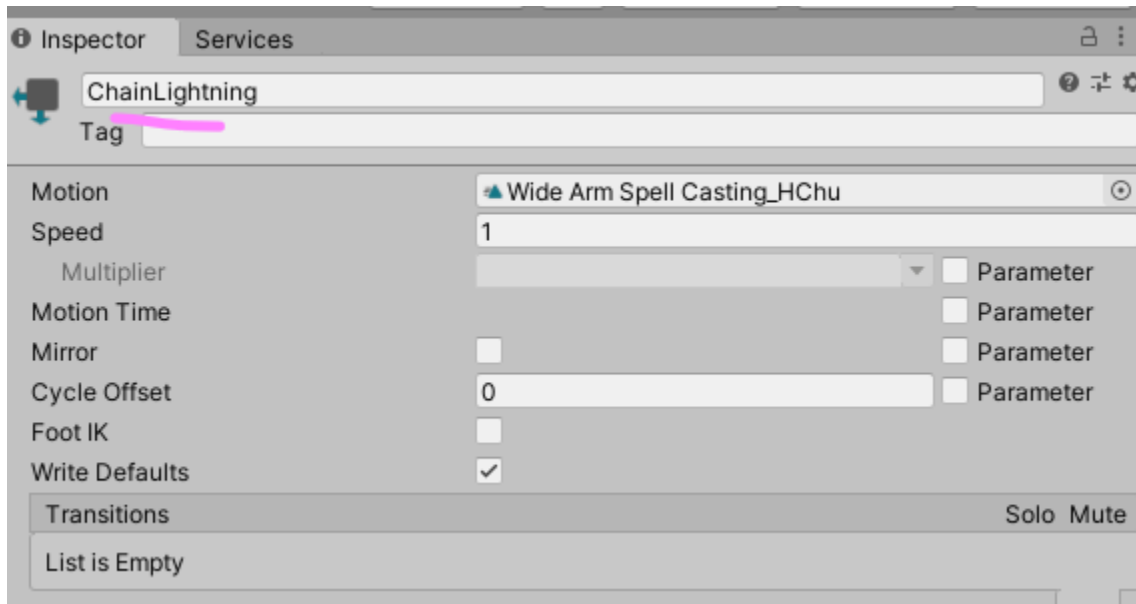


Those are sub-states that merely organize the states inside. Example of 'Abilities' sub-state:

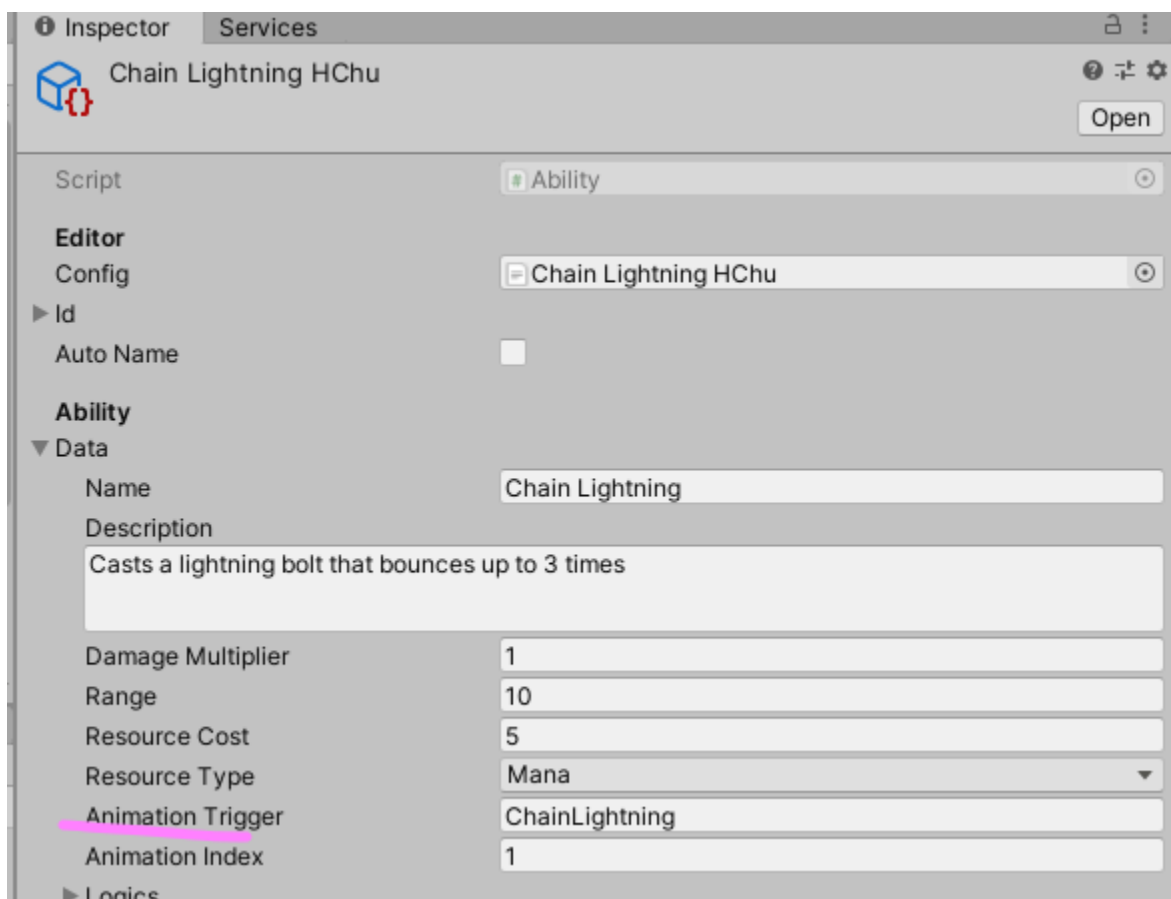


The default ability system calls the state directly

Quick Start - Ultimate ARPG - Return to [Main Document](#)



See example:

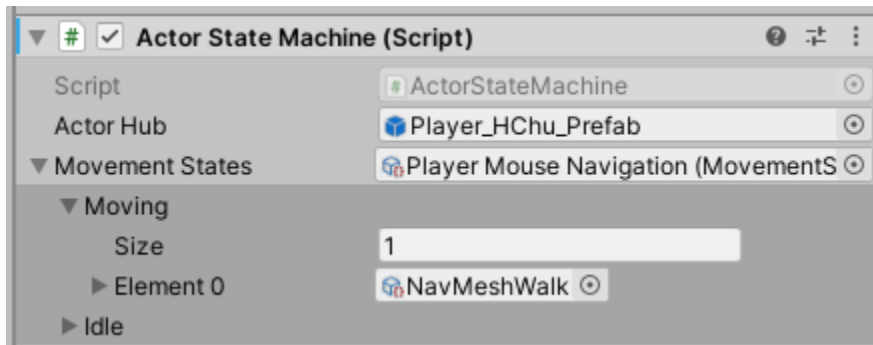


You can change the animation behavior by overriding `TriggerAbilityAnimation` in the `PlayerAbilityUser` script found on the `AbilitySystem`.

Quick Start - Ultimate ARPG - Return to [Main Document](#)

```
1 reference
protected virtual void TriggerAbilityAnimation(Ability toCast)
{
    if (actorhub.MyAnimator != null && triggerAnimations == true)
    {
        actorhub.MyAnimator.Play(toCast.Data.AnimationTrigger, 0, 0);
    }
}
2 reference
```

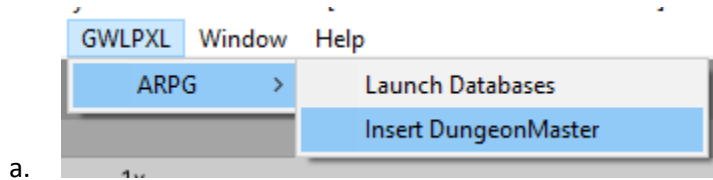
The generic animation locomotion states (e.g. walk, idle) are controlled by the Actor State Machine found on the player root.



Quick Start - Ultimate ARPG - Return to [Main Document](#)

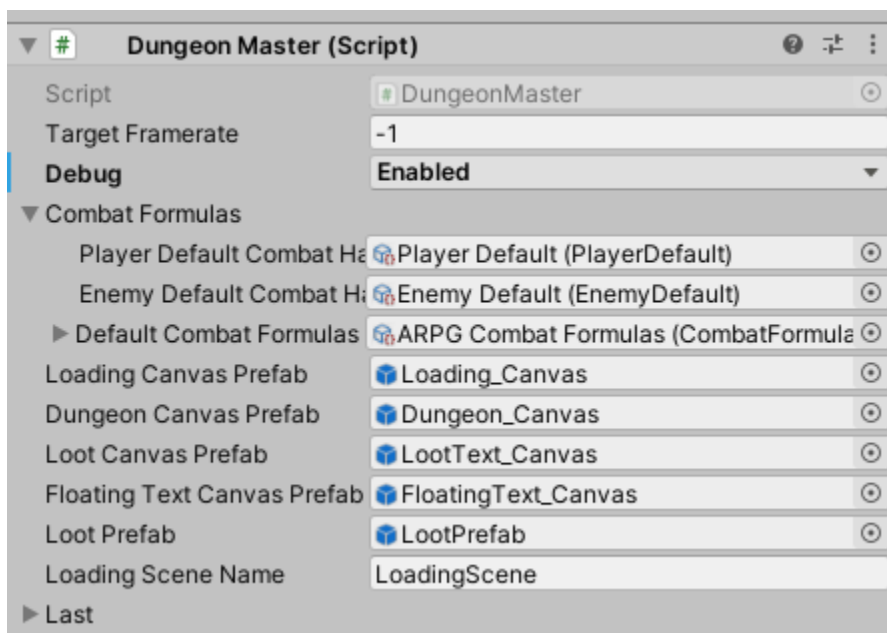
Quick Creation of Scene

Insert the DungeonMaster singleton.



The DungeonMaster singleton controls the creation of the FloatingText, LootText, and Dungeon Canvas elements. It's also what carries over the player data between scenes.

If you plan to use Unity's NavMesh and the built-in movement, proceed to the next steps. If not, your scene setup is done.

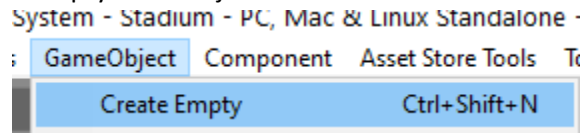


- Target Framerate will set the desired framerate at Start. Leave -1 for uncapped.
- Debug will enable the ARPG Debugger, which will provide messages in the log.
- Combat formulas control the damage results for interactions. These can be overridden and replaced.
- LootPrefab is the default Loot Prefab if one is not set on a LootDrop.

Quick Unity NavMesh Setup

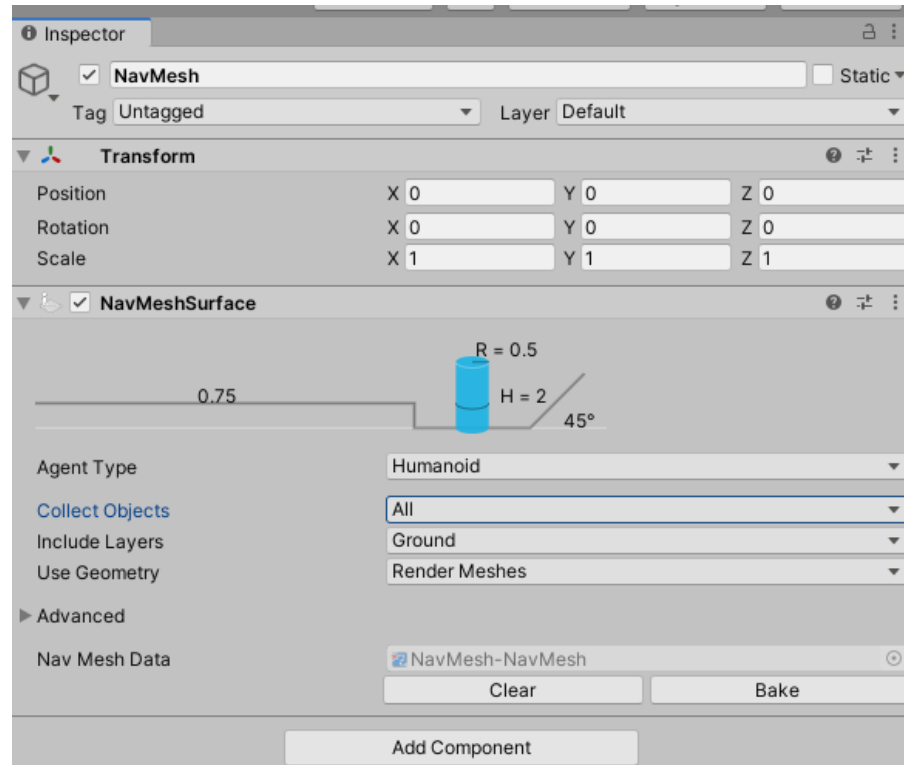
This is just Unity's normal NavMesh, see the many online tutorials and examples to further customize it, but this is a quick start.

1. Create an empty GameObject.



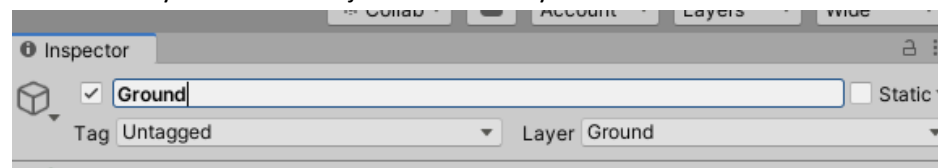
a.

2. Add a "NavMeshSurface" component to the empty gameobject.



a.

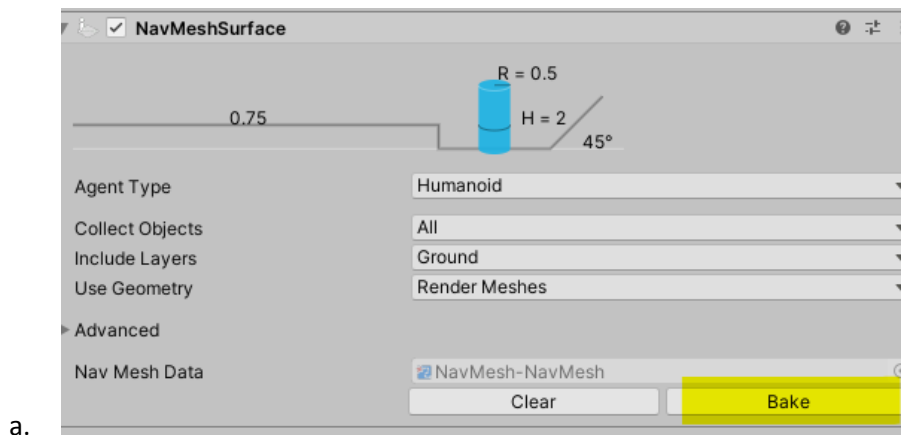
3. Ensure that "Include Layers" has Ground.
4. You must mark all of your Ground objects with the layer Ground.



a.

5. Click "Bake".

Quick Start - Ultimate ARPG - Return to [Main Document](#)



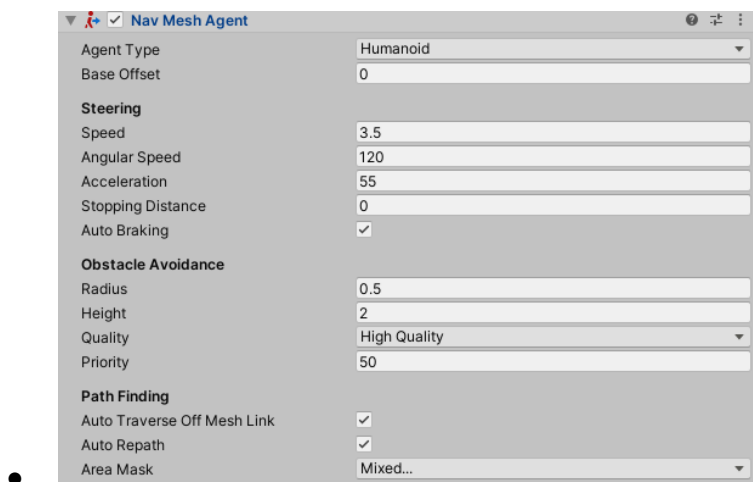
I highly suggest organizing all of your objects that are on the “Ground” layer to be a child of the NavMesh game object we just made. And to have “Collect Objects” be “Children”. (This isn’t required, just a good practice).

Done.

If using the ARPG Mouse Input, place the character into the scene. The character should now move upon entering Play.

The character won’t animate until animations are set in the Animator Override Controller or you control it with your own assets.

The NavMeshAgent is what controls the speed, acceleration, and other steering options. These are values I use for the demo:



Again, this is Unity’s NavMeshAgent, so see the various YouTube tutorials or documentation on how to further manipulate the Agent.

Quick Start - Ultimate ARPG - Return to [Main Document](#)

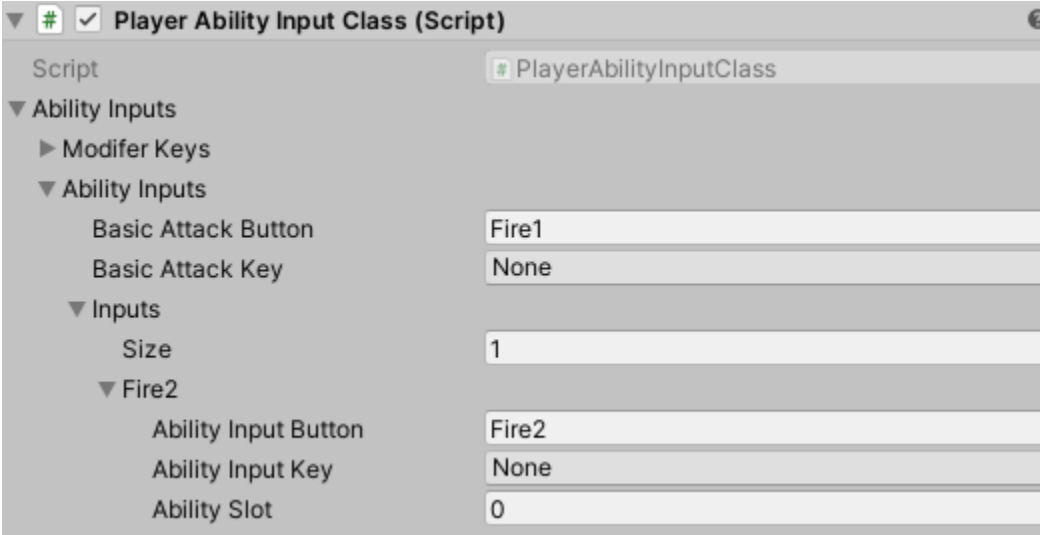
Quick Player Input Setup

The following classes control the player's input:

- ▶ # Player Mouse Input Class (Script)
- ▶ # Player Aura Input Class (Script)
- ▶ # ✓ Player Ability Input Class (Script)
- ▶ # ✓ Player Canvas Input Class (Script)

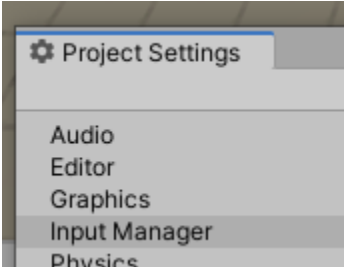
Open the classes and adjust the Buttons or KeyCodes, accordingly. If Buttons are left empty, they will be ignored.

In the example below, "Fire1" performs the basic attack. Unity's default "Fire1" is left mouse click. "Fire2" is right mouse click.

- 

Repeat the setup for the other classes (Mouse, Aura, Canvas).

The Buttons are strings and plug into Unity's Input system. To Adjust Unity's Input Buttons, navigate to **Edit -> Project Settings -> Input Manager**.

- 

See the various YouTube videos or guides on how to further customize Unity's Input Manager.

If you want to adapt Unity's new input system or a different one, have your new input class inherit the appropriate interface(s).