Concrete And Steel Modding SDK 0.1.1
April 2016 - Matterhorn Software LTD

Create a Buildable object (mod) for Concrete And Steel

Index:

Introduction & Prerequisites

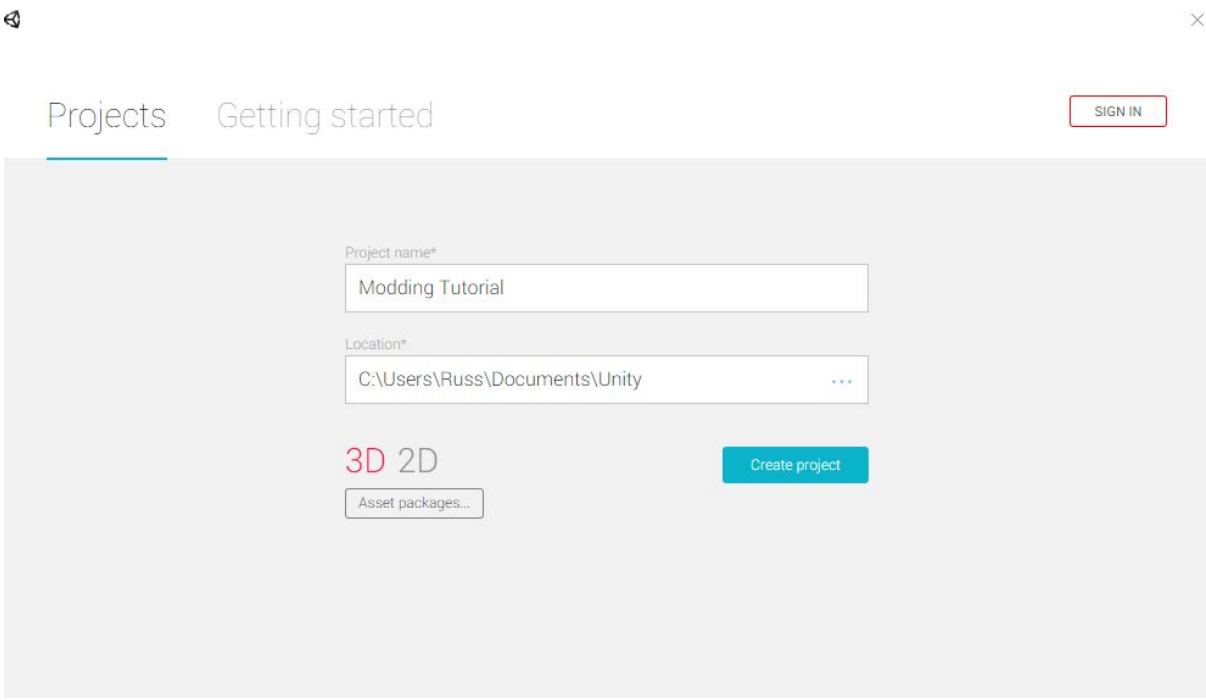
This documentation primarily explains the concepts for creating a Concrete And Steel “buildable object”, including setting up the editor for operations which are essential for composing the mod.

A “buildable object” is an object within the Concrete And Steel game which is used for construction. The object’s purpose may be for decorative effect or as part of a structure.

You should already have composed the 3d object you wish to import into the game. Furthermore, this documentation does not go into how to texture your object, set the hierarchy of the objects or import and apply materials or textures. It will really help if you already have a good grasp of how the Unity editor works.

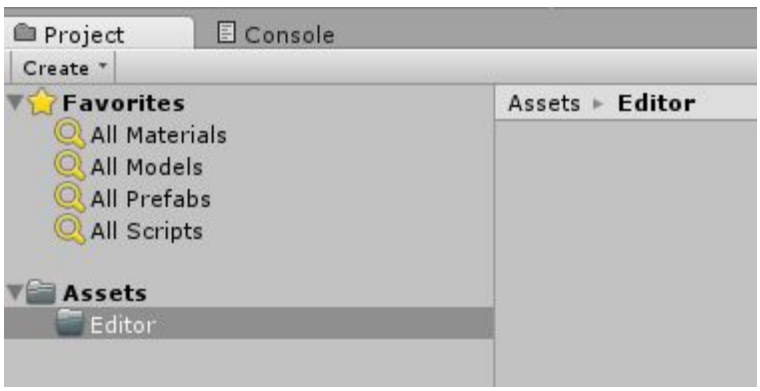
Set Up Unity

First you need to install Unity editor. Once it's installed, run Unity, create a new project - you can name this whatever you want. Choose 3D, with no asset packages.

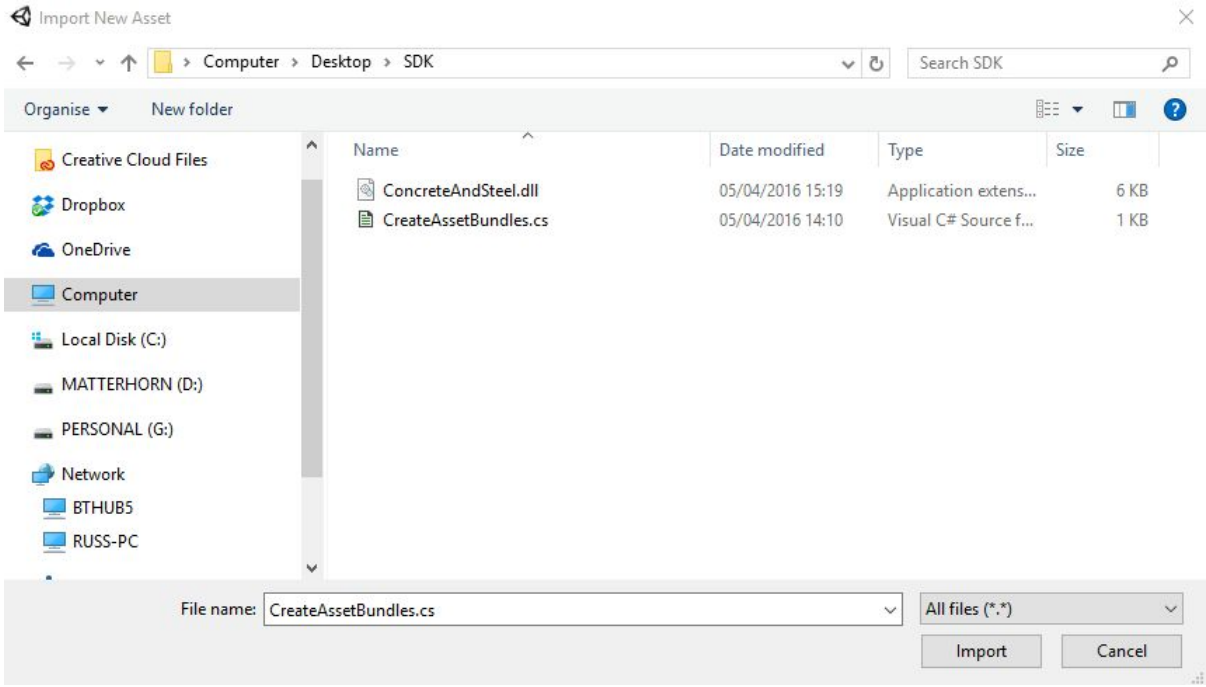


Import Editor Script

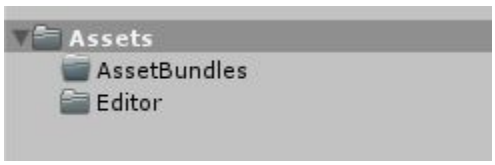
In order to export this mod as an "assetbundle", we need an editor script. Create a folder under "**Assets**" called "**Editor**"



Next, on Editor, right click and select "Import New Asset". We're going to import the C# script "**CreateAssetBundles.cs"**



We now also need to create a folder called “**AssetBundles”** under “**Assets**”:



This is where our mod will get exported to.

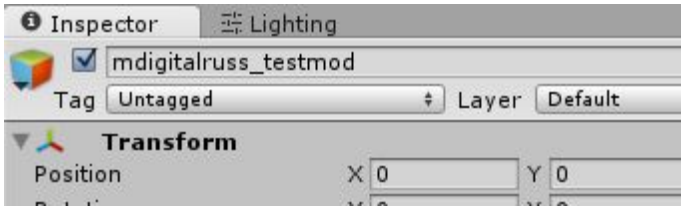
Import DLL

To use Concrete And Steel’s features, we need the special Concrete And Steel modding DLL. Under Assets, right click and select “**Import New Asset**” and choose “**ConcreteAndSteel.dll**” (in the same way as we imported the C# script).

Create and name the Buildable object

Now in Unity, in the main toolbar, select “**GameObject > Create Empty**”. This empty object will be our Buildable container. In the **Inspector,** rename the object to something unique. I recommend you prefix it with your username, to avoid accidentally overwriting someone

else's mod. I'm going to call this mod "***mdigitalruss_testmod***". Press Enter to make it happen.



Set Object tree

Our mod needs a structure like below:

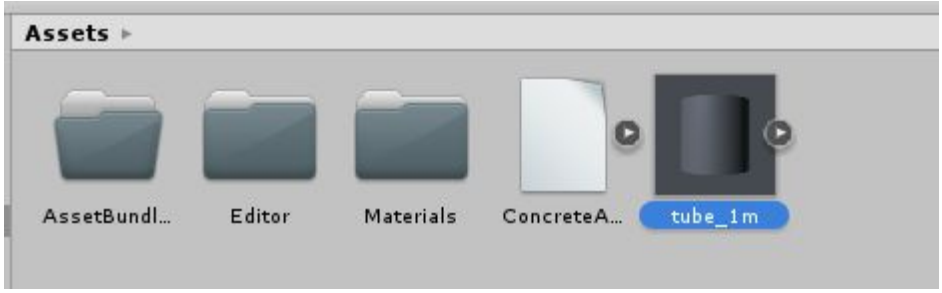
- Root Object
    - Aim Root
        - Aimer object
    - Built Root
        - Built Object

Under our mod, we should now create another empty gameobject. Right click on our root object we just made (in my case, *mdigitalruss_testmod*), and click "**Create Empty"**, and name it **"Built".**

Import objects

Back in our Project view, under "**Assets**", right click and select "**Import New Asset"**. In this example, I'm going to import a maya file called "***tube_1m.mb*".** This takes a while, as Unity is painfully slow at importing maya files. It will automatically import the materials from Maya into the "**Materials**" folder.

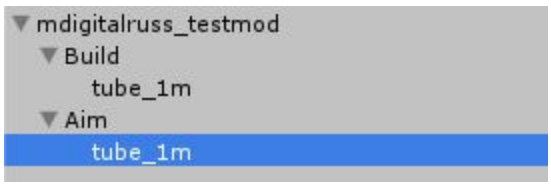
Here's my project so far:



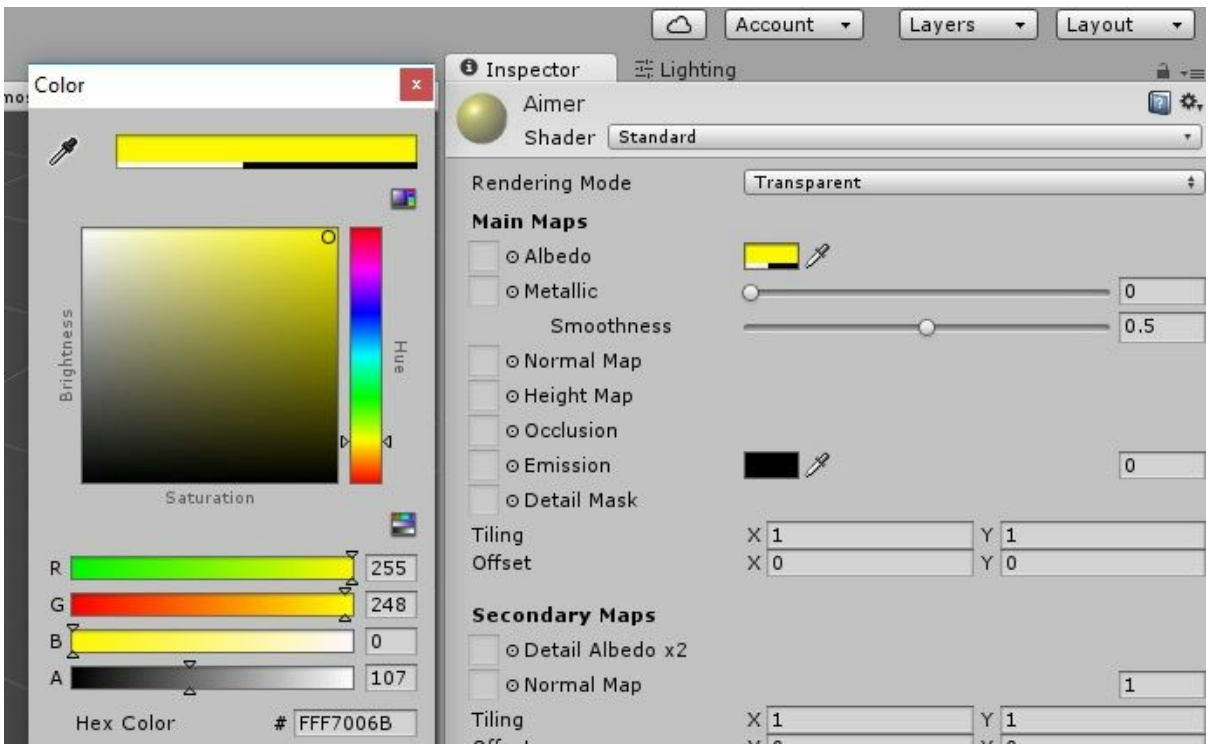
I'm going to drag "**tube_1m**" onto our "**Build"** gameobject that we just made:

Create an Aimer object

So far so good! Before we carry on, we need to create an Aim object. This is the object that will appear while we're aiming the object, before we build it. Duplicate what we've done so far by selecting "**Build"** and pressing **CTRL+D.** Rename it to "**Aim**".



Now, in the project view, right click and select "**Create > Material**". Name the material "**Aimer**". In the Inspector view, set "**Rendering Mode**" to "**Transparent**", set the **Albedo** colour to something like ***#FFED00AB*** (In the Hex Colour box at the bottom):



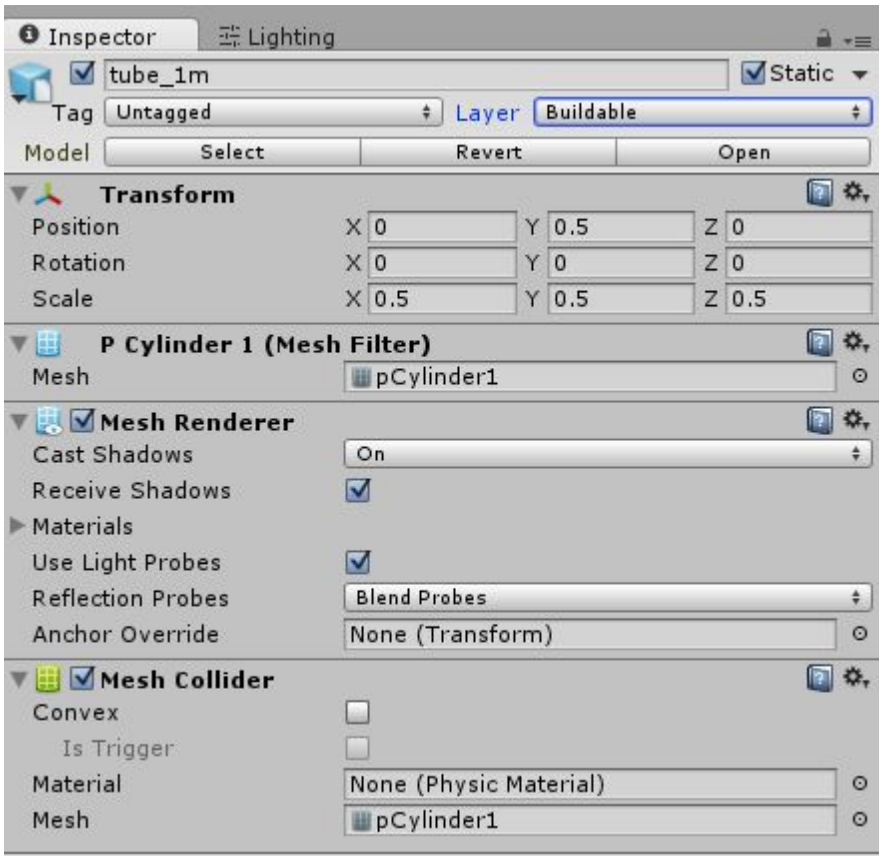
Now drag this material onto our "**Aim**" object in the **Hierarchy view**. This material will render the "**Aim"** object in a highlighted and semi-transparent way.

Make the Buildable

Now we need to configure our buildable object and assign the script.

With "***tube_1m*"** (under "**Build**") selected, in the Inspector tick "**Static**". Then select "**Layer > Add Layer",** and in the box next to User Layer 8 type "**Buildable**". Then go back to tube_1m, back to the inspector, and click "**Layer > Buildable"**.

At the bottom of the inspector, click **"Add Component"** and type **"Mesh Collider"** and press enter. The object is now buildable upon!

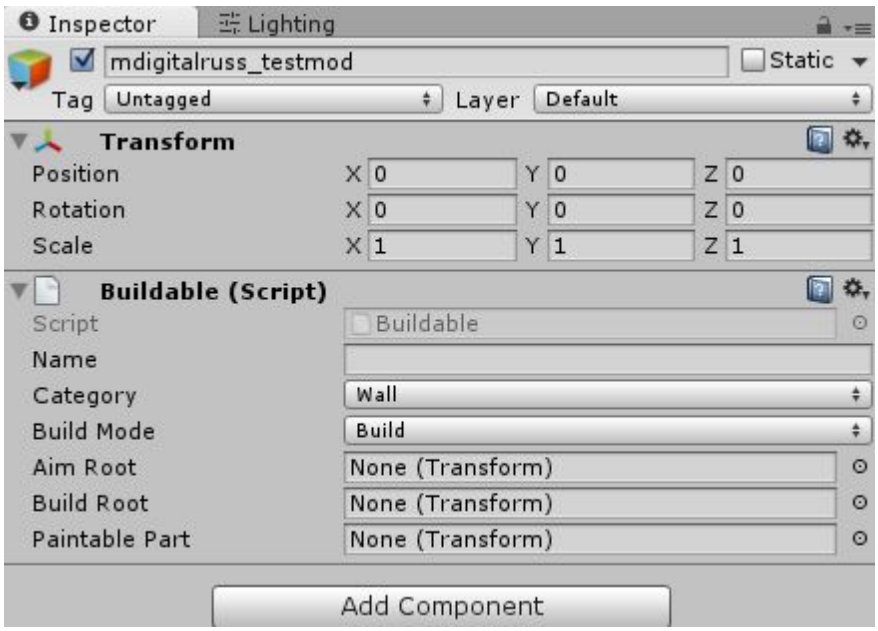


Tip: Ensure the Aim object never has a collider and is never set as Static!

Assign Script

Now we need to assign the script which tells Concrete And Steel to load this object, and tells it which is the Build and which is the Aim object.

On the root object (“*mdigitalruss_testmod”* in my case), in the Inspector, click Add Component.Type “**Buildable**” and press enter.



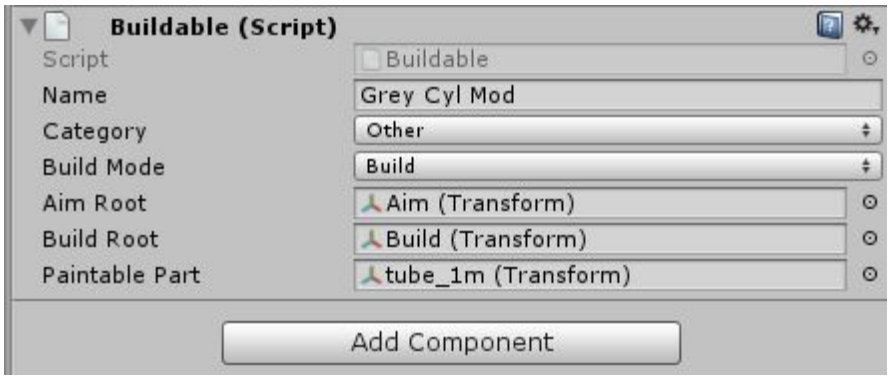
In the name field, type the name of your mod. I’m going to call this “Grey Cyl Mod”. This is what users will see in the build menu.

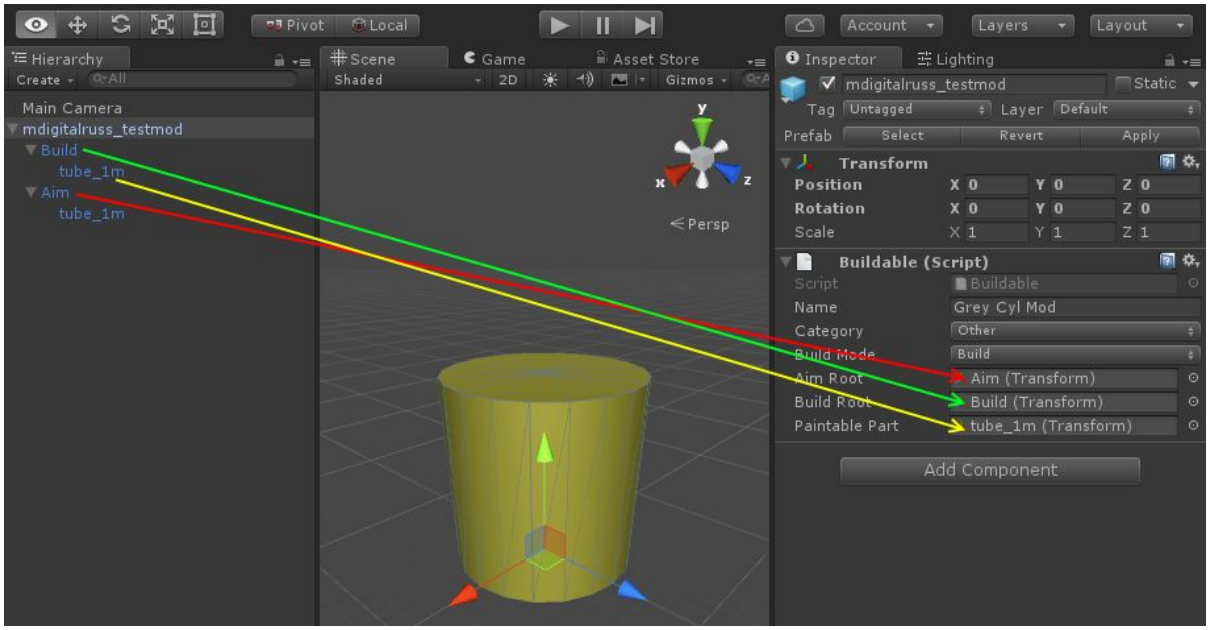
Under category, I’m just going to select Other. Pick the one most relevant to your mod. “Unlisted” means it won’t appear in the build menu, so it’s not wise to select that category.

For build mode you have two options, “Build” and “Place”. “Build” constrains the object to the 10mm grid and 90-degree rotations, whereas “Place” allows the object to be placed in any position and any location. It’s advisable to set structural components to “**Build**”, and props or decorations as “Place”.

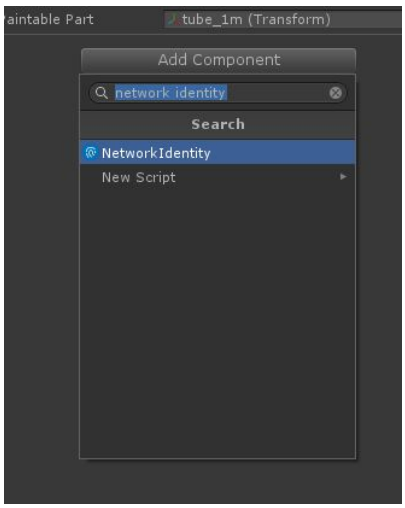
Now drag the “**Aim**” object (from under the root object) onto the “**Aim Root**” field. Repeat this for Build Root, dragging the “**Build**” object onto that.

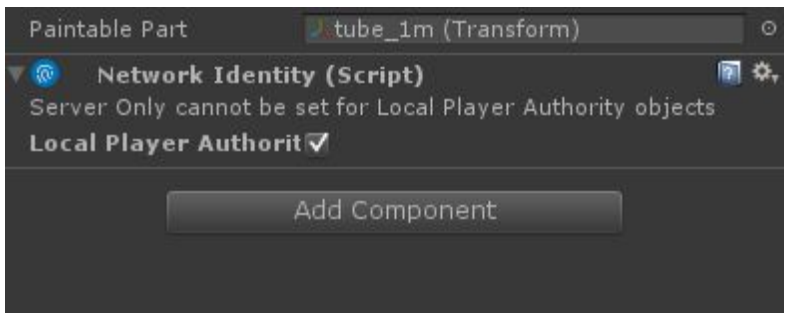
For paintable part, this is the object we’re going to see change colour when we paint it with the spraycan tool. This can only be one mesh currently. Drag “***tube_1m***” from under “**Build**” to the Paintable Part field. If you don’t want any part to be paintable, leave that field blank.





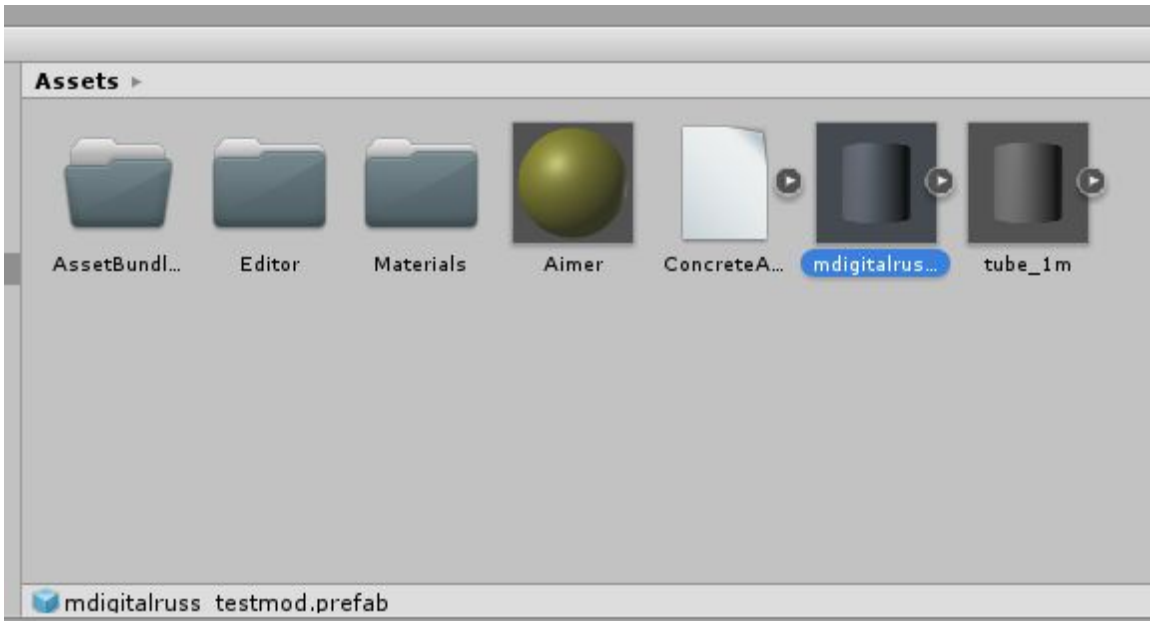
Finally, add a "**Network Identity**" component under the Buildable script, and tick "**Local Player Authority**". This is important for making the object paintable.



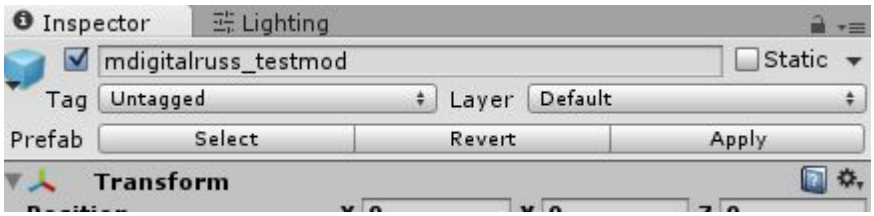


Create Prefab

Now we need to turn this object into a prefab. Drag your root object (in my case, mdigitalruss_testmod) to under “**Assets**” in the **Project** view. This will create another object called “*mdigitalruss_testmod.prefab*” or whatever you named your mod.



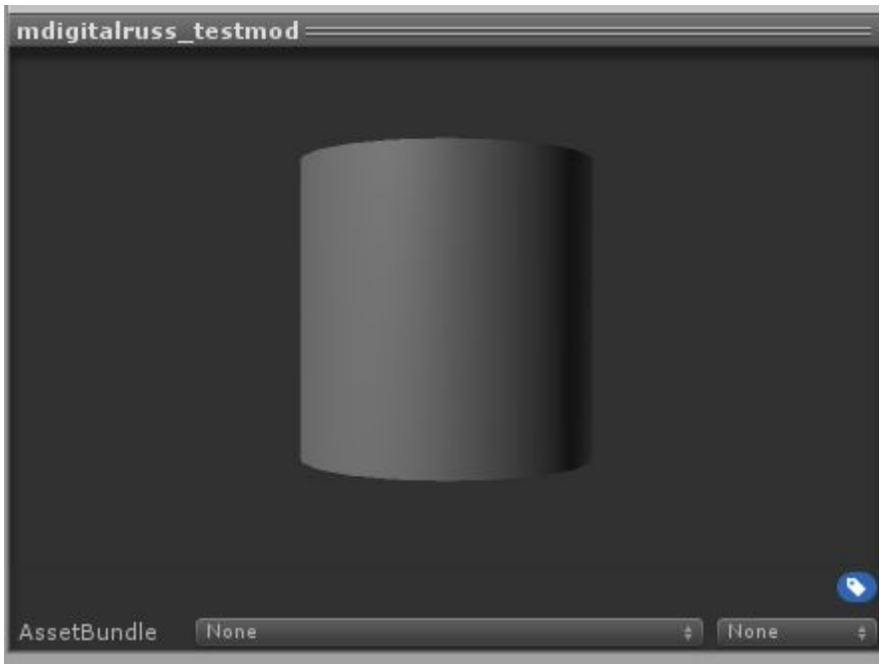
IMPORTANT: Once you’ve made the prefab, if you need to go back and make changes to the mod, remember to select the root object of your mod and click “**Apply”** to make the changes save to the Prefab.



If you don’t click “**Apply”**, your changes won’t get saved to the Prefab, and therefore won’t get exported.

Create Assetbundle

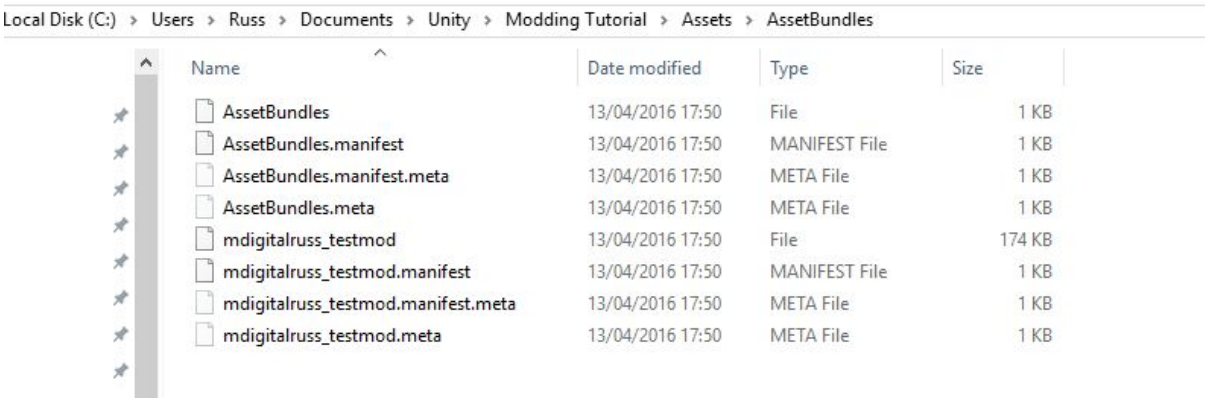
If you’re still alive by this point, and not bored to death, we need to finally set an assetbundle. In the bottom right (with the prefab we just made selected) we should see a preview:



Under “**AssetBundle**”, click “**New**...” and type the name of your mod (in my case, you guessed it, *mdigitalruss_testmod*) and **press enter** to make it so.

Export Assetbundle

Finally, in the Unity toolbar, select **“Assets > Build AssetBundles”.** You will get a progress bar, and then eventually you will get the mod files in your “**AssetBundles”** folder. On “**AssetBundles”**, right click and select “**Show in Explorer”,** and select the “**AssetBundles”** folder.



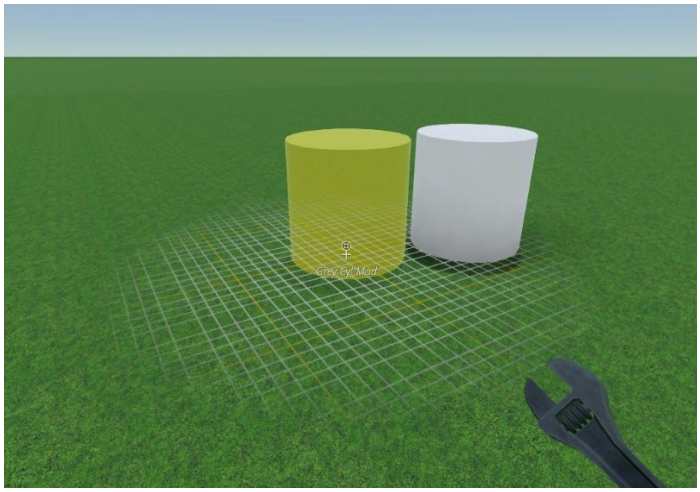
The only file we need from here is the file with the EXACT name as your mod. So in my case, ***mdigitalruss_testmod*** (174kb). You can ignore the other 1kb files.

Place in StreamingAssets Folder

Find your mod folder, it's normally here:

C:\Program Files (x86)\Steam\steamapps\common\Concrete And Steel\ConcreteAndSteel_Data\StreamingAssets

Copy and paste your mod file from the "**AssetBundles"** folder to the "**StreamingAssets"** folder. Now run the game. Your mod should be available in the game!



Further Consideration

- For multiplayer games, both the host and the client need the same mod in their StreamingAssets folder. This also applies to dedicated servers.

- To make changes to your mod, remember to click Apply to make the change save to the prefab. Then repeat the build process by clicking "**Assets > Build AssetBundles**" and copying the mod over to your "**StreamingAssets"** folder in the game's directory.

- If you make changes after the release which could break compatibility, consider adding a version number to the name of the mod - this means when you replace the mod with your new mod, players joining the game with the old version of the mod will be forced to seek the new mod.

- Please obey the EULA for Concrete And Steel when creating and distributing mods. Be aware that you are not, for example, permitted to sell or trade any mods using this SDK for profit or reward.

Authored by Russ Peterson. For help, information and bugs contact software@m.digital.

Changelog

0.1.0 27/04/2016 - Initial Alpha release to testers.
0.1.1 28/04/2016 - Added Network Identity / LPA information
