Where to Start?

Checked with version: 2017.3 - Difficulty: Intermediate

Before we start to explore rendering inside Unity, we first need to get our assets into a format suitable for our eventual intent. Setting up a proper workflow from your Digital Content Creation tool into Unity is critical. While we won't go through the export steps in the multitude of DCC tools available, there are a few things that we need to consider:

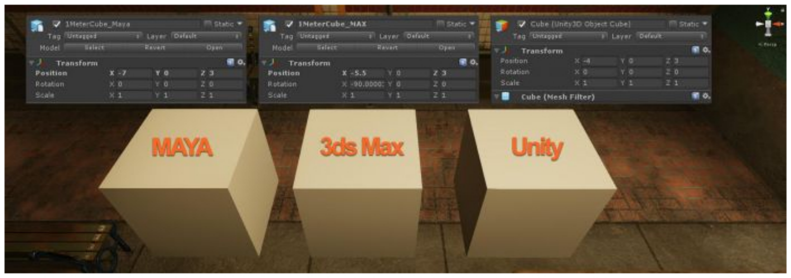
Scale and units

Scale and your unit of measurement play a very important role in a believable scene. In many "real world" setups, it is recommended to assume 1 Unity unit = 1 meter (100cm) as many physics systems assume this unit size.

Unit translation DCC 3D software to Unity

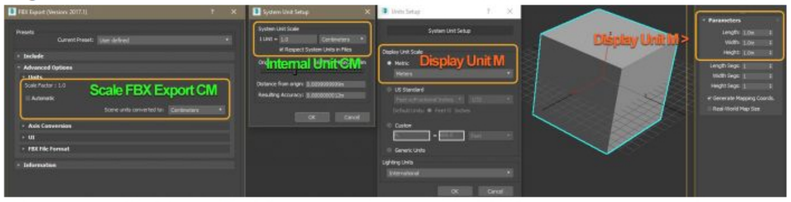
To maintain consistencies between DCC 3D software and Unity it's always a good idea to validate imported object scale and size. DCC 3D software such as 3ds Max/Maya/Blender/Houdini,etc have units settings and have scale settings in the FBX export configuration (consult each software manual to configure). In general setting these tools to work in cm and export FBX at automatic scale will match the scale in Unity import properly. However it's always reassuring to confirm it matches whenever starting a project.

A quick test to validate your export settings would be to create a simple 1x1x1m cube in your DCC 3D software. This can then be imported into Unity. The default Unity Cube (GameObject > 3D Object > Cube) is 1x1x1m and is therefore a useful scale reference to compare with your imported model.



These cubes should look identical when scale is set to 1,1,1 in the Transform component within the Unity Inspector.

NOTES: - Maya and 3DsMax can override your default unit depending on the last opened file. - Be aware that 3D Software can display different unit in the workspace while having different settings in their "internal unit". This might cause some confusion.



Shown example is from 3ds Max.

Point of reference scale model.

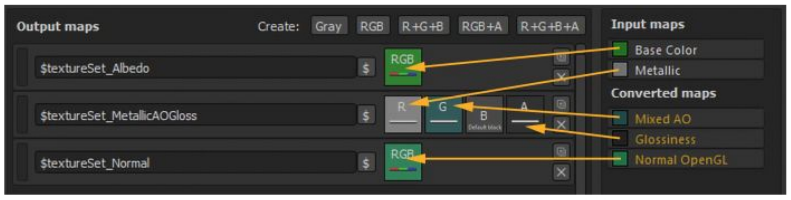
When blocking out a Scene with placeholders or sketching geometry, having a point of reference scale model can be helpful. Depending on what Scene you're making, choose a point of reference scale model that is relatable for the Scene. In the Spotlight Tunnel Sample Scene case, a common park bench was chosen. This doesn't mean the Scene has to be exactly the same proportions as real life, instead it allows consistencies of scale to be relative between objects even if the Scene is intended to have exaggerated proportions.

Creating Believable Visuals

Texture output and channels

In the same way that Unity expects consistent unit translation between 3D software and Unity, the information inside a texture needs to contain the correct information to give a proper result when added to a material.

Example of presets configuration for Substance Painter to output textures to be used with Unity Standard Opaque material.



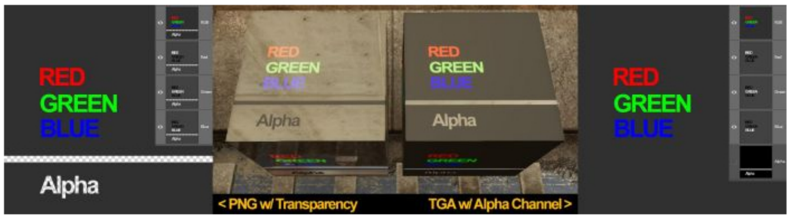
- **Texture assignment in Unity Standard Material:**

$textureSet_Albedo: assigned to Albedo Slot.
$textureSet_MetallicAOGloss: assigned to Metallic and Occlusion, smoothness Source set to Metallic Alpha.
$textureSet_Normal: assigned to Normal Map Slot.

NOTE: Packing multiple channels to a single texture such as the Metallic, AO, Gloss saves texture memory compared to exporting Ambient Occlusion (AO) as separate texture. This is the typical way of working with Unity Standard material.

Texture authoring software such as Photoshop and Substance Painter will output consistent and predictable textures with proper configuration. There are however some cases where content creators can get mixed up mainly dealing with alpha channel.

The example below shows "transparency" in a PNG can be confusing to author in Photoshop because of its native way in dealing with PNG Alpha without external plugin. In this case an uncompressed 32 Bits TGA with a dedicated Alpha Channel might be a better option, assuming source texture file size is not an issue.

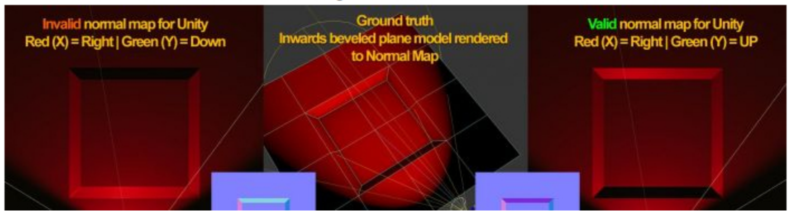


Photoshop authored “transparent” PNG above has alpha coming through as a black value, while the TGA with dedicated Alpha Channel shows expected value. When each texture assigned to Standard material that reads alpha channel as smoothness, the result is the unexpectedly inverted smoothness in the material with PNG textures while the material with TGA texture displays expected results as shown above.

Normal map direction.

Unity reads tangent space normal maps with the following interpretation: Red channel X+ as "Right", Green channel Y+ as “Up”(OpenGL style).

For example, 3ds Max “Render to Texture” normal map output Green Channel Y+ as “Down” by default. This will cause inverted surface direction along the Y axis and create invalid results when lit.


Invalid normal map for Unity
Red (X) = Right | Green (Y) = Down
Ground truth
Inwards beveled plane model rendered to Normal Map
Valid normal map for Unity
Red (X) = Right | Green (Y) = UP

To validate normal map direction, create a simple plane with concave bevel (middle picture on the example above) and bake it to a flat plane. Then assigned the baked normal map into a plane in Unity with identifiable light direction and see if any of the axis are inverted. Refer to your Digital Content Creation software manual for proper axis settings.

Tweet Like 3