

## Quick Start Guide

- This document last updated: 1/25/2021
- [Link to most recent.](#)

## Contents

Quick Start Guide .....	1
Quick Creation of Database .....	2
Quick Creation of Actors .....	3
Quick Creation of Scene .....	6
Unity's NavMesh Setup .....	7
Setting Up Player Input .....	9

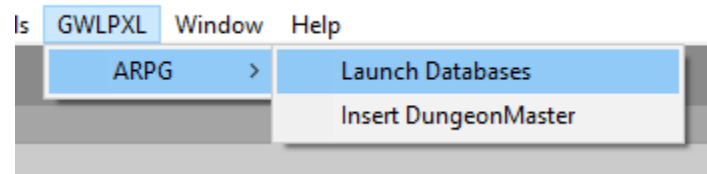
This quick guide assumes you go in order:

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

## Quick Creation of Database

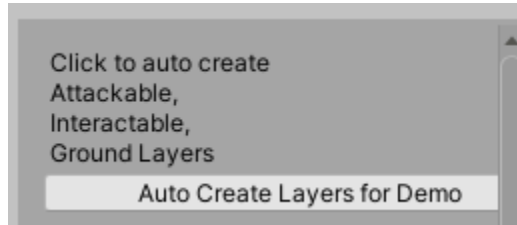
1. Launch the database launcher.

019.4.0f1 Personal [PREVIEW PACKAGES IN USE] <DX11>



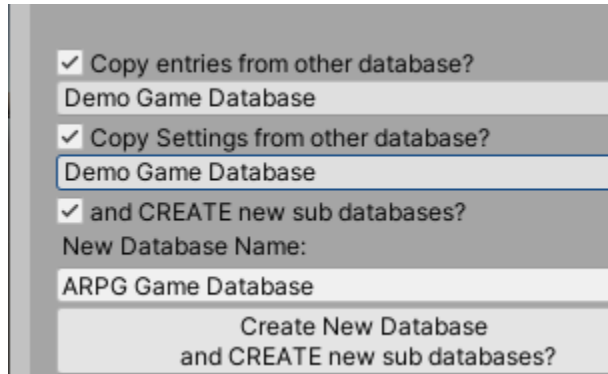
a.

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



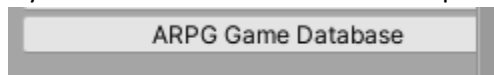
a.

3. Enable Copy Database and Copy Settings, select the Demo Database.



a.

4. Click Create and Create a new folder. Select the new folder as the destination for your databases.
5. Allow the asset to create and copy over the necessary files.
6. Once complete, relaunch the database window if it isn't still open.
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. The game database editor should open.

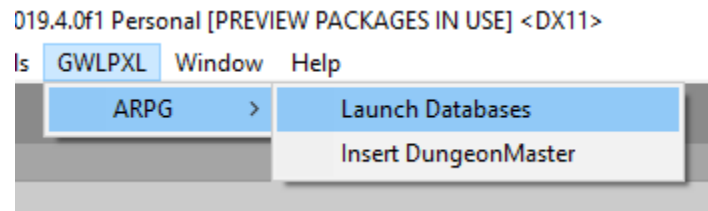
## Quick Creation of Actors

Demo characters are found in

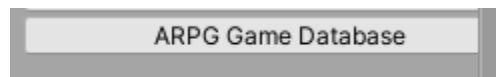
- Assets/GWLPXL/ARPG/Data/\_Characters/\_Player

You can drag in one of those prefabs or read on to learn how to create a new one.

1. Open the Launcher.



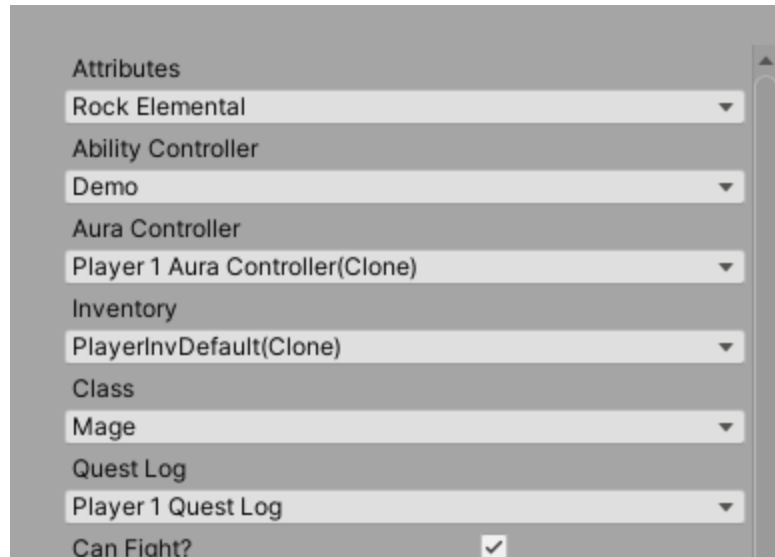
- a.
2. Select your database.



- a.
3. Choose the 'Player' option in under the "Create Actors".

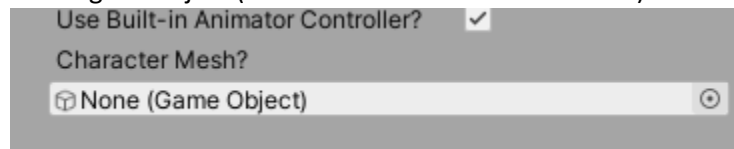


- a.
4. Assign the options (you can make custom ones later, just assign to see the process for now).



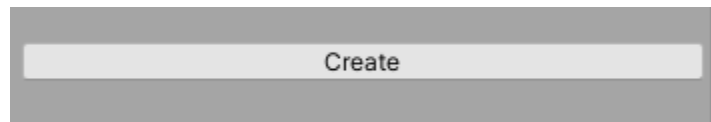
a.

5. Assign a mesh gameobject (the visualization of the character):



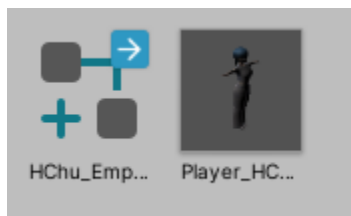
a.

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



a.

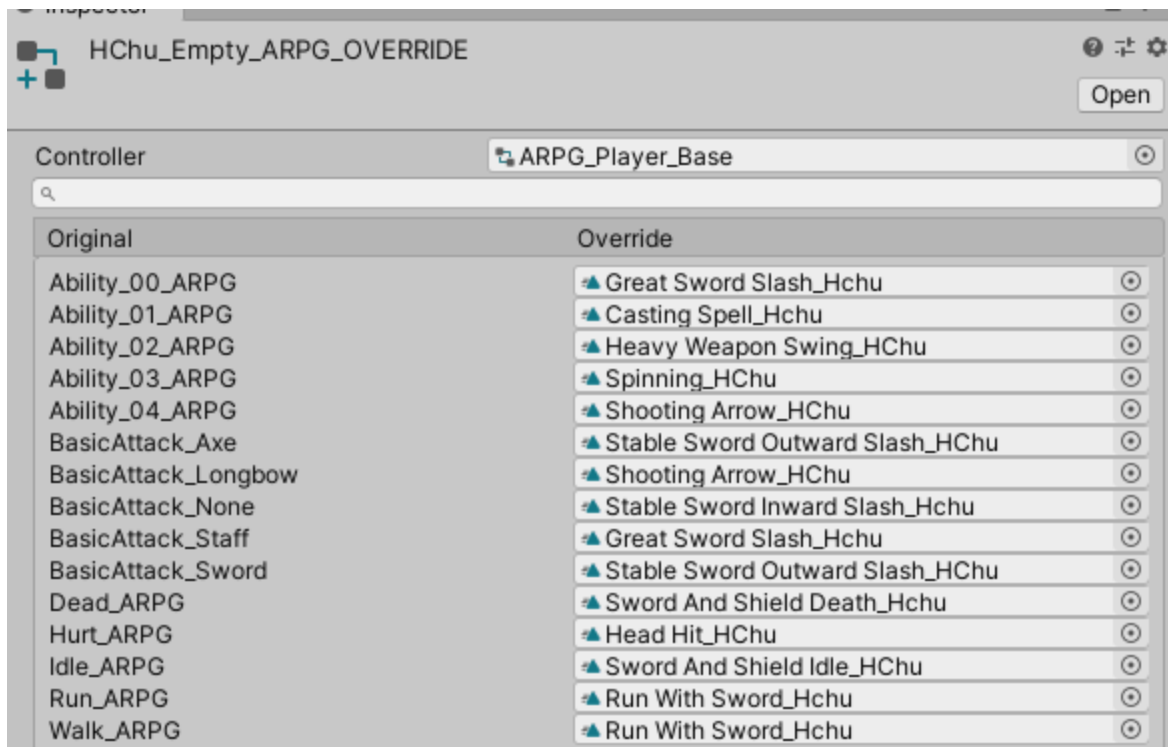
If you selected **mesh** and are using the **ARPG Animation system**, the Player and Enemy actors will create two things: The Prefab and an Animator Override Controller. Example pictured below.



•

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

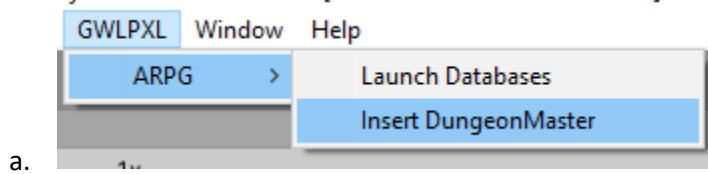
An example of an override with animations:



The Animator Override Controller is part of Unity's Animator system, see the various YouTube and Documentation on it to learn more about how to use them.

## Quick Creation of Scene

Insert the DungeonMaster singleton.



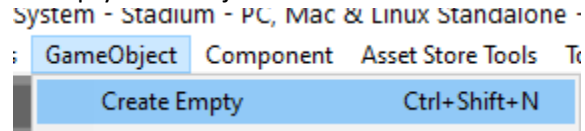
The DungeonMaster singleton controls the creation of the FloatingText, LootText, and Dungeon Canvas elements. It's what also what carries over the player data between scenes.

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.

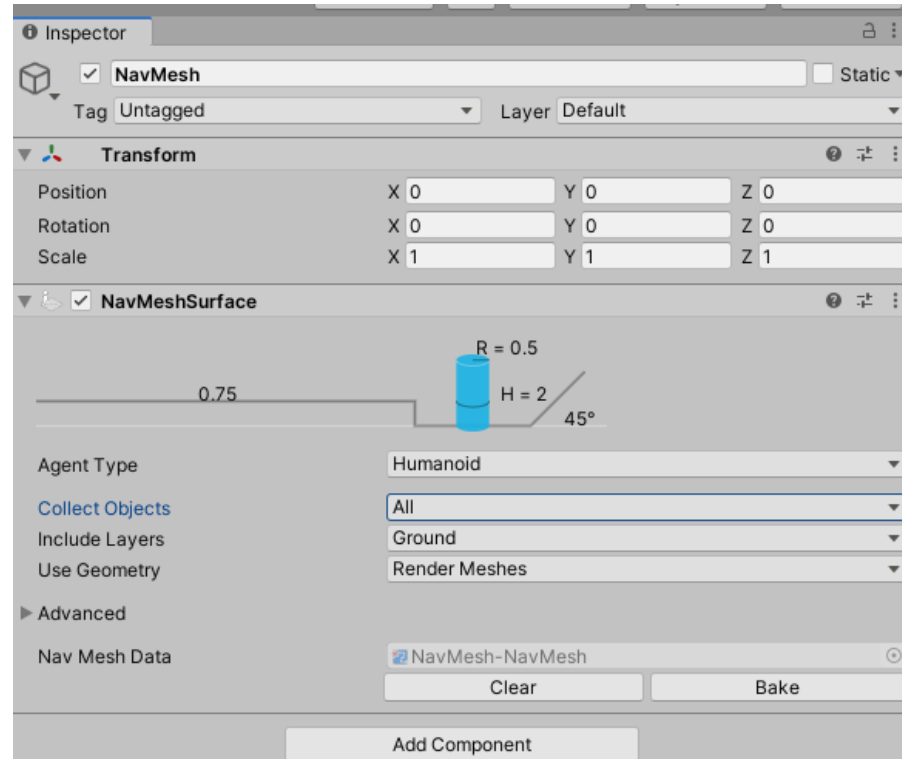
## Unity's NavMesh Setup

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

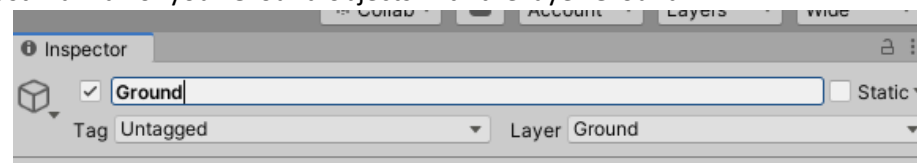
1. Create an empty Gameobject.



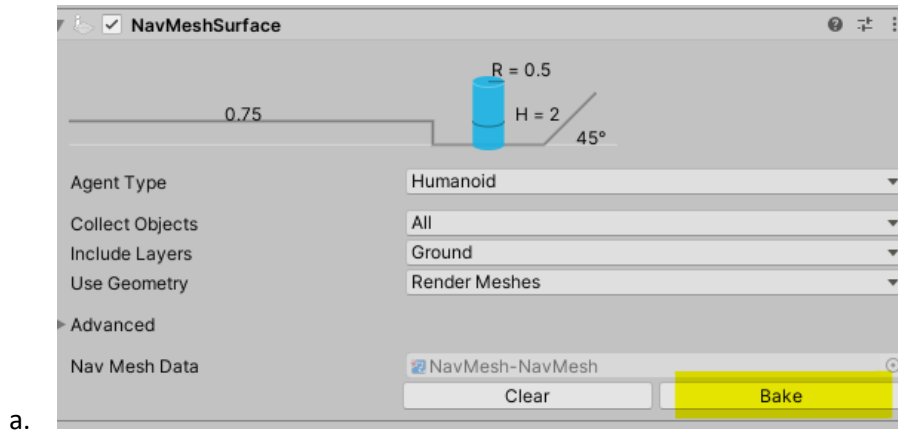
- a.
  2. Add a "NavMeshSurface" component to the empty gameobject.



- 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".



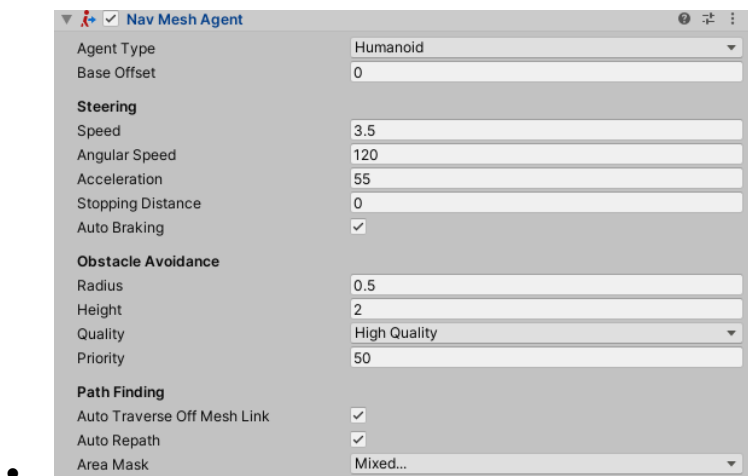
I highly suggest organizing all of your objects that are on the “Ground” layer to be a child of the NavMesh game object we just made. And to have “Collect Objects” be “Children”. (This isn’t required, just a good practice).

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.



## Setting Up Player Input

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)

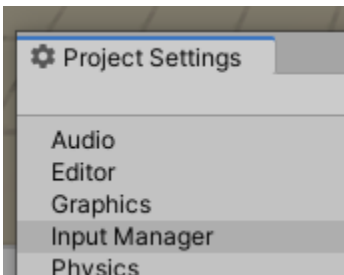
Open the classes and adjust the Buttons or KeyCodes, accordingly. If Buttons are left empty, they will be ignored.

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

- ▼ # ✓ Player Ability Input Class (Script)  
Script: PlayerAbilityInputClass  
▼ Ability Inputs  
▶ Modifier Keys  
▼ Ability Inputs  
Basic Attack Button: Fire1  
Basic Attack Key: None  
▼ Inputs  
Size: 1  
▼ Fire2  
Ability Input Button: Fire2  
Ability Input Key: None  
Ability Slot: 0

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 by navigating to Edit -> Project Settings -> Input Manager.

- Project Settings  
Audio  
Editor  
Graphics  
Input Manager  
Physics

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