Clayxels Documentation

For support please use discord: https://discord.gg/Uh3Yc43
You can also contact me via email for refunds and business enquiries: ainterguglielmi@gmail.com

Follow the updates on twitter to see what's coming up next and what other people are working on:
https://twitter.com/clayxels

Table of Contents

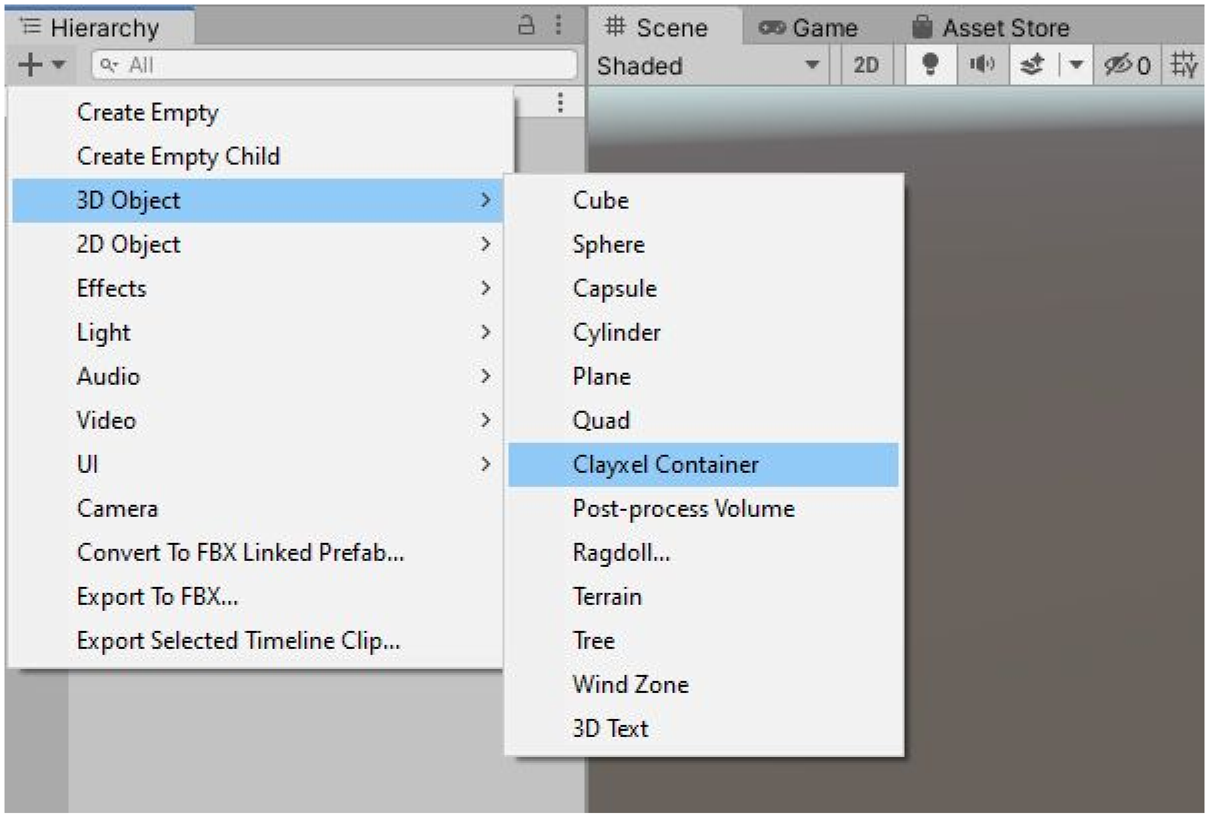
1) Introduction

Clayxels is an interactive volumetric toolkit that doesn't rely on per-pixel ray-marching. Instead it produces a compact and lightweight point-cloud that can be used in a whole bunch of different ways. Both in editor and in game.

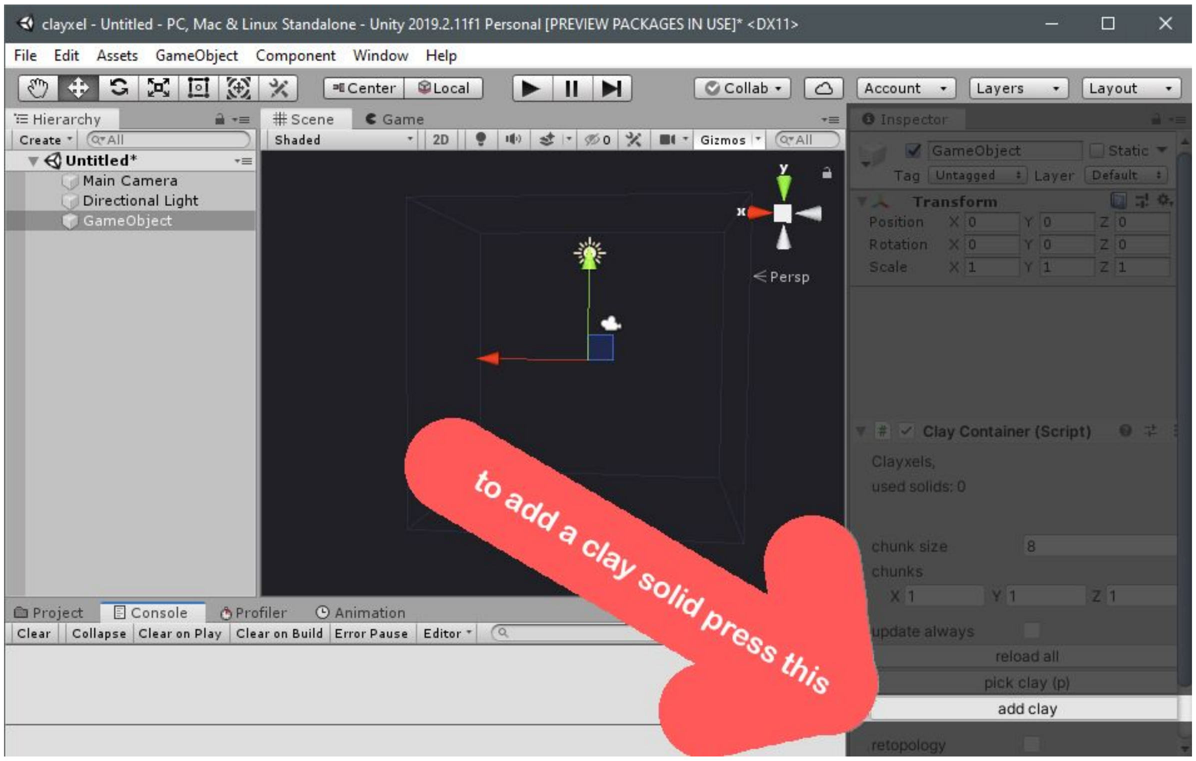
Clayxels will work on all render pipelines in Unity 2019.3 and 2019.4.

2) Getting Started

Getting started with Clayxels is simple, first create a ClayContainer using the sub-menu located under 3D Objects.

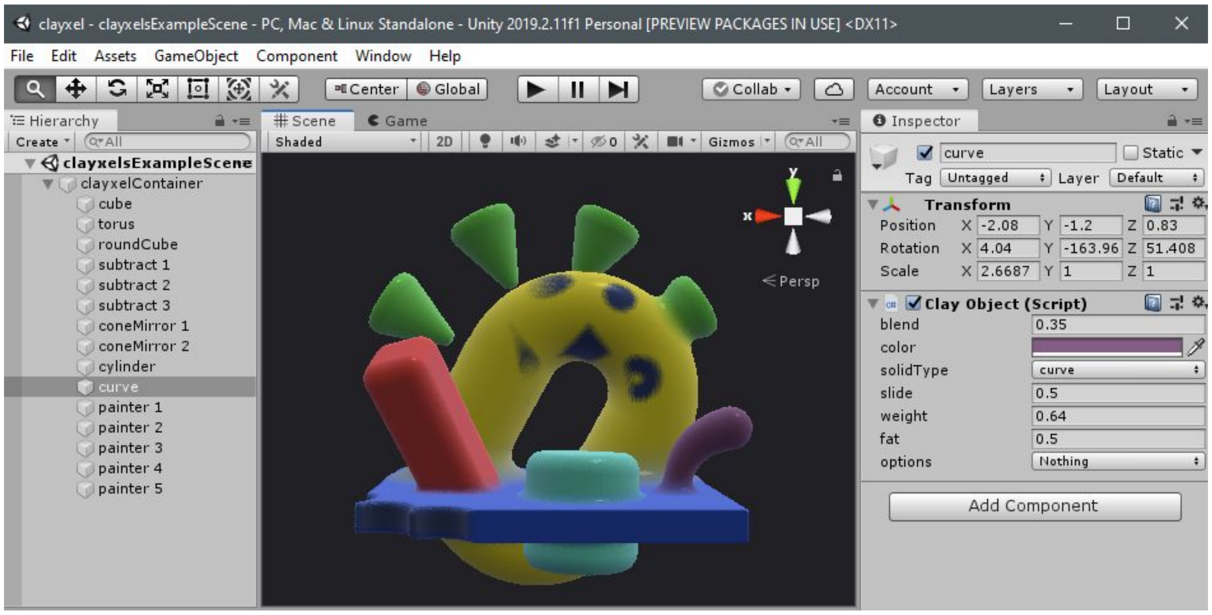


Now that you have a ClayContainer you can start nesting ClayObjects under its hierarchy.
To add a ClayObject press the "add clay" button, it's located in the custom inspector that shows up whenever you select a ClayContainer.


clayxel - Untitled - PC, Mac & Linux Standalone - Unity 2019.2.11f1 Personal [PREVIEW PACKAGES IN USE]* <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Collab Account Layers Layout
Hierarchy
Create
Untitled*
Main Camera
Directional Light
GameObject
Scene Game
Shaded 2D Gizmos
Persp
Project Console Profiler Animation
Clear Collapse Clear on Play Clear on Build Error Pause Editor
Inspector
to add a clay solid press this

GameObject Static
Tag Untagged Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Clay Container (Script)
Clayxels,
used solids: 0
chunk size 8
chunks
X 1 Y 1 Z 1
update always
reload all
pick clay (p)
add clay
retopology

ClayObjects have a few options of their own showing in the inspector. You can change their shape, the blend value to add or subtract shapes together, change their color and other properties that will vary depending on the solid-type.



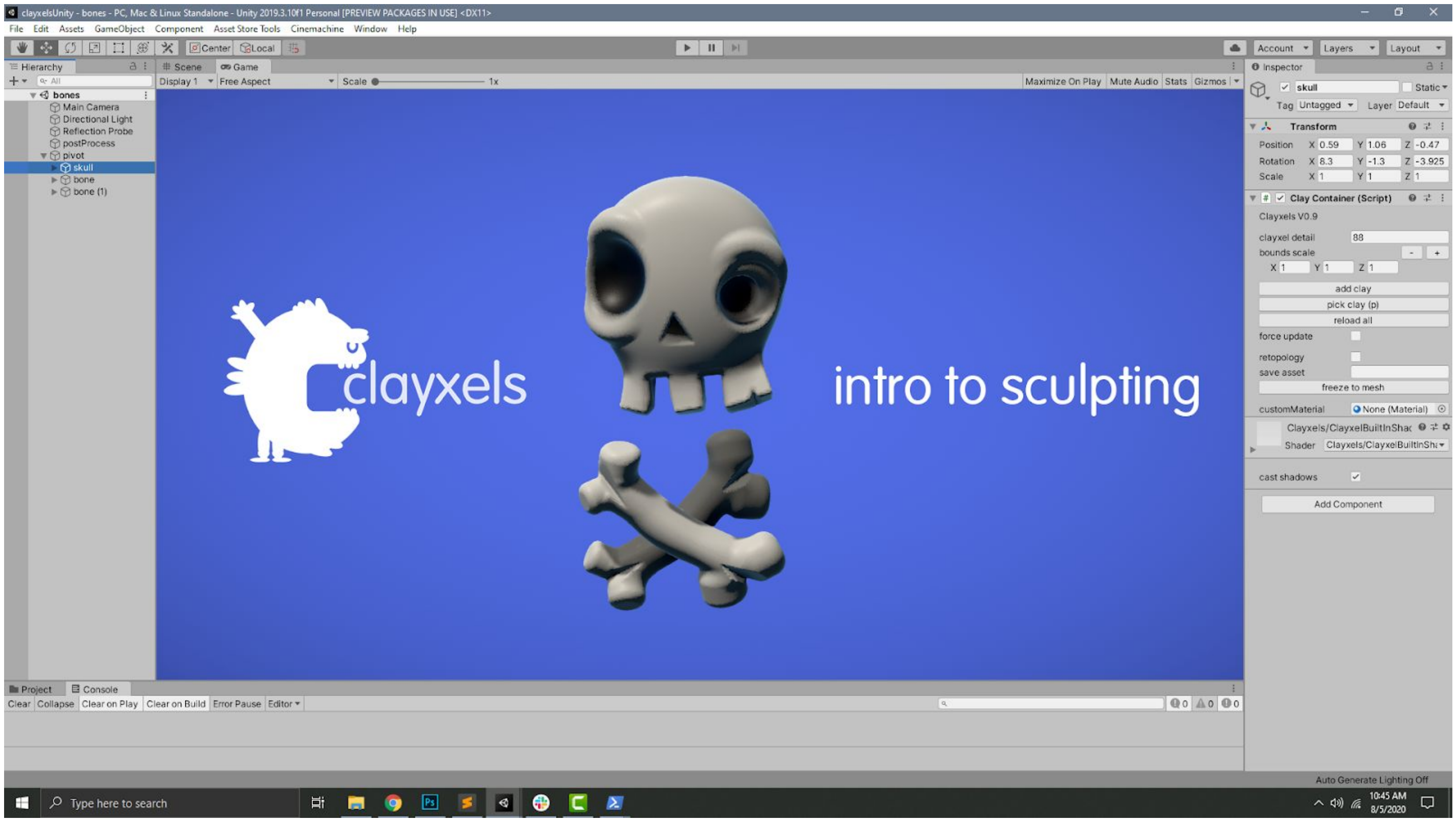
curve Static
Tag Untagged Layer Default
Transform
Position X -2.08 Y -1.2 Z 0.83
Rotation X 4.04 Y -163.96 Z 51.408
Scale X 2.6687 Y 1 Z 1
Clay Object (Script)
blend 0.35
color
solidType curve
slide 0.5
weight 0.64
fat 0.5
options Nothing

Please check the tooltips on the inspector, that’s the best way to learn this simple UI.

3) Usage Basics

- Press "p" on your keyboard to pick-select clayObjects directly from the viewport.
- Drag solids up and down the hierarchy to change how they affect each other (solids influence each other from bottom to top in the hierarchy).
- Split your sculpt into many containers to make large and complex models.

Here's a video tutorial to get you started on sculpting with Clayxels:
https://youtu.be/pJOk7aRLYzY



Just don't forget to have a look at the examples shipped with the package, Clayxels can do so much more than just sculpting static assets, and the examples are the best way to learn all of that.

4) Troubleshooting

- To allow lower-end integrated graphic cards to work with clayxels you might need to specify a lower amount of maximum threads.
To do that, open the text file Clayxels/Resources/claySDF.compute and change #define MAXTHREADS to a value of 4.

- Some older video cards will feel particularly slow even with just one clayObject in scene, if that's the case go to Project Setting -> Player -> Other Settings -> Scripting Define Symbols, then add:
CLAYXELS_GPU_FIX1;

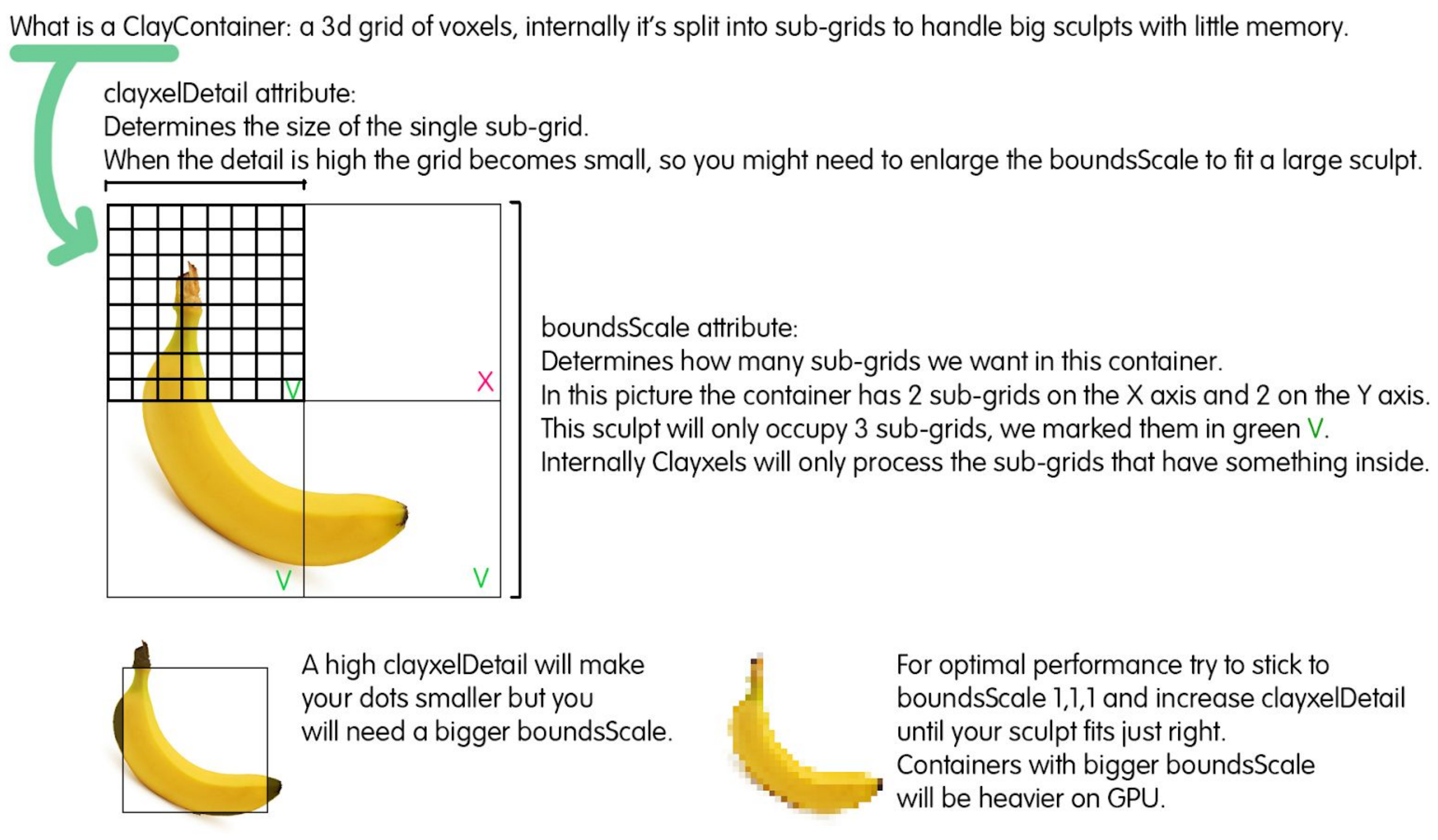
5) Optimizing

Clayxels is a GPU heavy system, but there are many ways to optimize it and eventually reduce its impact on the scene to the point of making it just as light as any other polygonal mesh. It mostly boils down to whether you need interactive shapes or not.

Non-interactive:
We refer to non-interactive clayxels when you sculpt from the editor but you don't change the clay shapes while the game is running. This workflow is the easiest to handle, as it will not impact your game's runtime performance.
Clayxels will re-compute the point cloud only when you move one of the ClayObjects inside a ClayContainer. But we don't need to do that at runtime unless we want to. The easiest way to use Clayxels in a game and avoid slow downs, is to simply move/ rotate/scale the ClayContainer and never touch the ClayObjects inside it once the game starts.
Clayxels will automatically optimize each sculpt to use as little memory as possible and you can expect each container to

perform as fast as any other mesh in your game.

Interactive:
If you plan on changing the content of your ClayContainers while the game is running, then it's important to understand how to use the bounds of your container effectively.
The following scheme is for artists and developers to understand how the underlying voxel-grid impacts your game when clayxels updates the point-cloud.



What is a ClayContainer: a 3d grid of voxels, internally it's split into sub-grids to handle big sculpts with little memory.

clayxelDetail attribute:
Determines the size of the single sub-grid.
When the detail is high the grid becomes small, so you might need to enlarge the boundsScale to fit a large sculpt.

boundsScale attribute:
Determines how many sub-grids we want in this container.
In this picture the container has 2 sub-grids on the X axis and 2 on the Y axis.
This sculpt will only occupy 3 sub-grids, we marked them in green V.
Internally Clayxels will only process the sub-grids that have something inside.

A high clayxelDetail will make your dots smaller but you will need a bigger boundsScale.

For optimal performance try to stick to boundsScale 1,1,1 and increase clayxelDetail until your sculpt fits just right.
Containers with bigger boundsScale will be heavier on GPU.

Freezing:
Freezing containers to mesh is yet another way to make clayxels more transparent on the final game executable.
Once frozen, the clayxel runtime will never run on the target platform, making clayxels easier to deploy on lower-spec platforms like mobile.

Fine tuning performance from clayxelsPrefs.cs:
- *ClayContainer.setMaxSolids*: keep it to a minimum to reduce video memory consumption.
- *ClayContainer.setMaxSolidsPerVoxel*: keep it to a minimum to improve runtime performance. Avoid placing too many clayObjects in a single voxel if you set this to a small number.
- *ClayContainer.setUpdateFrameSkip*: try skipping as many frames as possible to increase FPS in your game.
- *ClayContainer.setChunksUpdatePerFrame*: If your containers use boundsScale bigger than 1,1,1, this will limit the number of sub-grids to compute on each frame. The resulting point-cloud is shown on screen once all sub-grids have computed, so it might take multiple frames to see a container update depending on how big is your boundsScale.

6) Programming

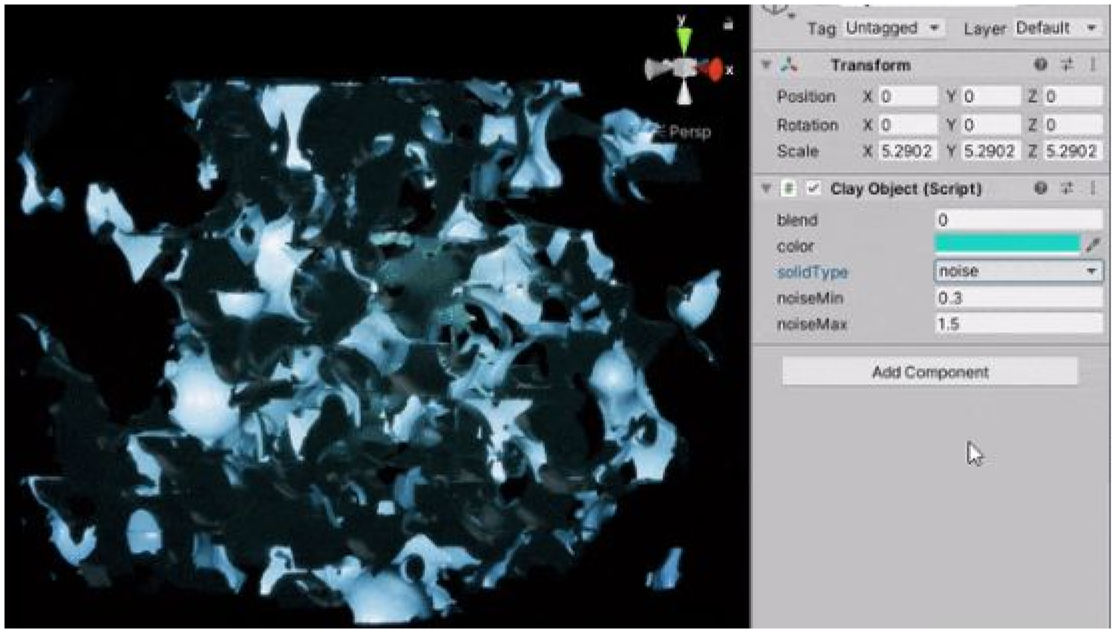
Clayxels has three main parts that can be used from code and expanded, c# APIs, signed distance functions, shading.

C# APIs:

The c# apis of clayxels are defined in ClayContainer.cs and ClayObject.cs, clayxels has all its c# code open for those that need a deep understanding of its inner workings.

The high level public interface is kept at the very top of the file and each public member has been documented for clarity.

Signed Distance Functions:



Tag Untagged Layer Default

Transform

Position X 0 Y 0 Z 0

Rotation X 0 Y 0 Z 0

Scale X 5.2902 Y 5.2902 Z 5.2902

Clay Object (Script)

blend 0

color

solidType noise

noiseMin 0.3

noiseMax 1.5

The core of clayxels is a compute shader that can be expanded via claySDF.compute, *computeClayDistanceFunction* is the single function responsible for handling all the volumetric shapes you see available from the inspector.

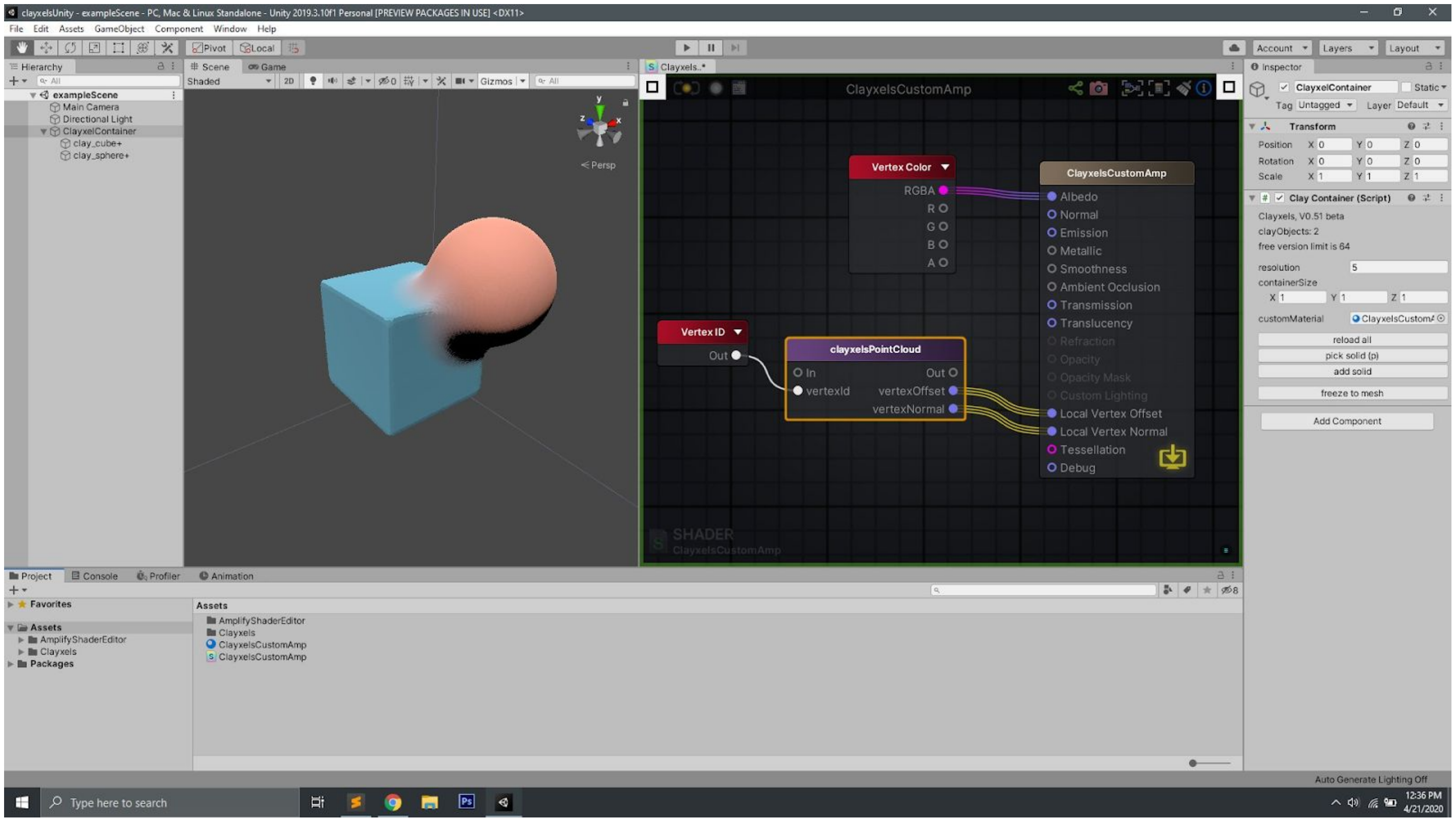
To expand clayxels with new solid-types all you need to do is chain your code within that function and declare a new interface.

Here's an example of how to do that (code to be added in claySDF.compute, *computeClayDistanceFunction*):

```
else if(solidType == 7){// label: mySDF, x: myParameter1 1.0
  // the comment above is auto-parsed from the inspector to register your custom SDF function
  // myParameter1 uses extraAttrs.x to get a value from the user, the inspector will show this parameter when
your custom SDF solid is selected.
  // 1.0 is the default value we are using for myParameter1
  dist = length(pointRotated) - myParameter1; // we just made a sphere with a radius equal to myParameter1
}
```

Whenever you change something inside claySDF.compute remember to click “reload all” from a container.

Shading



Shaders in clayxels can be written via code or wired visually using Amplify Shader Editor*.
These shaders have to be specifically written to render the buffer coming from the compute shader.
The best way to start with shading clayxels is to refer to the examples provided and the default shaders for each render pipeline.
Adapting shaders to the different render pipelines is usually just a matter of starting from the default clayxels shader on a given render pipeline (example ClayxelHDRPShader.shader), and then copying the nodes over from one of the existing shaders (example ClayxelFoliageShader.shader) using Amplify Shader Editor.

* The reason why you can't use Unity's built-in shader-graph is that it lacks the vertexId node. Hopefully this is something that Unity will eventually resolve in future versions.