s_tomanec@fanshaweonline.ca

Course: *1029 – Texturing 1*

Week 1 – Understanding UVs and working with UV Layouts in Photoshop

Texturing 1 will be focusing mainly on the process of unwrapping 3D objects into workable 2D layouts that can then be brought into Photoshop where we will generate and discuss the functions of Diffuse, Opacity and Emissive maps. We will also be touching on lighting and using the systems inside of 3D Studio Max to generate light or Ambient Occlusion maps for use inside of our texturing software.

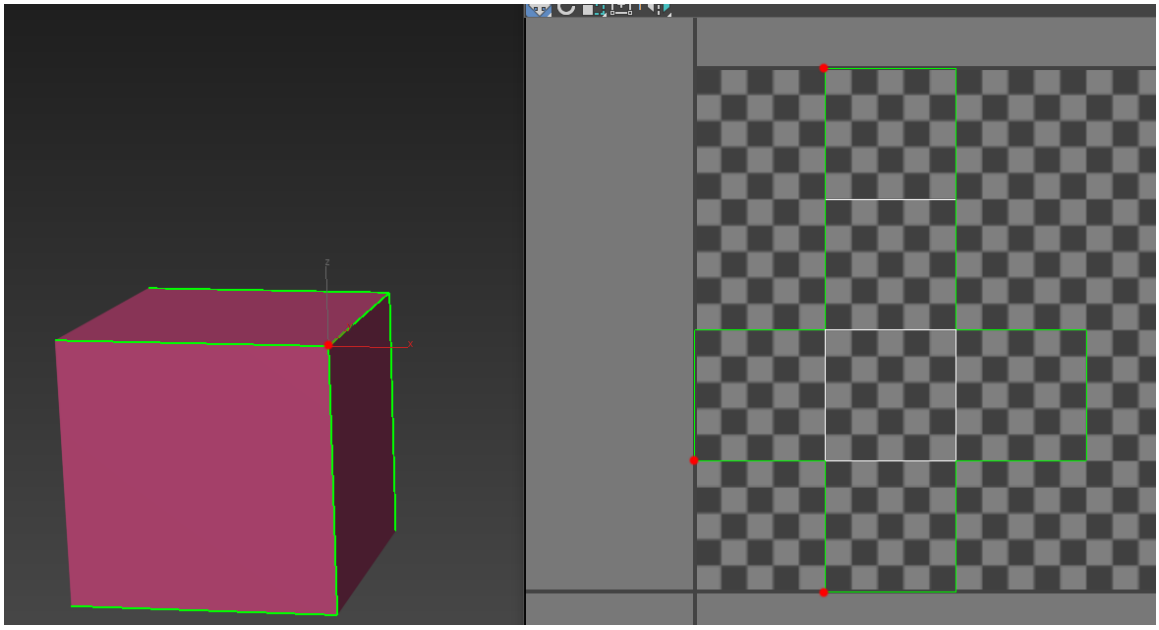
Emphasis will be placed on the process of UV unwrapping, because if the unwrap of an object is done poorly or without proper consideration given to the needs of the textures then no matter the amount of skill in actual texture generation, the object will make it impossible to work with.

What are UVs and how do they relate to 3D Geometry?

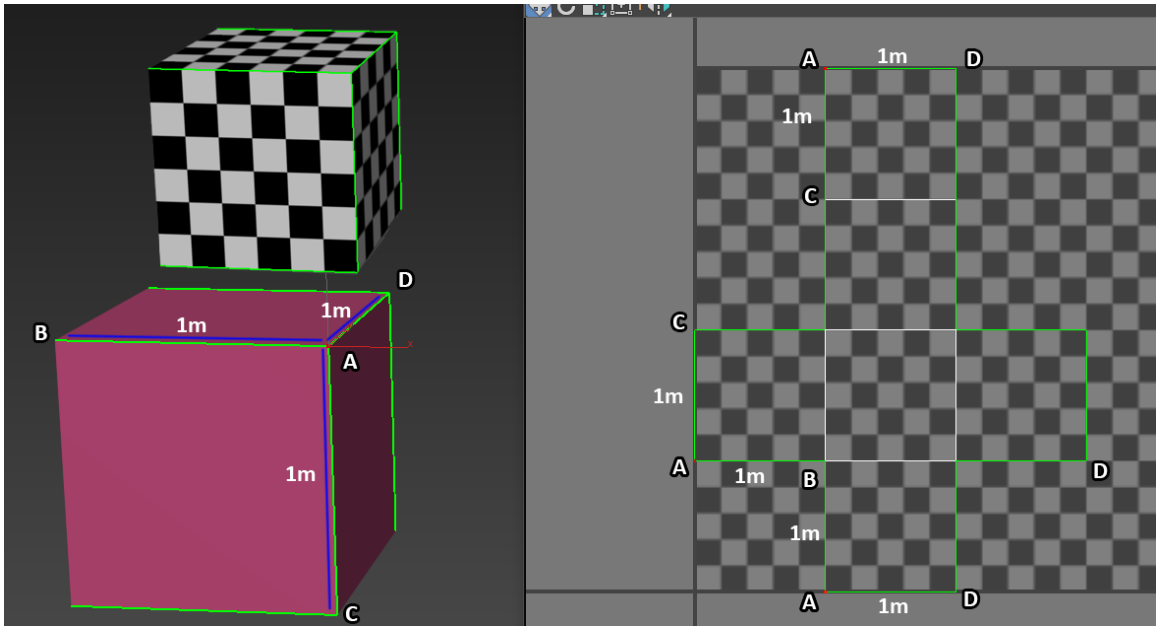
Even if you are utilizing 3D paint software (Bodypaint, 3D Coat, Zbrush, Mudbox, Substance Painter, Viewport Canvas) your actual textures themselves are still being stored out as 2D images. So how do you translate an object that has points in space that require three axis of information (X,Y,Z) into a 2D format? UVs.

Just as 3D objects have a Width, Depth and Height (XYZ), unwraps have a width and height represented by U and V. Why U and V? Because the directions of Up, Down, Left and Right on a UV sheet don't remain consistent, objects can and will be inverted compared to similar or adjacent structures so it wouldn't make sense to us the standard X,Y and it makes sense in the naming of the other three axis (UVwXYZ).

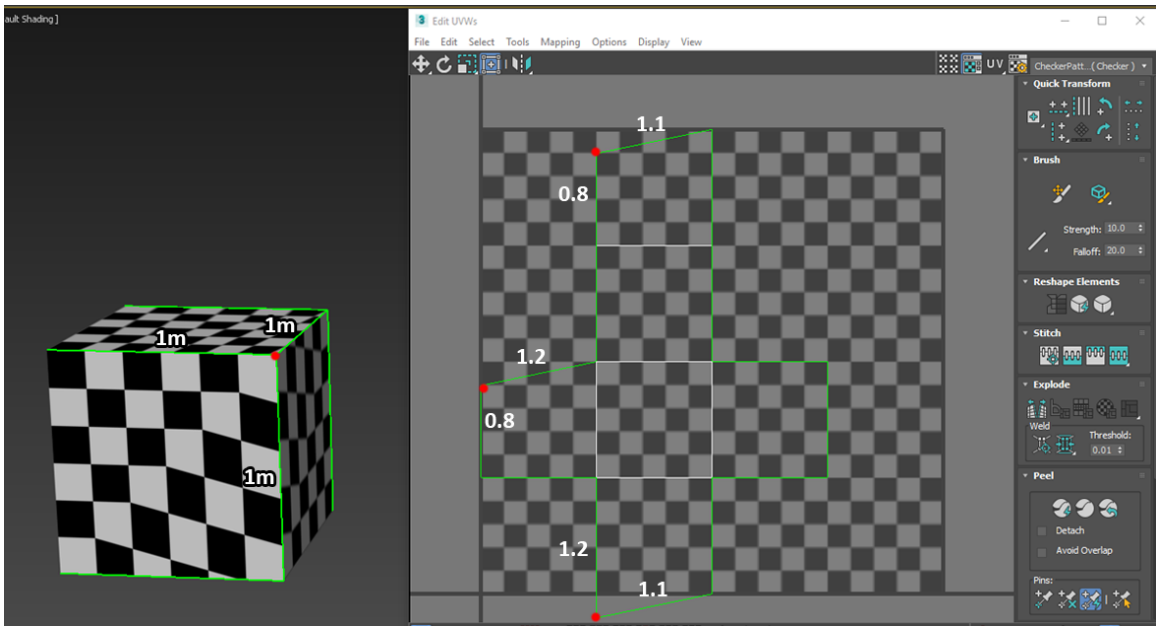
When unwrapping an object, the vertices occupying 3D space of the object are converted to UV points on a 2D plane, but since that third dimension of “volume” is lost, that’s where the trick to UV unwrapping comes in. Consider the following images:



The image above has a 1mx1mx1m box that has already been fully unwrapped. On the left you can see that a single vertex has been selected, and with the UV Editor window open you can see three UVs that have been selected at the same time. Because you have lost the third dimension of depth, each polygon of the box has been first separated and then re-stitched to keep the box as a single set of attached polygons, otherwise known as a “shell”. Because that vertex is shared by three polygons that are no longer adjacent to one another, it now has three points of reference in the UV window, allowing your to move each of them separately.



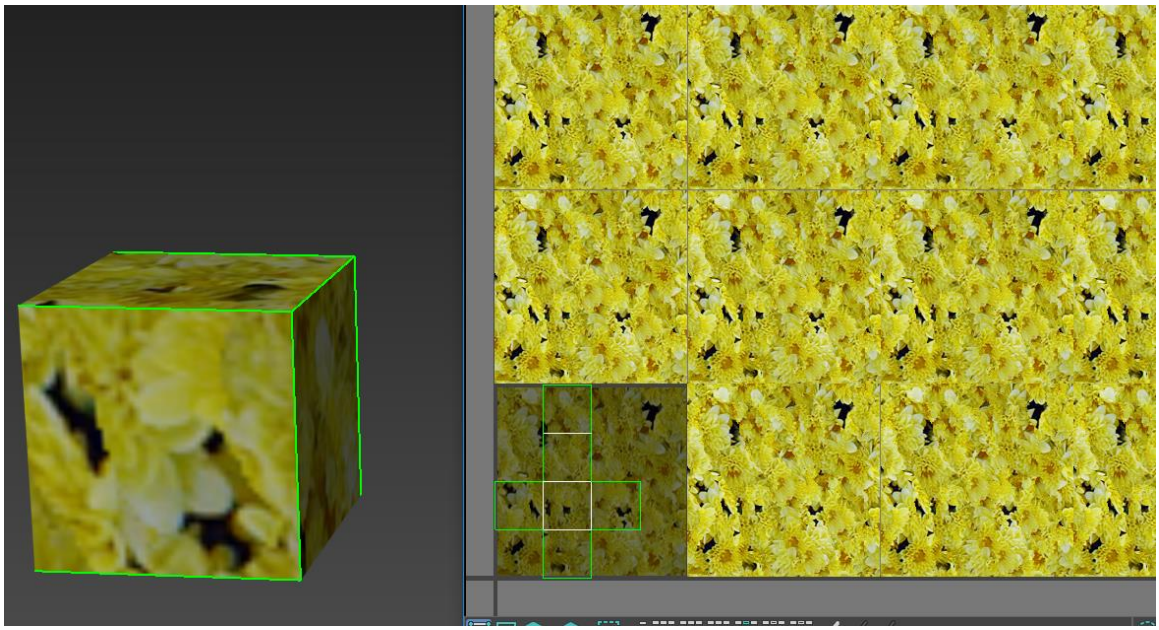
The distance between UVs needs to match the distance between their represented vertices in 3D. Since we won't be concerned with using a unit of measurement inside the UV window for now, I've left the metric units of measurement. You won't need to be tracking individual vertices and UVs as is displayed, they have just had letters assigned to show how many times a single vertex can be referenced in a UV unwrap. Above the original pink cube you can also see what happens when a simple checker pattern is applied to it. The expected result is that the grid lines appear perfectly straight, and all of the sections of the checker pattern are relatively the same size, because the relative distance between vertices is equal to the relative distance between UVs. What if that isn't the case?



In this image the three UVs that were named “vertex A/UV A” have all been shifted down slightly, we no longer have straight lines in our grid pattern and we no longer have consistently sized boxes because the distances between UVs no longer matches that of their 3D counterparts. This is typically referred to as UV skew, UV stretch/squash or simply UV distortion. Minimizing UV distortion is critically important as it can create issues when creating and applying textures which can be compounded when you start altering the mesh during animation. You will also notice that the “bottom-most” UV is no longer inside the grey grid in the UV Editor window, otherwise known as 1,1 UV space.

What is 1,1 UV space?

1,1 UV space is the basic range of values inside the UV editor that correspond to the texture that you have assigned the object. So, the bottom left of that grid will be the bottom left of your image, however values do exist going beyond that grid; 1,2 2,1 etc. Whatever texture you currently have applied will by default repeat, so 1,1 space displays the same image as 13,2 etc. This is particularly useful if you are using tileable textures



Shader/Material Vs Texture/Map - Types of texture maps

"Material" refers to the method being used to apply overall color, specularity or glossiness, transparency or translucency as well as detailed structures being provided by an image such as normal or height map to a 3D object. "Shader" is used more often in regard to game/rendering engines and refers to the combination as well as the behavior of those images and deals more directly with how the object is rendered, making an object double sided for example.

Think of materials as a container for all the aspects of an objects surface, such as its color, how opaque it is, and those will be further modified by your shader. You will get into this more in your Unity classes. The images that have been mentioned thus far are often referred to as "textures", or "texture maps" shortened to "maps", but they are referring to the same thing; an image that is controlling the surface properties of an object. Here are the four maps we will be looking at in Texture 1.

Diffuse Map – (RGB) Controls the objects color. Something to be aware of is that a diffuse map (or diffuse color map) will often have highlights and shadows painted in with the color, the shadows between boards and the highlights on raised areas on a wood texture for example. Be aware that these areas will get lighter or darker depending on the light

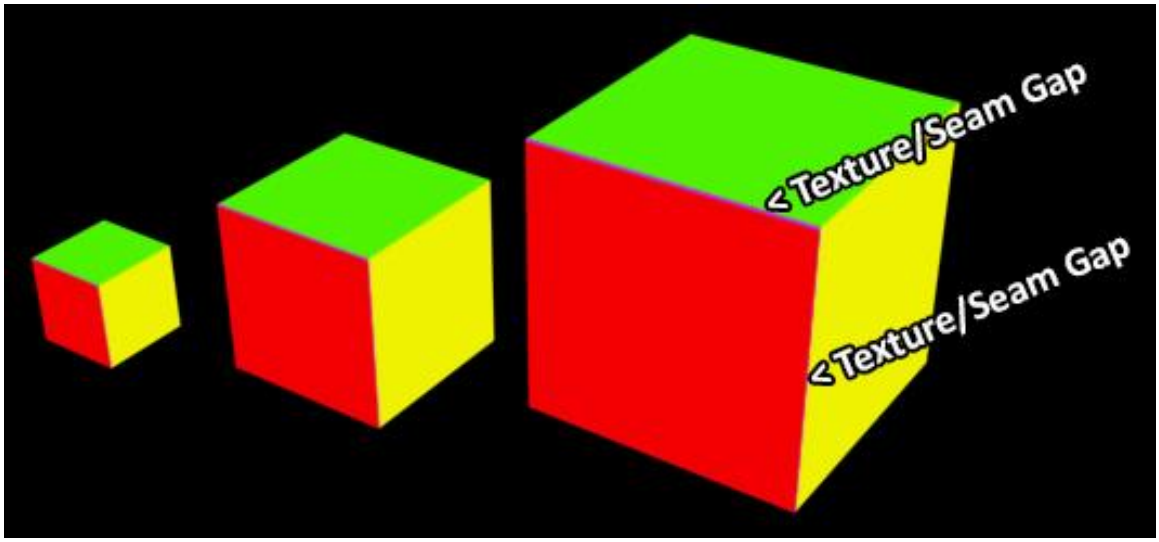
hitting them (assuming they are not painted pure white or black), but they will always be in shadow/highlighted regardless of the direction of nearby lights and depending on the overall style that you are going for may appear awkward.

Opacity Map – (Black and White) Controls whether parts of a mesh are visible. These are incredibly important, as they can be used to substitute areas that would otherwise require too much geometry to display properly, such as foliage and hair, in addition to the obvious use of having glass appear mostly transparent.

Ambient Occlusion Map – (Black and White) When two objects are close together, they begin to cast shadows or rather block light on one another due to their proximity to one another, you can observe this by bringing your open hands close together and assuming there is no light source directly in line with your hands you will see the shadow on each of them. This effect can be done in real time in a game engine, but to take some of the load off and get a higher level of detail you can also create an ambient occlusion map and make it a permanent part of your diffuse texture or material, making it appear much more believable.

Emissive/Self Illumination Map – (RGB) Controls whether part of the object produces light, from headlights to sci-fi engine exhaust to HUD components.

What is “bleed”/”padding”?

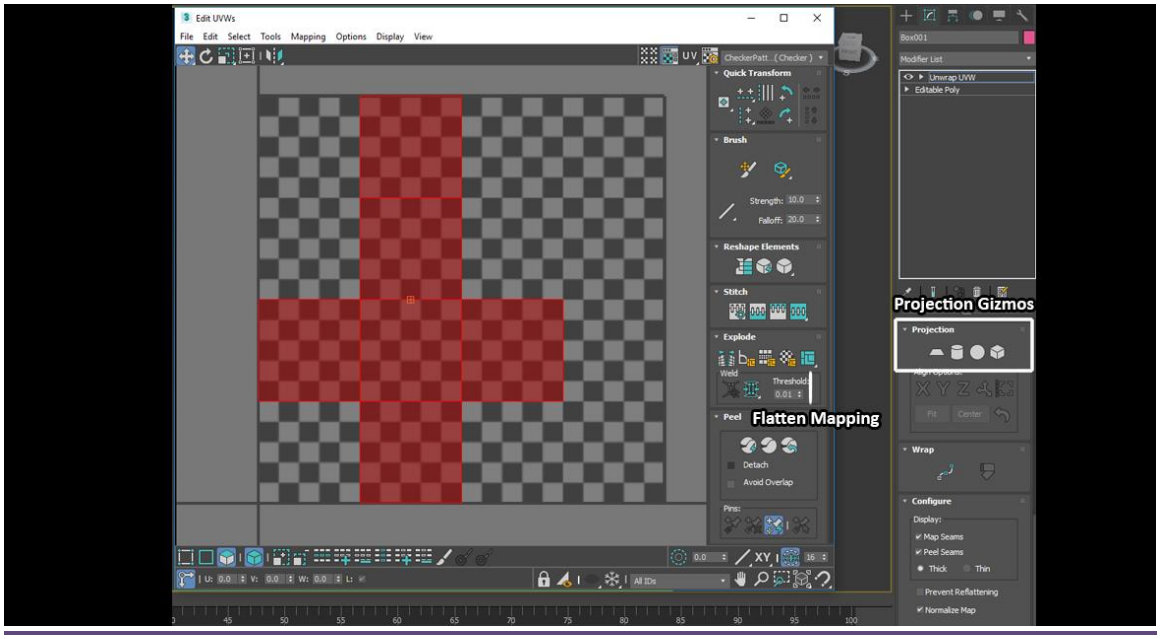


The purple lines showing on the top and side of the cube are examples of textures that end right on the border of the UV shell, which while not visible on the smaller cube becomes more and more evident the closer the camera gets.

Pixel bleed, also referred to as “padding” refers to map information that extends past the boundaries of your UV shell. This is especially important to consider with opacity maps as well as normal maps which you will get into later in the program. The key function of bleed/padding is to make sure that there is no issue displaying your textures at any distance to the camera, and that information is consistent right up to the border of the UV shell, as well as allowing you to resize the texture without too many issues. In regard to resizing textures:

If you are not sure, work one step “above” what you expect your final file size to be. So if you think you might need a 1024x1024, work from a 2048x2048. Resizing an image down causes far less issues then sizing a texture up.

Flatten Mapping vs Projection Mapping



There are two basic methods of unwrapping: Projection mapping and Flatten mapping, each with their own pros and cons. Projection mapping will create a gizmo in the viewport and based on its rotation and position, will attempt to create UVS that match the basic unwrap for that type of object, so using a cylinder projection will try to attach all of the polygons on one axis, spherical will try to keep everything attached as much as possible, and cube does a bit of a mix. The ones that you will use most often are cylindrical and planar unwraps. The point to using projections is that they can often make your workflow a little faster by doing some of the stitching and splitting for you. Flatten mapping on the other hand will simply look at your mesh and break it into UV shells based on the angle of intersection between faces, which by default is set to 45 degrees, so that any edge that has faces meeting at greater than a 45 degree angle is used as a border edge. It can be more time-consuming using flatten mapping, but as with most 3D processes, it often comes down to using the correct method at the correct time, which comes with practice.