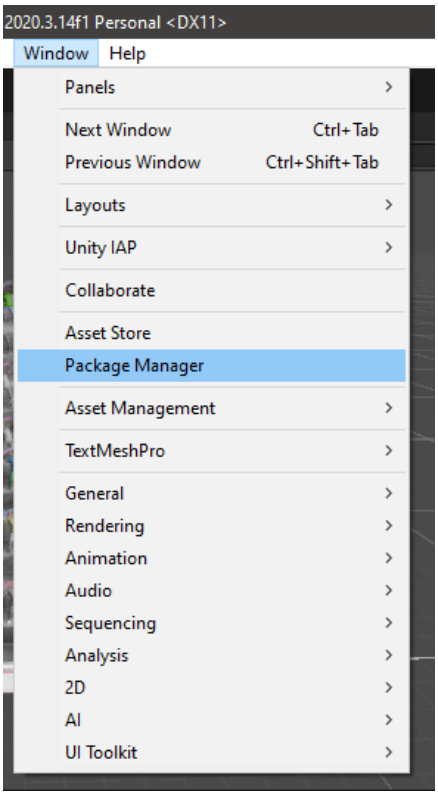
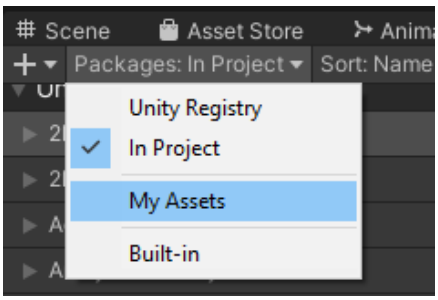
Crowd Simulation System

Installation

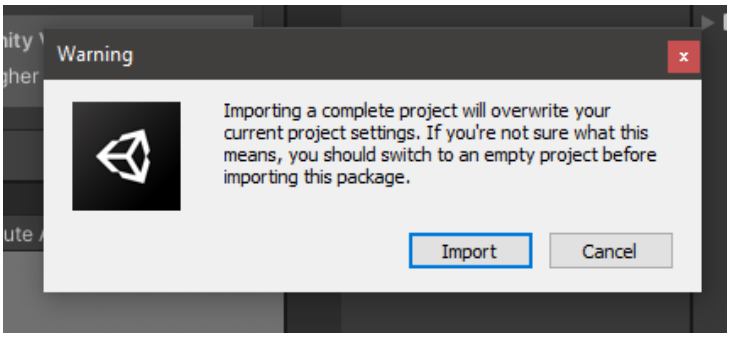
Crowd Simulator is extremely easy to get installed and working. The system will install quickly and doesn't require much hard drive space to install into your project.

Simply open up the package manager and select "My Assets" and then search for the Crowd Simulation System.



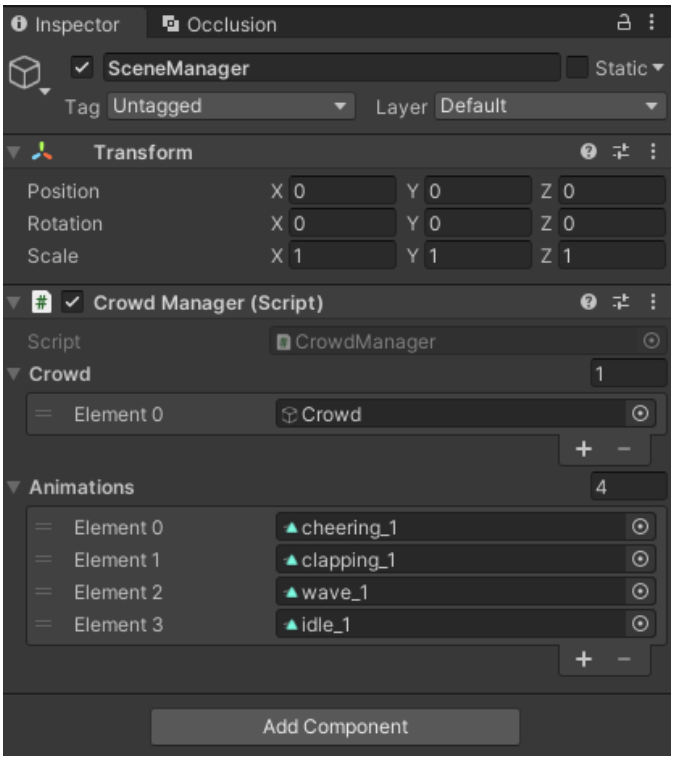


Crowd Simulation System using custom tags and layers to work and therefore Unity will inquire about overwriting current project settings. It is highly recommended that you back up your own settings first, and implement these tags and layers into your project. Alternatively, you can import this system into an empty project first and then merge.



Functionality

Crowd Simulation System is easy to use and doesn't require a lot of knowledge of Unity to use. The system uses custom tags and layers to identify the crowd assets, through the "SceneManager" game object. The "SceneManager" game object is vital to any scene that uses the system, do not delete it.



There are two types of crowds included in the package, these are single and grouped crowds.

Grouped crowds offer better performance and are the better option when creating large crowds.

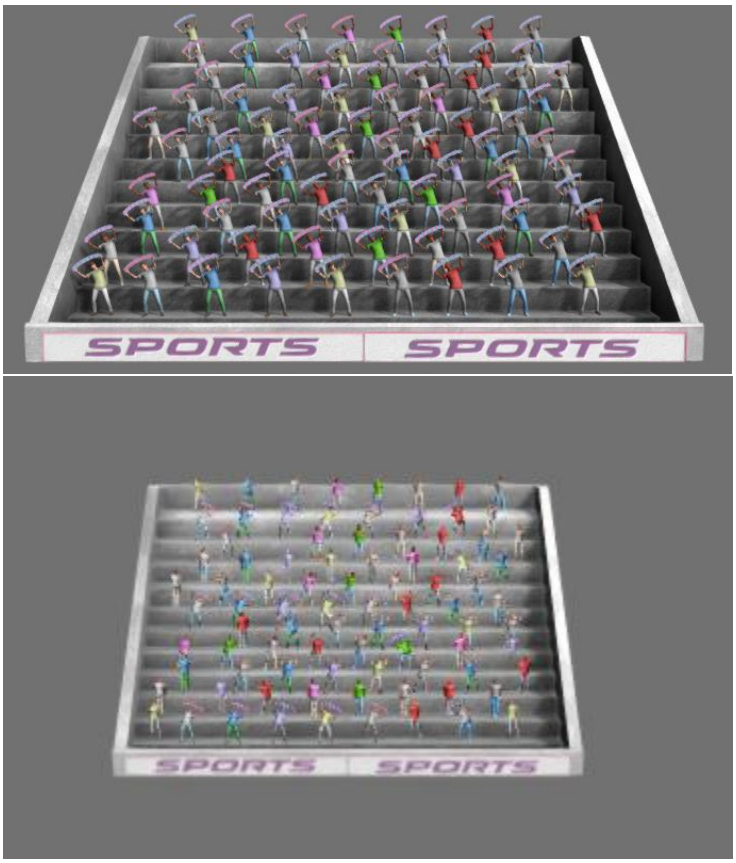
Single crowds are good to fill up gaps with, but should be used sparingly.

Both 3D and 2D crowds are implemented into the package. This is to ensure the performance always maintains an acceptable level. All the pre-defined crowds already have LOD's set up and as such are working with 3D and 2D dynamically already.

When the LOD shifts to a distant instance, the 3D models are replaced with 2D sprites.

Sprite sheets are all based off the 3D models, and as such always correspond to the correct 3D gameobject.

See below for a 3D crowd vs. a 2D crowd.



Implementation

This system is very lightweight and only contains a handful of code calls and functions. You can easily control the crowds yourself by implementing these functions in unique ways. In this example setup the system works through the interface buttons that are pressed but in a game scenario you would want these functions to happen dynamically. You can do this by calling the relevant functions at runtime.

In this system everything works through the “SceneManager” game object. The “SceneManager” game object contains a script called “Crowd Manager”, this is where the crowds are controlled from.

In this script all crowds are found at runtime and then called to with various functions.

```
public GameObject[] Crowd;

public AnimationClip[] Animations;

Unity Message | 0 references
public void Start()

{

    Crowd = GameObject.FindGameObjectsWithTag("Crowd");

    CrowdRandom();

}
```

So, if you would want to control the crowds through your own scripts, you can do so by calling any of the functions that control the crowds, through the “SceneManager” gameobject. Below is an example for calling the “Idle” state in all crowds.

```
0 references
public void CrowdIdle()

{

    foreach (GameObject CrowdPerson in Crowd)

    {

        if (CrowdPerson.GetComponentInChildren<Animation>())

        {

            CrowdPerson.GetComponentInChildren<Animation>().Play("idle_1");

        }

    }

}
```

So, as an example, lets say you have a soccer game and a goal is scored, when the ball crosses the line you’ll add a code line to your script that calls the “SceneManager” gameobject and then invokes the “Cheer” function.

Below is an example snippet:

```
if(GoalScored)
{
    0 references
    SceneManager.GetComponent<CrowdManager>().CrowdCheer();
}
```

Support

If you have any questions or comments, please email me at (alignedgames@mailbox.co.za)

I hope you enjoy the Crowd Simulation System and thank you for buying!