STADIUM CROWD GENERATOR

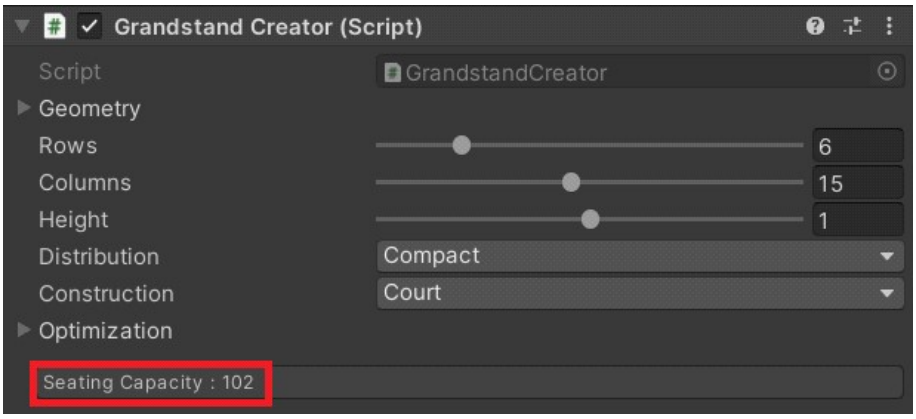
AIKODEX

ASSET OVERVIEW

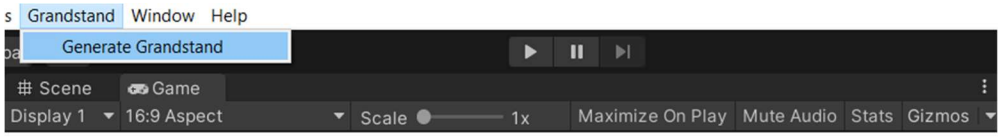
Editor

The editor folder contains the editor scripts that will:

- Help you know the seating capacity of the grandstand in the inspector.



- Help you in creating a template grandstand from a menu bar shortcut:



Materials
The material folder (different from spectator VAT materials) contains materials that form beams, planks, railing and seating. Their texture and color can be changed. This will be reflected in the grandstand materials once OnValidate() is called (often just by changing any parameter like rows and columns, the OnValidate() function will be triggered).

Models
The models folder (different from spectator models) consists of the building blocks of the grandstand in fbx format.

Prefabs
The prefabs folder (different from spectator unique VAT and skinned mesh prefabs) consists of the building prefabs that are instantiated during the OnValidate() method in the grandstandcreator.cs

Scenes
The scenes folder contains the scene files that have various features of this asset.

Textures
The textures folder consists of the textures contained in the grandstand building model.

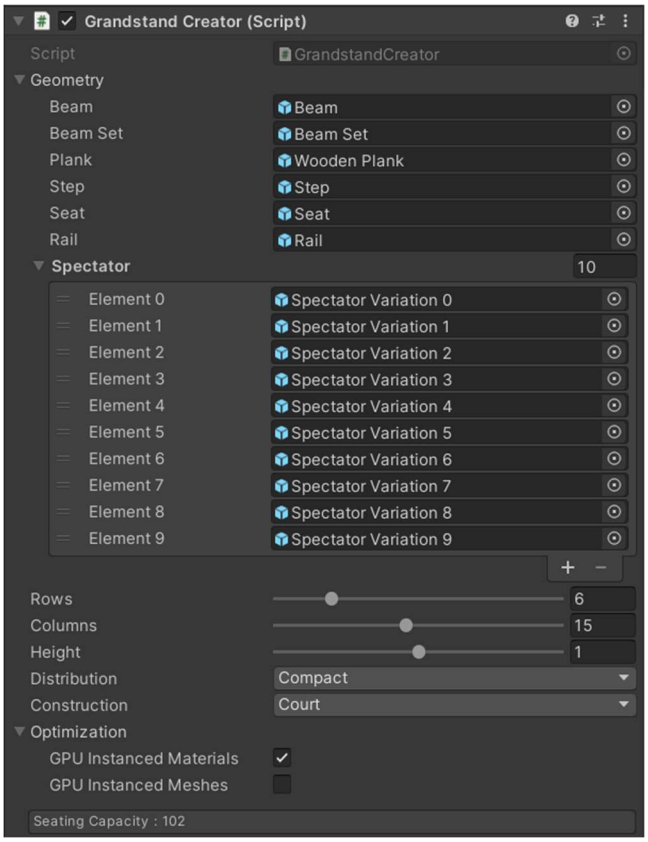
URP compatibility
Please note that you will require Shader Graph version 12.0.0+ to run the models on High performance VAT textures. As of October 2021, shader graph 12.0.0 can be found in Unity 2021.2+.

SCRIPTS

The scripts folder consists of the C# scripts:

- GrandstandCreator.cs
- SentimentHandler.cs

GrandstandCreator.cs works entirely on OnValidate() method wherein if anything is changed in the script from the inspector, the function will get called.
This method works by first destroying all the gameobjects found in the Grandstand transform. Then using for loops, it recreates the position of the beams, planks, steps, rails and seats. Since the models have specific dimensions, the transform values passed in the instantiate function have been tweaked to snap to each other without gaps.

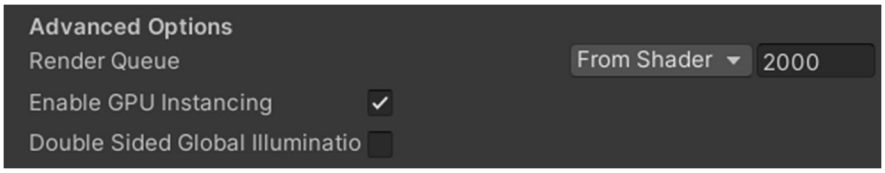


- **Geometry class**: holds all the data for building prefabs as well as spectator prefabs. Please note that using if(geometry.x!=null) in the script will make the field option. An example of this is the step object which is optional to use in the grandstand and would not throw an exception if it were null. If you want to add your own material/texture, please edit the prefab and connect it to the geometry class public members.

- **Rows**: Specifies the number of rows the grandstand must contain

- **Column**: Specifies the number of columns the grandstand must contain

- **Height**: Specifies the height of the grandstand.
- **Distribution**: Specifies the spacing between the spectators.
- **Construction**: Specifies the architecture of the grandstand.
- **Optimization [GPU Instanced Materials]**: If enabled, will enable the GPU instancing in materials



- **Optimization [GPU Instanced Meshes]**: Enabling this option will scan through all the children of the main parent and will store all the information at into Lists and Arrays in the Start function. It will then pass on this information to the DrawMeshInstanced() function in the update cycle. Check this option for a massive boost in FPS. From around 200 for 100 skinned mesh renderers to 550 on enabling VAT GPU instanced meshes.

More on GPU Instanced Meshes

GPU instancing using DrawMeshInstanced can draw up to 1023 meshes in a single draw call. Adding a new spectator prefab will take in all its positions, rotations and scales into its matrix4x4 and display it in a single draw call. This is extremely efficient way of rendering and removes the game object overhead completely. This technique can potentially render thousands of meshes at a single time and thousands of spectators at a single go maintaining a very high framerate.
Some considerations while using this technique for your project:

- The names of the grandstand building objects are independent but for spectators, the algorithm is name specific. The names of the VAT generic prefabs **must begin with "Spectator Variation x" or "Active Emotions x".**
- While this option is checked and you try to change the parameters of the grandstand, the script will fail to load the grandstand. Please check this option only when you have completely configured the grandstand and are ready to hit play.
- As of now, the GPU instanced meshes option does not work with Active Emotions.

SentimentHandler.cs is used for VAT animation switching. It works by replacing the materials of the prefab at runtime to the desired emotion of the crowd. As of now, there are only four emotion textures provided for every one of the 10 characters : Light Cheer, Heavy Cheer, Criticizing, Disappointed.

SPECTATORS

The spectators folder consists of:

- Shaders
- SkinnedMesh Color Offset Prefabs
- SkinnedMesh Prefabs
- Spectator Variations 0-9
- VAT Prefabs

Before discussing the contents of the spectators folder, VAT technology needs to be understood first.

Vertex Animation Textures (VAT)

The Vertex Animation Texture (VAT) technique, also known as Morphing Animation, consists of baking animations into a texture map for use later in the vertex shader. This technique enables us to keep the mesh completely static on the CPU side (changing the SkinnedMeshRenderer component to MeshRenderer component) and use the encoded vertex information from the baked texture to move vertices with a decoding shader on the GPU.

Vertex Motion is encoded from XYZ > RGB to form a texture. Texture is decoded from RGB > XYZ to form animation.

With this technique, the CPU no longer has to animate bones, skin the mesh or have a lingering animation overhead. SkinnedMeshRenderer is inefficient for rendering a live crowd with repetitive actions. This can add anywhere from 100FPS to 300+FPS on very large crowds. Using this technique, 4000 animated unity Adams were able to run at 90FPS (Yifei Boon – Unity field engineer).

This technique has not been widely adopted as it is recent and very less documentation and resources are available for developers to take advantage. With this asset we hope to widen the user base for the usage of this very effective technique.

Contents of the Spectator Folder:

Shaders: The shaders folder consists of RP specific shaders to run the VAT animations. Built-in uses cginc assemblies and URP uses shader graph 12.0.0. There is another shader included which works by offsetting the color and world position of the mesh – Standard Spectator Offset.

Skinned Mesh Prefabs: These prefabs have skinned mesh renderers attached to them with the Standard Shader.

Skinned Mesh Color Offset Prefabs: These prefabs have skinned mesh renderers attached to them with the Standard Spectator Offset shader. These prefabs do a little better performance wise compared to only skinned mesh renderers.

Spectator Variations 0-9: This folder contains the original models, their animator controllers and the VAT materials and textures for the specified RP.

VAT Prefabs: Divided into two parts; Active Emotion prefabs 0-9 and Spectator Variations 0-9. Spectator Variations 0-9 play designated animations for a certain character whereas Active Emotion prefabs can be controlled by sentiment Handler script. Live Audience scene has an example of how active emotions can be utilized.

Extensions

If one wants more variations and feels that even with 10 animated prefabs, they can still see repetitions, more prefabs with a few changes can be added to eliminate repetition. There are a few performance implications however, with every new prefab there will be additional batches. Only in the case of Skinned Mesh world offset prefabs there would not be a significant increase in batches.

Grid based UV mapping (Color Atlases)

By changing the Offset value in the shader (in increments of 0.1 from 0-1) one can achieve a different color for different parts of the Spectator Mesh.

From left to right in increments of 0.1 vertically all the colors are arranged. Matched with different 10 different animations with different delta times and 10 meshes with 10 different meshes, one can theoretically make 1000 different spectators (albeit with minor changes).

If you have any questions, suggestions or feature requests, please send us an email at info@aikodex.com and we'd be more than willing to answer and incorporate them!