Modular Concepts for Game and Virtual Reality Assets

A hot topic and trend of the games industry is modularity, or the process of organizing groups of assets into reusable, interlinkable modules to form larger structures and environments. The core idea is reusing as much work as possible in order to save memory, improve load times, and streamline production. There are however drawbacks to overcome with these methods. Creating variation in surfaces is important so that repetition of modules is not noticeable, and a viewer feels immersed in a *real* place.

One of the biggest issues with real-time environments is that we cannot do all the creation in-engine. As artists, we rely on a plethora of programs that all get consolidated into a workflow, with the final product reaching the engine. This is a challenge because it is important that the whole scene shares consistent and equal detail, rather than inconsistent pockets of micro detail where one spent more time. This requires a lot of rapid iteration and early testing. With the current next gen tools and engines, it is possible to add detail within the final scene by leveraging the use of advanced materials/shaders to increase the visual quality across the scene and break up repetition.

Pros >

Build large environments quickly

Memory efficient

Cons >

Extra planning time required at the start

Can look repetitive, boxy or mechanically aligned; boring

In this article I aim to share my experiences learned in creating environment art for virtual reality (VR) games that can be applied to any 3D application or experience.

How To Think as a Designer: Basic Fundamentals

Understanding how elements of architecture come together to form details and interesting spaces is just as important as knowing level design strategies of how to convey importance to a user when it comes to creating modular assets. Understanding how to simplify visuals into believable and reusable prefabs while working within design and hardware constraints is all a balancing act that gets easier with practice and study. In addition, this plays into how we as artists look at reference and decide which assets to make first. I highly recommend aiming for the minimum amount of work, or the maximum amount of reusability when beginning a scene. Iterative strategies such as this improve the overall quality without stretching budget, as well as inform next steps.

Games Versus Commercial

The biggest difference between games and commercial applications should be that the art created accommodates a player whereas, in commercial, our art accommodates a consumer. While a player is still a consumer, a game has rules and mechanics that must be emphasized in the layout and design of our spaces to accent what makes the game fun. Poor level design begets a poor user experience. Whereas with just a consumer, we must focus on an invitation to someplace comfortable and easy on the eyes. With VR, these lines are merging together regardless of the application you are making, whether it's a game or an experience. Games typically require more lateral thinking and communication between artists, designers, and programmers; it is important to get feedback early so as not to cause problems down the line.

As we build our interlocking assets, we must always check out the pieces in the engine, seeing how simple untextured or textured models line up or don't line up when trying to put pieces together to form simple structures. During these tests of assets the key things to check are:

Ease of use: Does the asset line up well with others? Is the pivot point of the mesh in the correct spot and snapped to the grid? Overall, is it not a hassle for you to use?

Repetition: Do we need new pieces to break up a kit that maybe is too small? If the viewer can easily see each piece and notice the prefab architecture we will have a hard time immersing people in a space.

Forms and shapes: Does everything have a unique silhouette and play with light well in the engine? Or does a flat surface break up well?

Composition: Is everything working together? Can an interesting image be composed?

Readability, color/detail, scale and the first two seconds: This is a later check after getting in some texture and base lighting for a scene. Make sure that pathing is visible or importance is conveyed right away. To get this right we may need more assets, or sometimes, less. This is all about visual balance and iterating quickly from rough geo and textures to final assets.

Learning how to think as a designer and understand how your work affects the whole process of development as well as the viewer experience is what ultimately allows you to make subjective decisions that can improve the design or limit cluttering of conveyance. Understanding why real-world places look and feel a certain way or how a shape suggests use of an object is the core of being an artist and a designer. I see a lot of technical breakdowns but understanding and studying these fundamentals through observing references will make you better, faster. Technology is just a tool. Modularity is a set of wrenches in the tool box. Building an airplane is easy if you know what you are doing.

Importance of Planning

Understanding the elements of design helps to make decisions when planning out our modular set and assessing its needs. As stated before, modularity requires a great deal of planning ahead of production. Reference should be the first place you start. Begin to break down images and pull out assets and ideas from real-world spaces. Good reference shows variety in a natural sense. Immersion is in the details.

The tools I use at this stage are Pinterest*, my ultimate tool when it comes to finding reference. Use it in tandem with a free tool called PureRef* to view the references and you have a very powerful combo. Trello* can help manage tasks based on images and show progress. These tools can help limit creative friction or indecisiveness, especially if screenshots are posted into Trello to keep track of what you did today so that tomorrow you know where to pick up. In time, this is a real

lifesaver for personal projects, to keep them going, as you see how far you've come.

When working with a client, get as much reference as possible; pictures, 360-degree videos, similar spaces, and any important details. In some cases, it may be good to do storyboards beforehand so as we work, we can think laterally as to the target goal, and the client is also on the same page. This could be as simple as photos of the place with a green screened character wearing a headset to imply what each scene is. Then move onto blockout or rough 3D layout, and then the final look pass. It is also important to consider the usage of sites like TurboSquid*, Cubebrush*, or the Unity Asset Store, that can help cut costs of production time on asset creation. Investing in Megascans* and Substance Source* can really help get in quality materials early. This also helps greatly since purchased assets oftentimes have lower quality textures.

General Process Overview

It is important to try different workflows and adopt other's workflows to see if one process speeds you up or slows you down. One thing I learned as an artist is that there is no one sacred way or secret sauce to being good. Everyone has a different process or tricks depending on how they work, what they are working on, and where they work. Here is a basic process you can use to see the general overview of what goes in to a complete personal environment. You have to be flexible and iterative. Know when something is good enough and minimize creative friction by keeping tasks and files organized. Figure out a good naming convention for your assets early.

- Gather reference: Learn your space. Map it out in your mind. What does it feel like?
- Sort reference into tasks: Assets, materials, modular components.
- Check 1: Attempt a basic 3D layout or blockout of primitive geometry. Sense of scale, space, and interconnectivity of assets is important.
- Test early lighting: Adjust elements to suit ideal composition.
- Once your meshes work well together and the scene is composed, improve the detail of the meshes to a final in game and begin unwrapping.
- Check 2: Apply textures early, focusing on basic albedo, basic normal maps, and the overall readability with temp lighting. Note if your albedo is too dark.
- Begin to follow your production schedule and continue to work up the detail. High poly creation, final texture passes; focus on larger or frequently used assets first as a baseline to always check if everything is cohesive.
- Create supporting props to fill the scene with desired detail giving a sense of character, story, and scale.

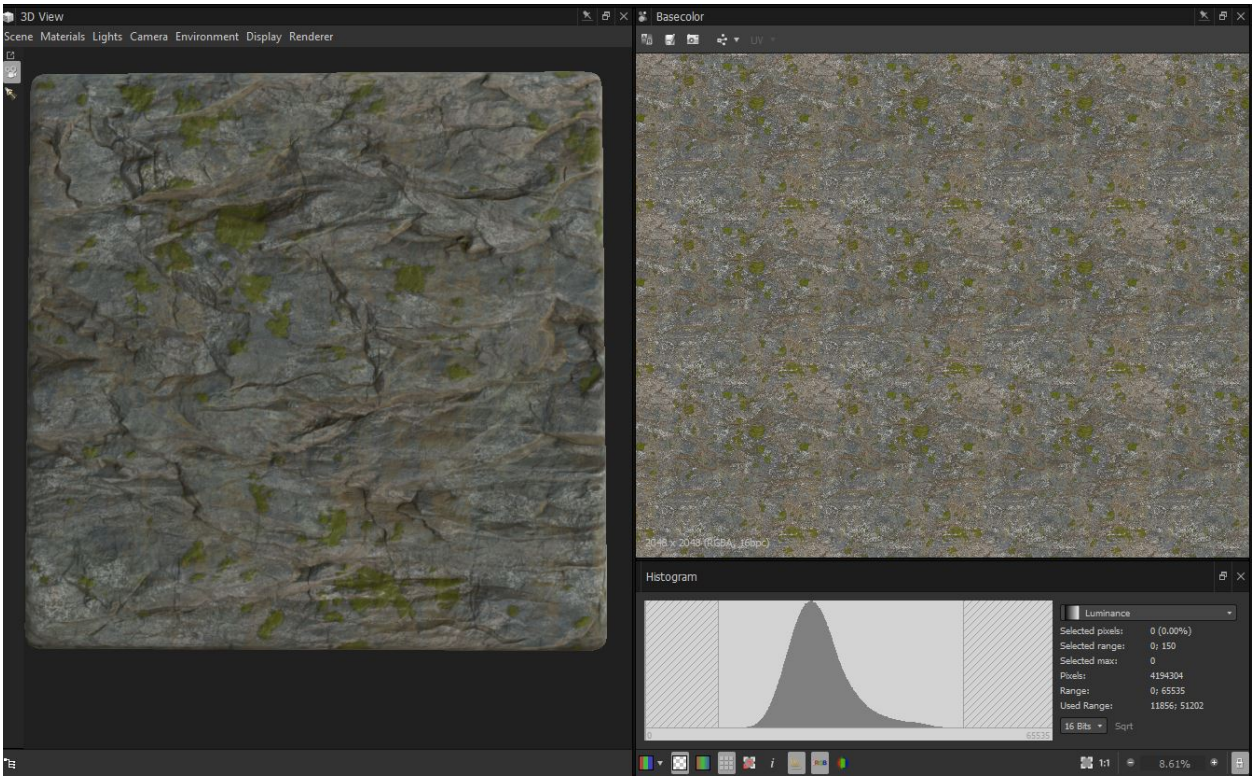
- Final lighting pass: Take screenshots, compare changes in external photo viewer, tweak until desired look is achieved.

  The greatest advantage in my opinion of working modularly is that you are ever aware of how many assets you have to be consistently updating. In a professional setting this helps a lot if changes need to be made, either in layout or optimization.

Texture Types

As we begin to look at the reference we want to capture the atmosphere, space, and harmony. However, it’s also crucial to be able to analyze each bit of reference and see where every material is used, so as to notice where the same materials are applied. The goal should be as few materials as possible to limit the work into a manageable scope and reduce the overall loading of textures, which take up a great amount of memory at runtime. To grasp modular concepts, it's necessary to understand the three types of textures we will be working with.

Tileable: Textures that tile infinitely in either all four directions or only tiles horizontally or vertically are also known as trim sheets. Tileable textures are your frontline infantry and should make up most of your scene as they are efficient and reusable. Oftentimes, we can start here and model directly from these textures to get our first building blocks. We will cover that further along.



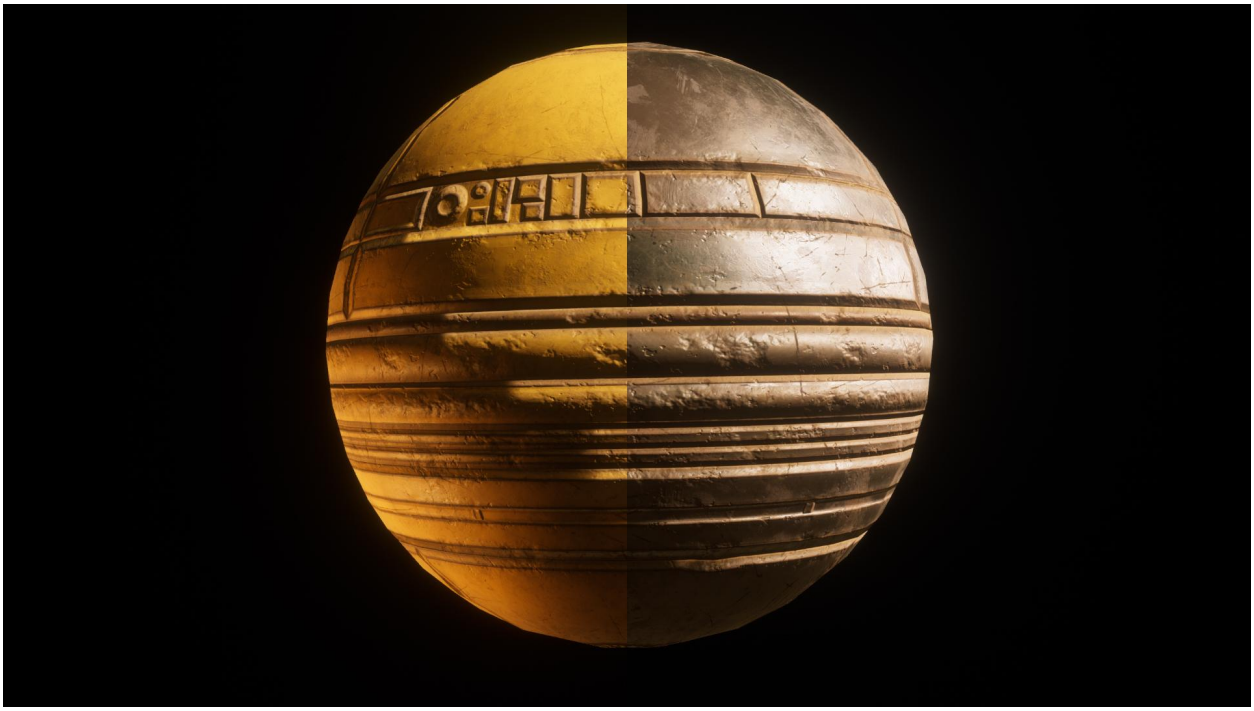
Tileable texture

Tile notes and tips:

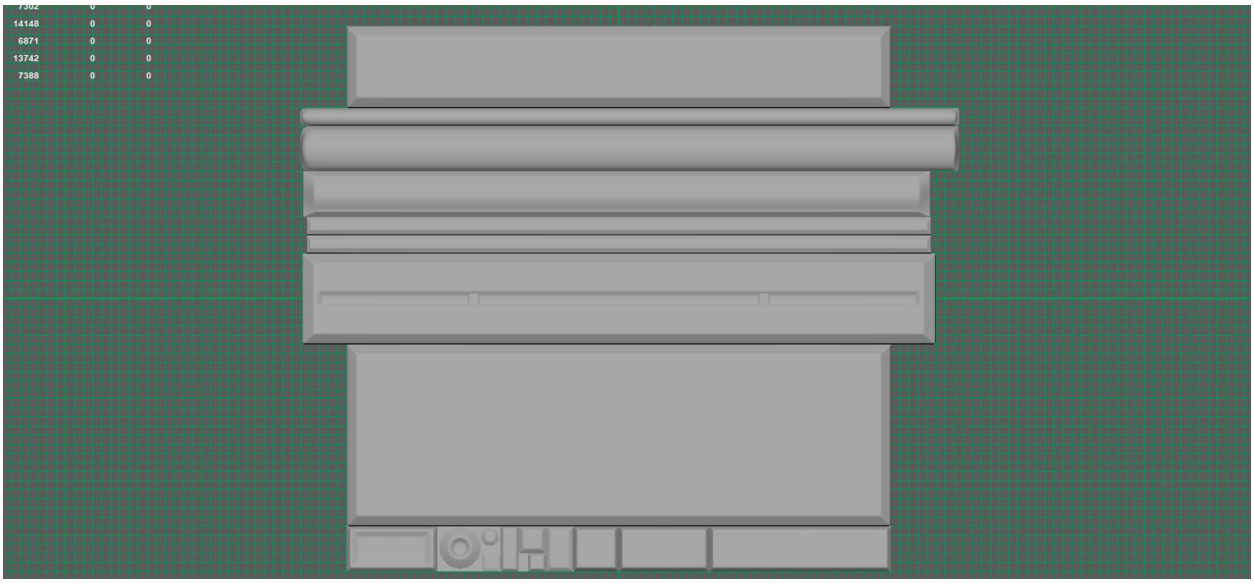
- Substance Designer* is incredible for making just about any tile texture or material imaginable. It excels at achieving hyperrealism without using scan data, and is completely nondestructive. Its node-based system allows for the ability to open parameters for use in engine, leverage random seeds for quick iterations, limitless customization to tweak on the fly, and happy accidents.
- Priority number one for physically based rendering materials is the height and normal, then roughness, then albedo. This way, the initial read feels correct and the details are lining up, but we can also use/create Active Objects(AO), curvature, and normal map color channels to create masks from the height data in order to create the roughness and albedo.
- Decide how the texture should tile and be aware of repetition! My rock texture is noticeably tiling, but this study example was to create a more unique piece that would blend between a less noticeably repetitive version of the same rock material. This is called having low noise; supplying areas of rest, but details that invite the viewer in.

- Use the histogram! To make an albedo work in any lighting condition and look natural, your albedo should have a nice cure in the histogram with the middle landing around a mean luminance of 128. This number can be lower, as albedo information for materials changes for naturally darker surfaces, and vice versa. Furthermore, getting a wide range of values for the color or a softer curve is more natural. Check the value curves on textures at Textures.com* to help give yourself proper value and color. It is worth mentioning that mid gray is a value of 186 RGB. I would further recommend using this value on all assets in a scene when tweaking early lighting tests to see how geometry/height maps add in light versus shadow details.
- Albedo and roughness are two sides of the same coin. Both interpret light to create value. Albedo is just the base coat, but if you look at a wall that has good specular breakup, you notice that areas with more reflectivity are brighter and inherit light color, while duller areas are darker and maintain more of the raw albedo color.
- Learn how to create generators in Substance Designer to speed up your workflow, and reuse graphs to give yourself a good base. These can be shapes and patterns, edge damage, cracks, moisture damage, and so on. Please, accredit generators and graphs supplied by other artists that you may use in a Frankenstein graph.

Trim sheets:



Using Substance Smart materials to create to interchangeable variants.

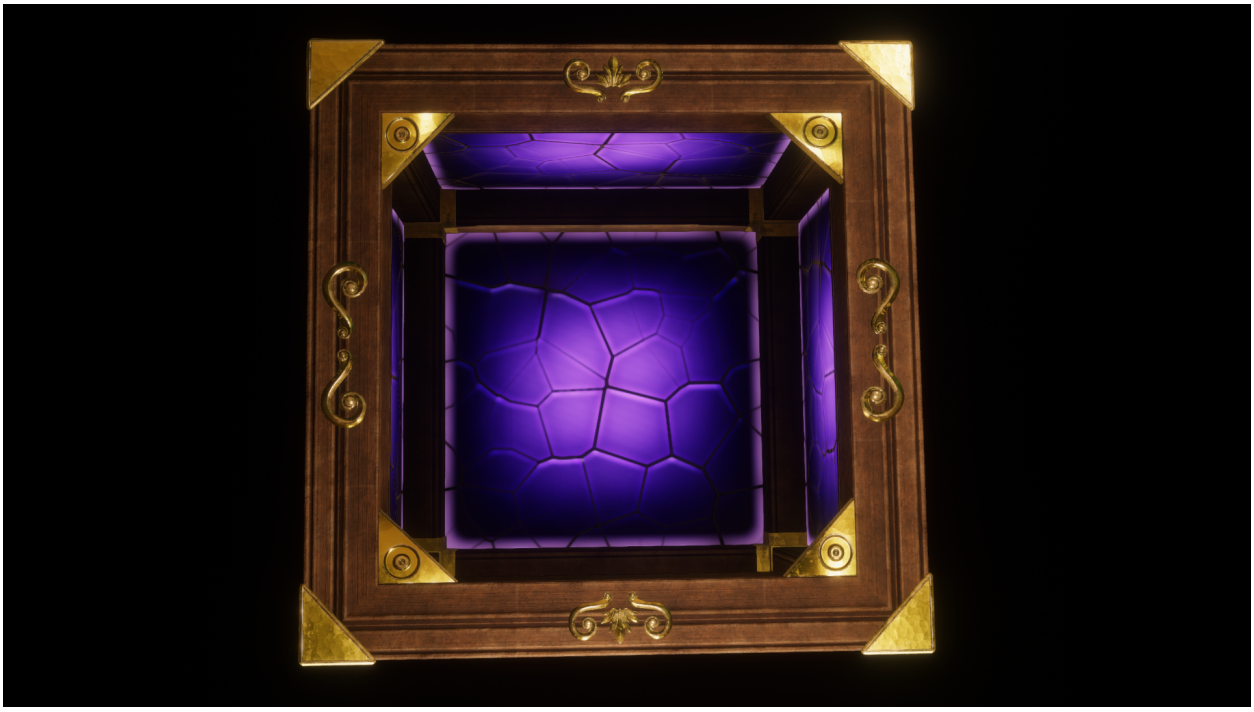


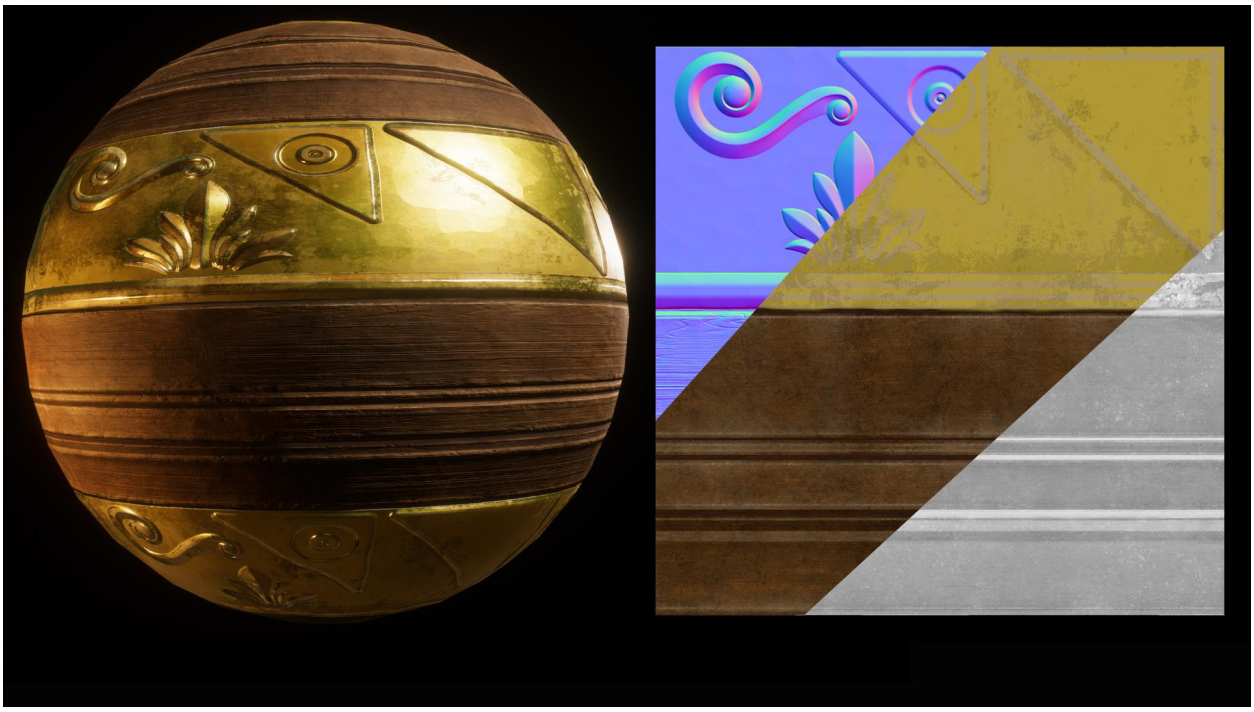
Our trim sheet high polygon mesh:

Trim notes and tips:

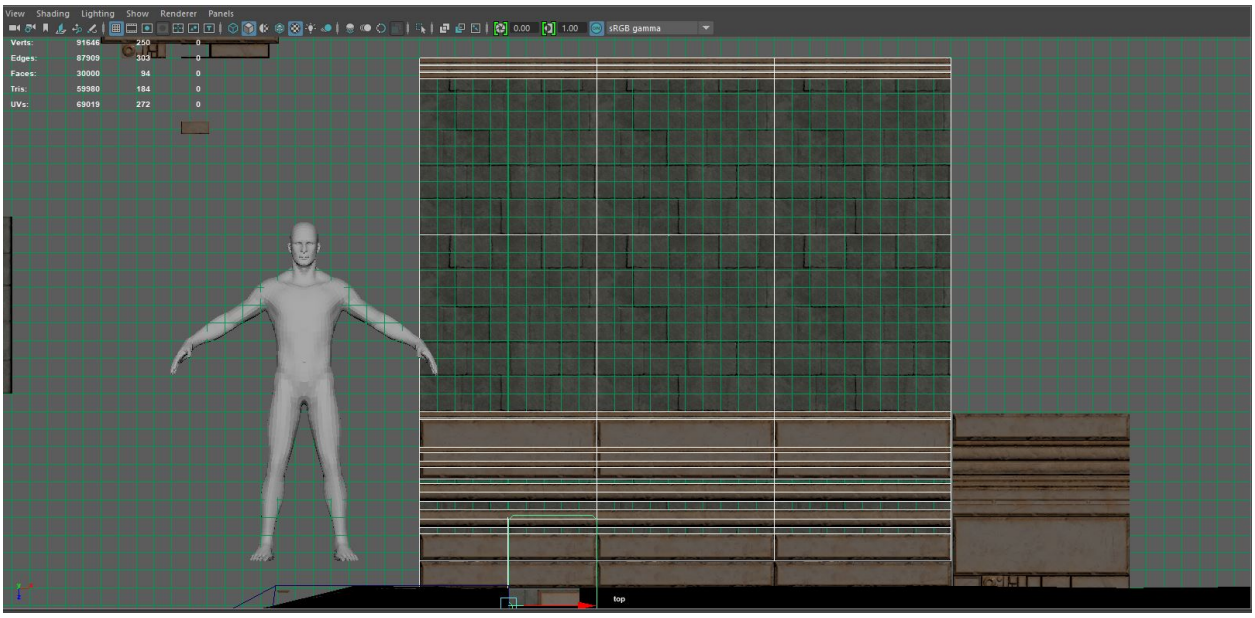
- Create a 1m by 1m plane as your template outline. Keep this to export as your low poly mesh as well as export with high poly to ensure no gaps between elements.
- Snap elements to grid! Makes for conversion to scene scale relative and accurate for texel density!
- In order for trims to tile in the bake, the geometry has to go past the 1 x 1 plane.
- Don’t unwrap! This gets baked to a flat plane so we do not have to worry about geometry or unwrap these objects. The base plane is already unwrapped to 0-1, so we are good to go.
- Floaters. We can make 3D objects to use as details that sit or float on top of other elements such as the small bolts in the bottom corner. Because the texture is baked flat, the bake won’t recognize the change in depth, as the normals should appear to blend seamlessly. This saves a lot of time as we don't need to model details into complicated meshes, but rather reuse these elements like screws, bolts, or any concave/convex shape to place where we want. This also makes it nondestructive if we are not happy with the bake result.
- Save space for small, reusable details. Sometimes floaters don’t turn out too well if they are small and there are not enough pixels to support the normal. By putting details at the bottom, we can reuse these on the game mesh by putting small planes as floaters that have the details mapped to them.
- Forty-five-degree bevels in the normal map help to smooth edges, and at this angle they are very reusable. *Sunset Overdrive** has a great breakdown of this technique. This isn't to say that each edge needs this, however; it is mostly for lower poly objects with hard edges.
- Simplicity is often better. Have some unique elements, but low detailed pieces are far more reusable and less noticeable than, say, a trim with lots of floaters, so find a good balance.
- Your trim can be composed of different materials such as metal trims, wood mouldings, or rubber strips all on one texture sheet. Baking out a material ID mask helps to make a sheet more versatile for saving memory. This is where really good planning helps.
- It is also possible to create trims without geo in Substance Designer and Quixel NDO*. Details can also be added to trim geo using alpha masks in Substance Painter*.

Further creative examples:

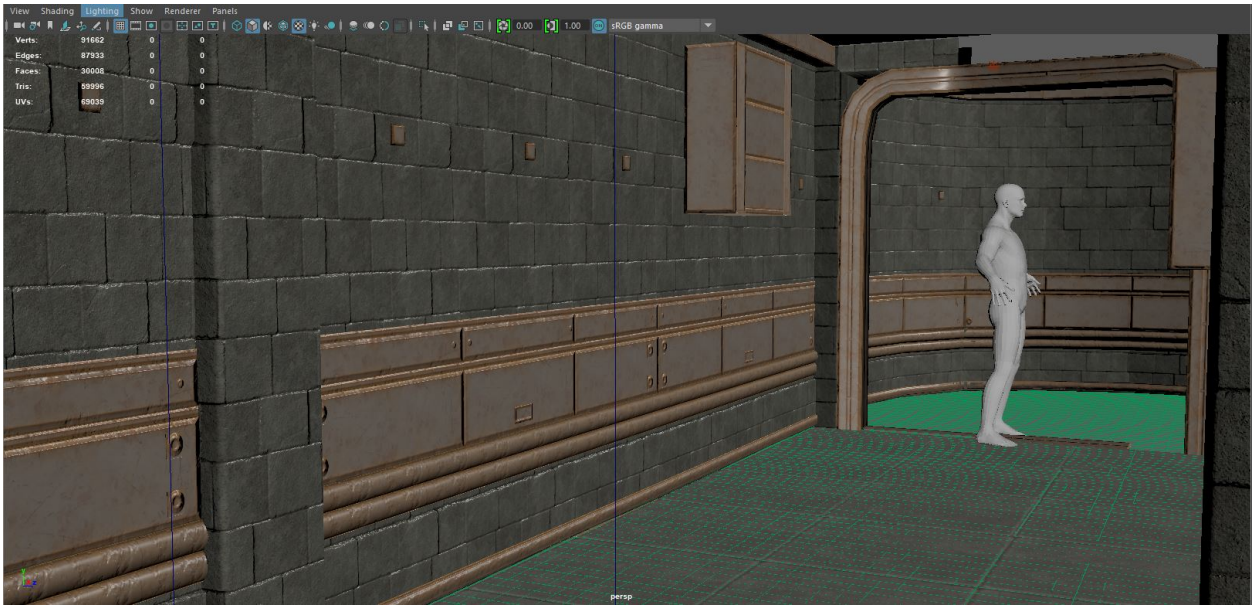




This was for a mobile AR project. This method kept the texture size low and the detail high.



If trim sheets are made to the grid, they can be easily used interchangeably by adding edge loops to a plane and breaking off the components. This method is quick for prototyping since the UVs will be one to one to ensure tiling. If planned accordingly, one can also interchange textures if the textures share the same grid space. Check out this breakdown by Jacob Norris for details.



Here, the elements come together to form a basic corridor using one trim and one tileable.

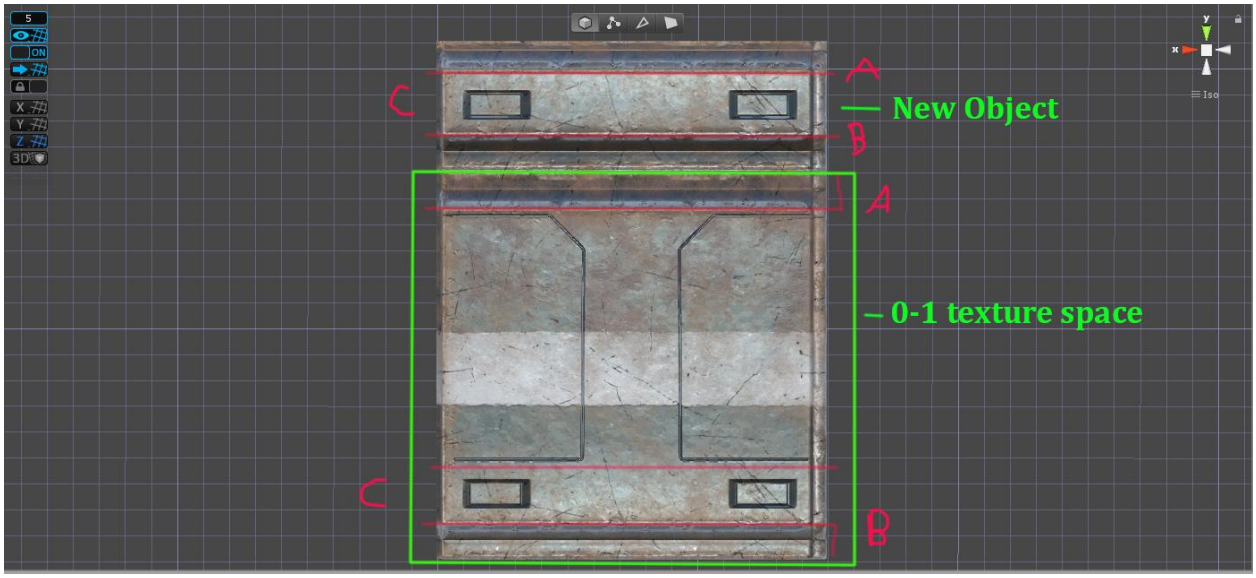
Unique: A texture that focuses on one asset like a prop. It utilizes baked maps from a high polygon model to derive unique, non-tiling detail. These should

be the finishing touches on a scene, and ideally, similar props always featured together are on a single texture atlas.

An exception to doing a unique asset first would be with a hero prop, which is a category for a unique asset of great significance to the overall scene. In this case getting a basic or final unique asset can come first.

Chess set uses one texture for all pieces. Each piece is baked from a high poly to a low poly and each surface face has unique detail.

Hybrid: Features the use of both tiling elements and unique elements on the same 0-1 texture. This could also be material blending on larger assets that use unique normals and masks to pump in tiling materials such as ornate stone walls or large rocks.

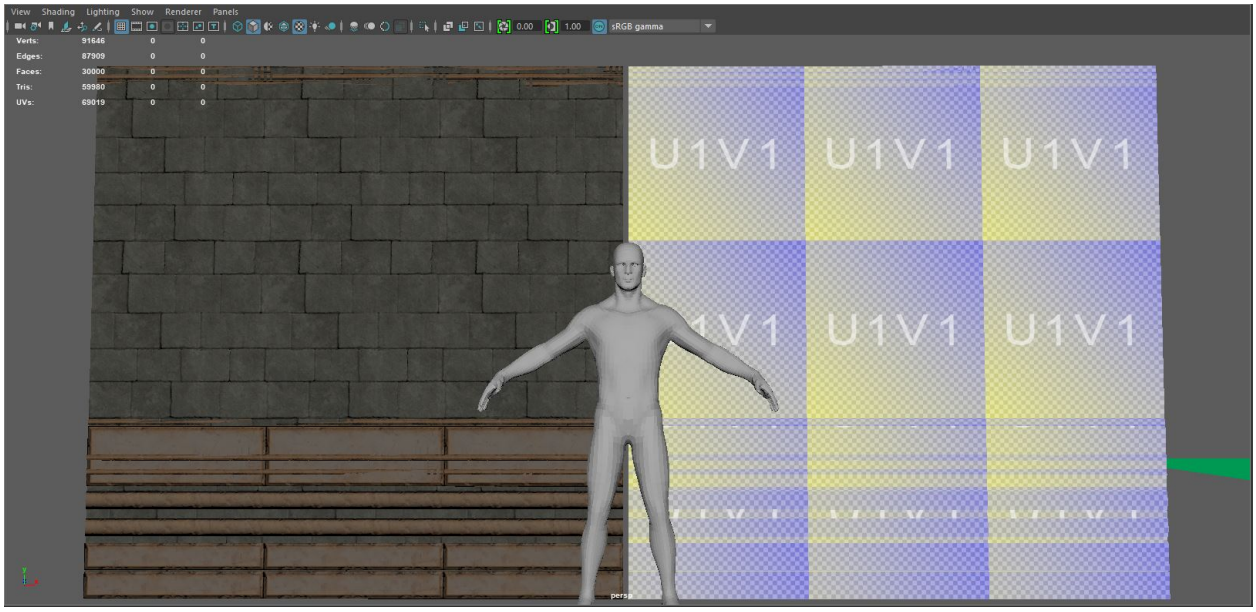


This hybrid example uses a unique normal map to drive the wear generators in Substance Painter to get masks that blend the rust and metals together. The red lines also indicate where I made cuts to form new geometry from the existing mesh. Yay, recycling!

The elements coming together to make an old sci fi warehouse kit. One trim (two color variants), one hybrid, and two tile textures.

Texel Density and Variation

One last thing to consider at an early stage is texel density or how many pixels per unit our assets will use. This can be pretty lengthy for first timers, and I highly recommend this article by Leonardo Lezzi. For first-person experiences we would want 1k or a 1024 x 1024 texture per one meter, or a texel density of 10.24 pixels per cm. We primarily want to use tileable textures in order to maximize visual detail on larger surfaces. Exception to these rules would be anything interactable that will be close to the player, such as a gun. I like to use Nightshade UV for Maya* when unwrapping. The tool has a built in texel density feature to set or track the pixels per unit. Below is an example of a 3m x 3m wall with a texel density of 1k per meter.



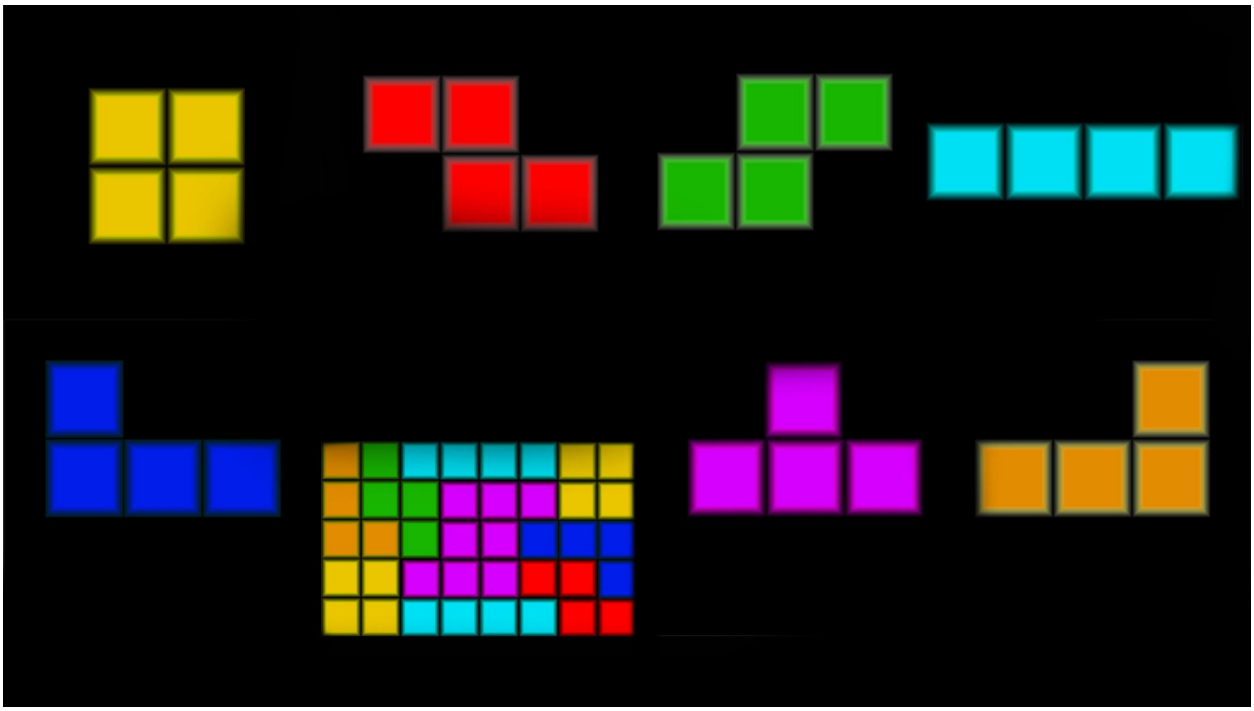
VR needs at least 1K per meter, but as the headsets evolve, that value will likely increase to 2K per meter. This poses a lot of challenges for VR artists as the hardware of computers will still limit texture sizes for many users. VR headsets already render two images at a wider field of view, making higher frame rates and nice details tricky. In order to combat this, it is possible to create shaders in Unreal* Engine 4 (UE4) and Unity* that use lower resolution detail maps to create the illusion of nearly 8K. A short demonstration of this technique can be seen on Youtube: Detail Textures: Quick Tutorial for UE4 & Megascans.

These shader setups are also critical to adding variation and breakup to surface tiling, either by using mask-based material blending or using vertex colors

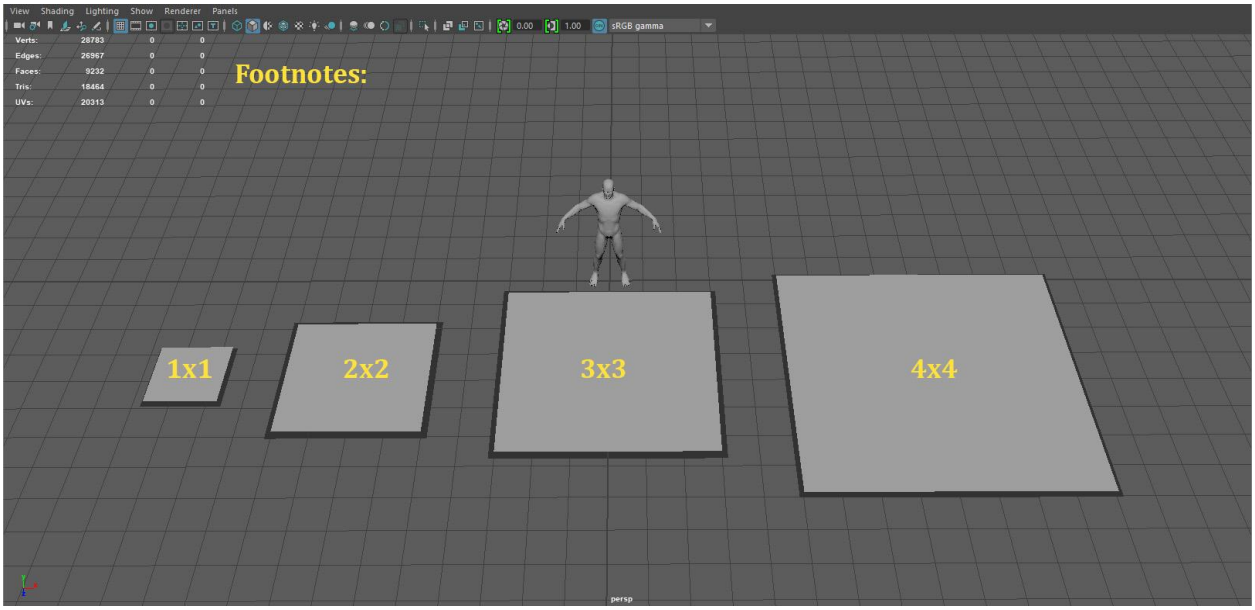
to blend materials or colors. This topic is bit of a rabbit hole as there are so many different ways to achieve variation through shader techniques. Amplify Shader Editor* for Unity is a great tool to allow for this style of AAA art development. Additionally, this great breakdown by senior environment artist Yannick Gombart demystifies some of these techniques.

The Grid

I remember that, at first, modularity was a difficult concept to grasp. Take a look at Tetris*. All Tetris is is a stacking game of *pieces* or *modules,* which are paired together to create new, interlocking shapes that all fall onto a **grid**.



The grid is the guide to our modular blueprints according to a set footnote or scale. The footnote informs the basis of our modular construction and usually depends on the type of game we are making. If we were in Dungeons & Dragons*, is the base tile size for our modular corridors 5 feet or 10 feet? How does that impact player movement and sense of space? In Tetris, it would be the base cube size.



Since we are discussing VR, we are talking about first-person games and applications; a good footnote would be three meters by three meters or four by four. Always work in centimeters when modeling, so that would be 300 by 300. The metric system is easily divisible, making modules easy to break up into round units, and it is what game engines use by default. When deciding a footnote, keep in mind that first-person VR tends to make objects appear smaller than they actually are, so exaggeration in shapes makes things feel more realistic or clear. To begin, we need to change our modeling application's grid to mimic our engine's grid so that they will integrate seamlessly.

How To Set Up our Scale in Maya*

First, let's ensure we are using centimeters in Maya.

Go to **Window > Setting/Preferences > Preferences**

Click on **Settings** and under **Working Units** check that Linear is set to Centimeter.

Now, let's set up our grid.

Go to **Display > Grid Options**

Under Size set the following values.

- **Length/Width**: 1,000 units. This is the overall grid size of our perspective view and not the size of each grid unit.
- **Grid lines every**: 10 units (this controls grid unit lines as it does in Unity or Unreal Engine 4 (UE4); so, if you can set this value to 5, 10, 50, 100 it will match the UE4 grid snaps. For Unity, I use a ProGrids* plugin that mirrors a more robust grid, like that of Unreal). Changing this value is what will mirror how assets snap and line up with each other in the engine.
- **Subdivisions**: 1. This changes how many grid lines we have per unit. At 1 it will be every 10 units, but at 2, we change this to grid lines every 5 units. It's a quicker way to divide the grid into different snapping values, by sliding the input, rather than inputting a unique number each time for the **Grid lines every** field.

  Finally, create a cube 100 x 100 x 100 and export it as an .FBX file, in order to see if your asset matches the default cube size in the engine (usually 1m by 1m).

- **Useful hotkeys:**
  - By holding **X**, you can snap an object to the grid as you translate.
  - Holding **V** snaps the selection to the vertices.
  - Pressing the **Ins** or the **D** key allows for moving an object's pivot point. In tandem with holding **X** or **V**, you can get the pivot on the grid.
  - Holding **J** snaps the rotation.

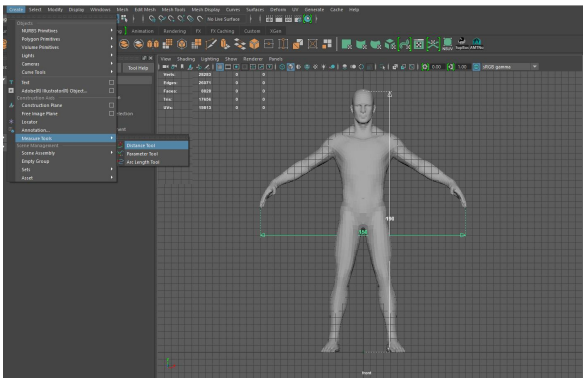
  Again, with whatever modeling software you use, the goal is for the grid to match that of the engine's grid.

  Here is a great walkthrough with images for grid setup, as well as a good site in general for good level design and environment art practices.

Bringing It All Together—The Blockout

Now that we are aware of our metrics and know what textures we need to look out for, we can go one of two ways. We can either make a quick texture set, usually a trim sheet and a tiling texture to create modules from, or we can go straight into a modeling program. Either way, breaking down the reference into working units and materials reduces creative friction early by having a good understanding of the space and sense of production scope.

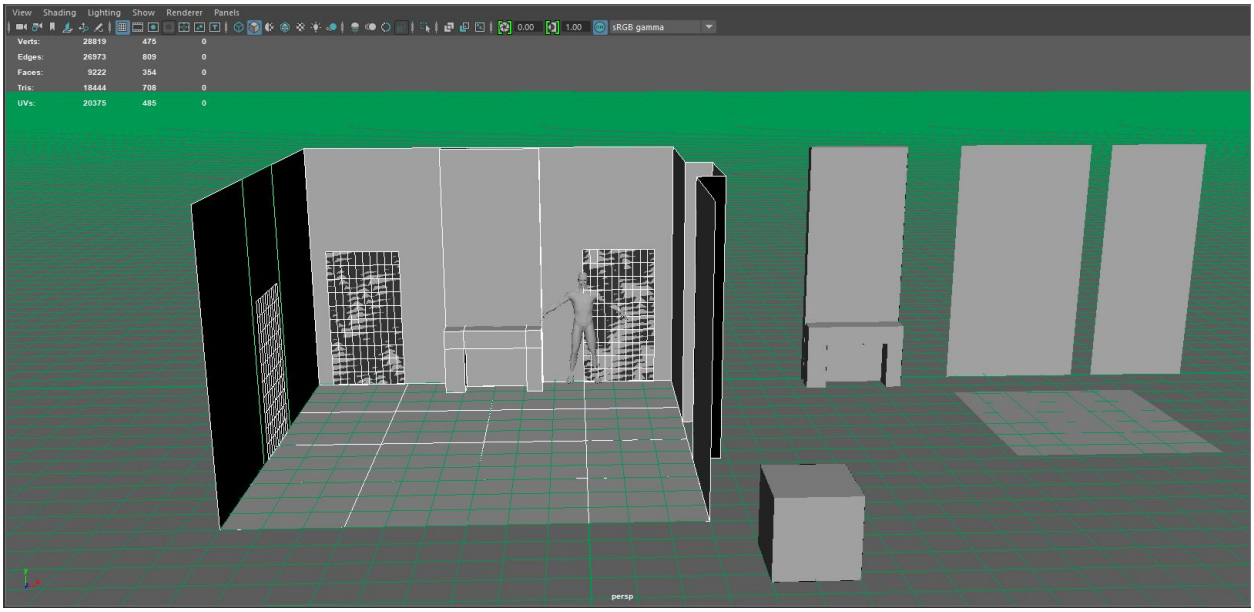
As an example, here are some ways to look at a reference, in order to plan:



Some modeling programs have a distance tool to measure your units. I have measured a base man here to use as a scale reference for when I work in my scenes.

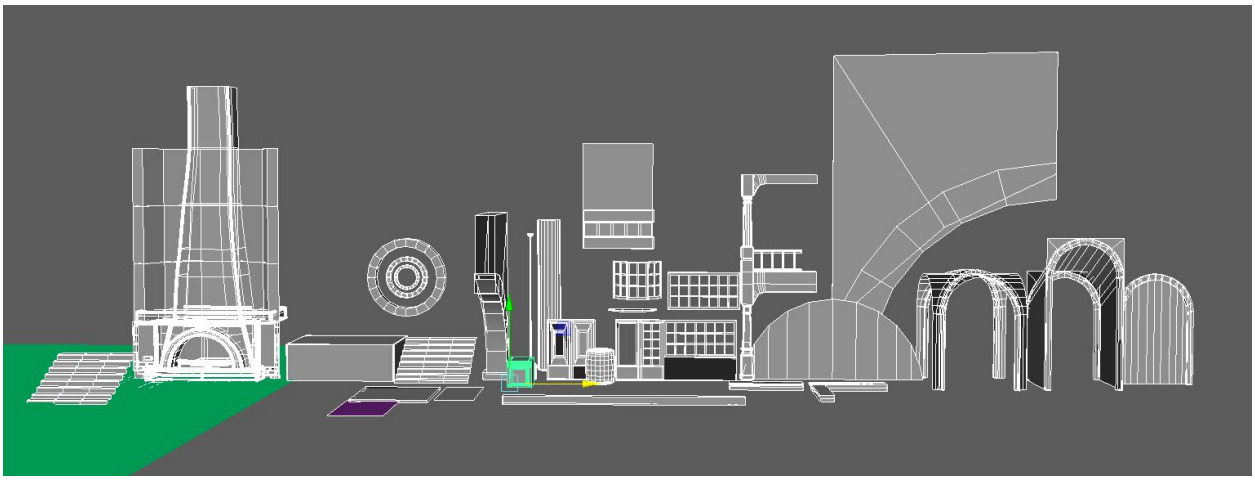
I can now overlay the humanoid reference and scale him accordingly to get the units for the reference. I have also color coated some of the texture callouts, highlighting essentials that I can use to plan a production schedule.

Here is another example of working from a photo reference. I used the park bench to estimate the building scale, sliced it into modules, and highlighted the trims.

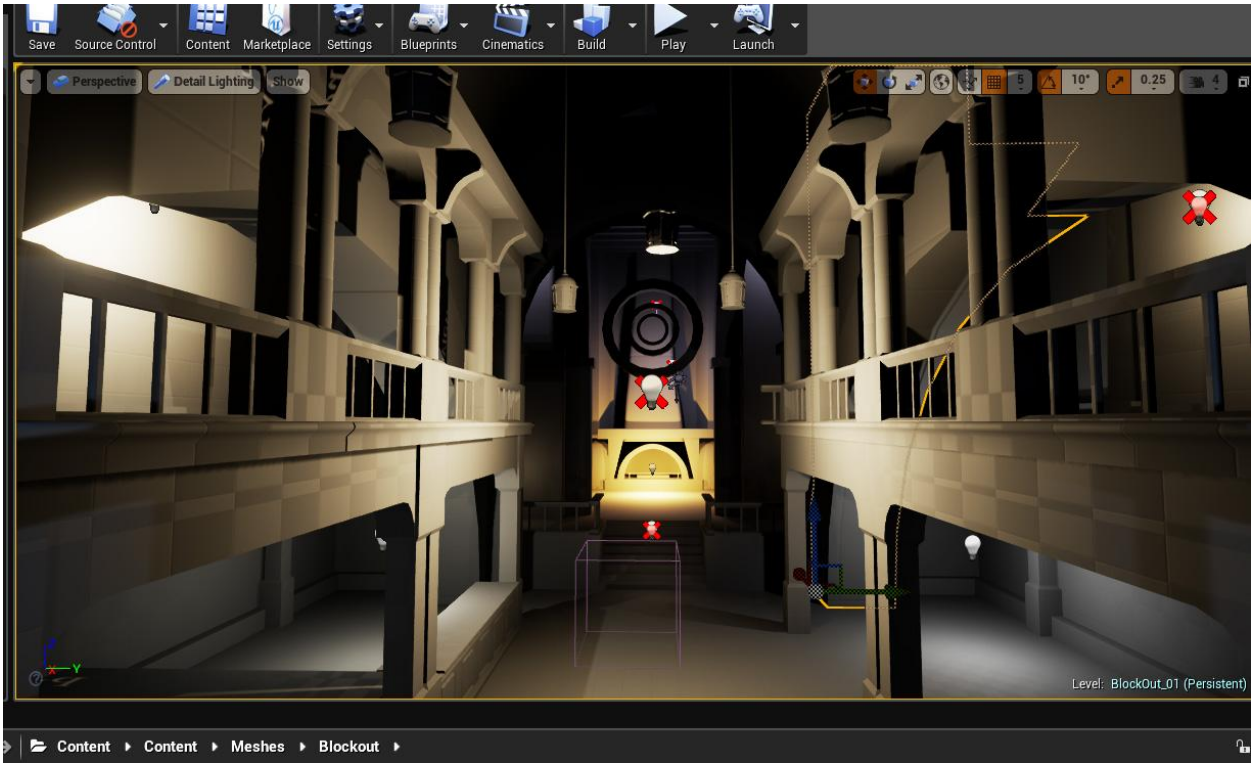


Back to the room image. With our units in mind, I can now do a quick blockout in Maya using simple planes to get my scale. This establishes my footnote and serves as the base or root guideline for all of my high poly meshes and game assets.

From here I can create my materials and set of assets such as this:

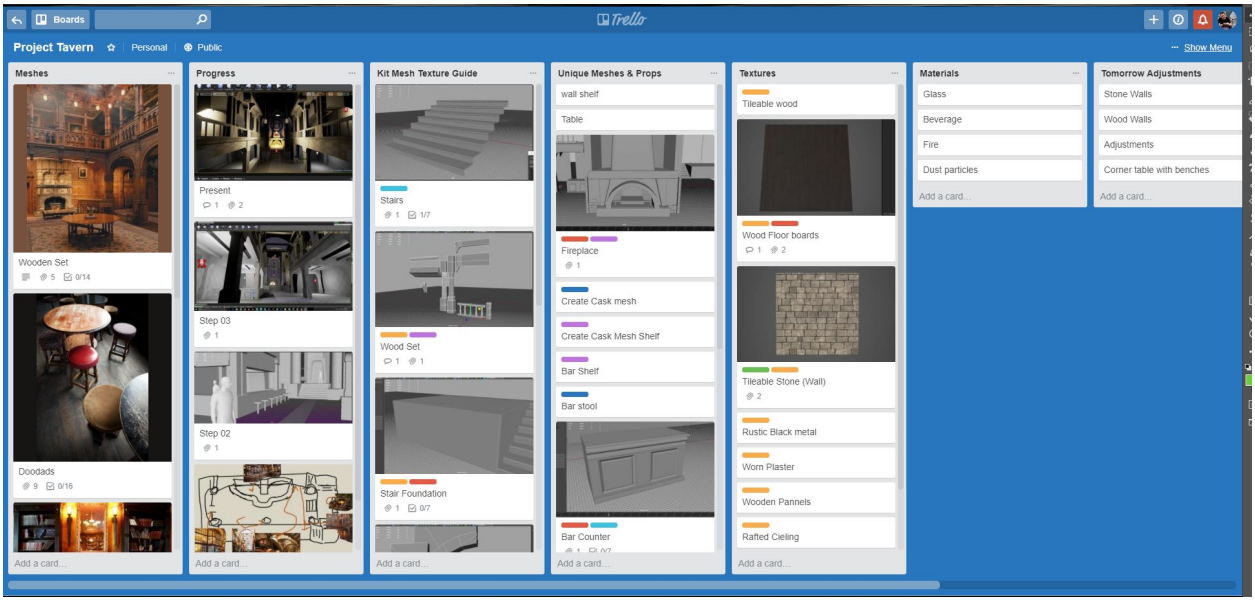


As mentioned earlier, we need to check this in-engine constantly to see if our modules line up as intended. Work with units of 5, 10, 50, 100 cm. Keep in mind that working with meters ensures perfect tiling of textures from asset to asset (if our texel density is 1k per meter). It is important to export modules with good pivot points snapped on the grid with the front face facing the Z-direction. It is also good to establish naming conventions so it will be easy to look up each module in the engine.



In Unreal, for this project I assembled my level using the base kit. From here it is about polishing upwards by creating materials, high poly meshes, and unwrapping. This scene was made from various references and is a custom space. For this I drew a basic top-down map as a floor plan, then started with my wooden beams and railings, then filled in the other details to support them.

Keeping track of progress looks something like this in Trello:



From the left we have my reference images, which help inform space, props, and materials. Next I have my progress, showing each stage. Lastly, creating different columns for an asset catalog, to keep track of each asset. Each card has a checklist and any images or notes I need to keep in mind. The color coding informs the material type (tile, trim, hybrid, unique) for each card. From here it's just about polish.

Process Continued—Look Dev

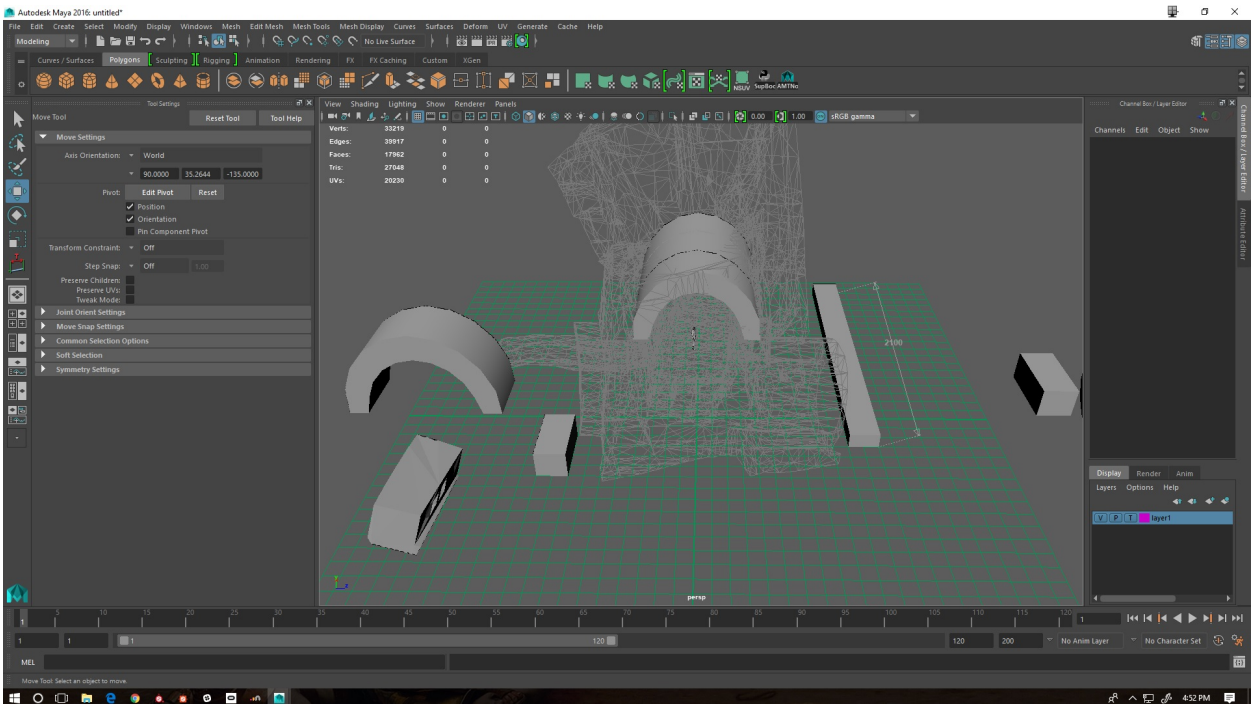
It is important to note that when working within games it’s ideal to be as fast as possible. Just because you know how to save time by using modular concepts does not mean your final will look the way you want it to. There is a great deal of back and forth to do, and in order to mitigate risk, it is key to iterate. Get to a *final* early and do a look dev test such as this one:

This image is by no means a final environment. There are many changes I want to make and materials to change out. To get to this stage, I used basic primitives, some premade assets, and a mix between my own materials made in Substance Designer, and materials downloaded from Substance Source. The goal is selling on a sense of atmosphere to inform an entire level. This scene was assembled in little over a week's time, which is very fast for one artist!

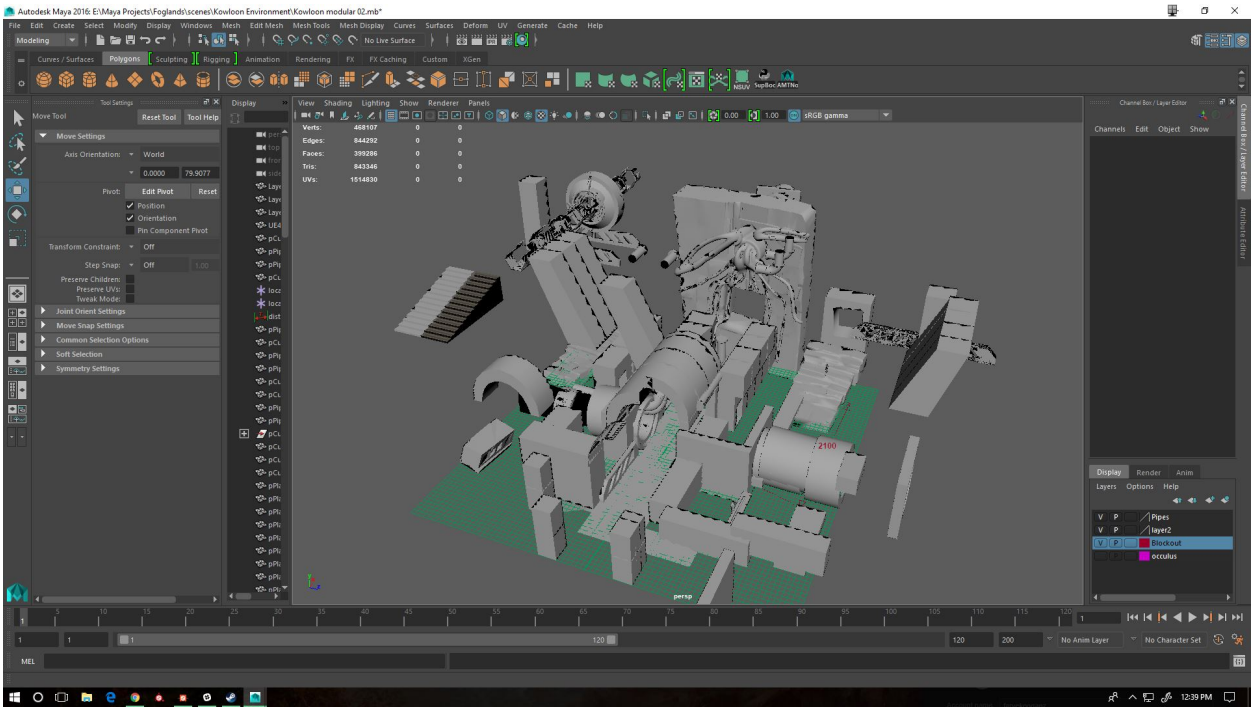
Now, something I realized early when using modularity and with level design in general is that if you always stay on the grid, a scene can become very boring. You want to strike a balance between a nice, organic composition, and the ease of use the grid offers.

In order to attain this balance, I use Oculus Medium* to start my blockout in. Working in VR with a voxel-based sculpting program is incredibly freeing, and simple for an artist to previsualize entire scenes. Using PureRef and Pinterest, it is possible to make large images that can be loaded into Medium to be used as reference, which makes the early creation and inspiration seamless. Additionally, because you are working in VR, I find it easy to judge scale and can get a sense of space far faster and easier than working back and forth between the engine and Maya. Furthermore, it is easy to perceive a scene in Medium from any angle, including basic lighting and ambient occlusion. This makes for a powerful iteration tool that gets you to the essence of your scene faster.

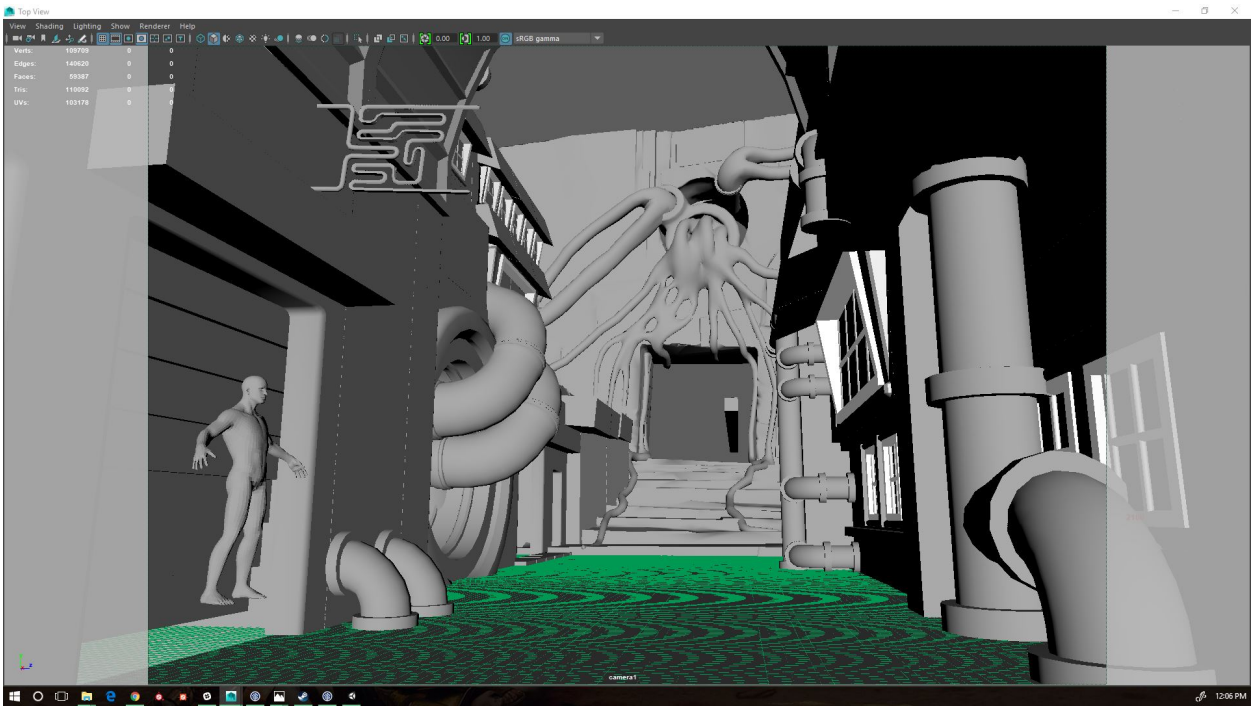
Here is my decimated mesh in Maya. It doesn't look like much, but it gives me enough ideas for how I can move on a space, early and more organically.



From here, I add the Oculus mesh to a layer to set as a wireframe, in order to reverse engineer it with modular pieces.

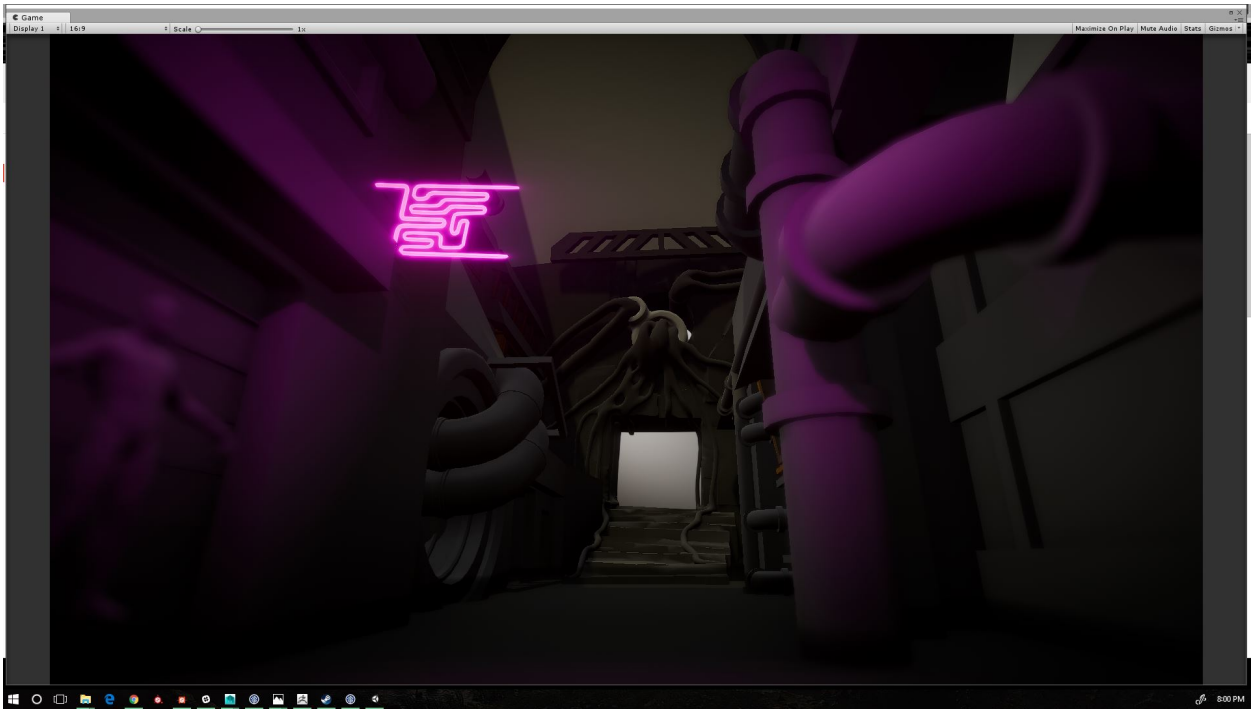


Next is to go wild with primitives, working around my footnote and getting things to snap together.

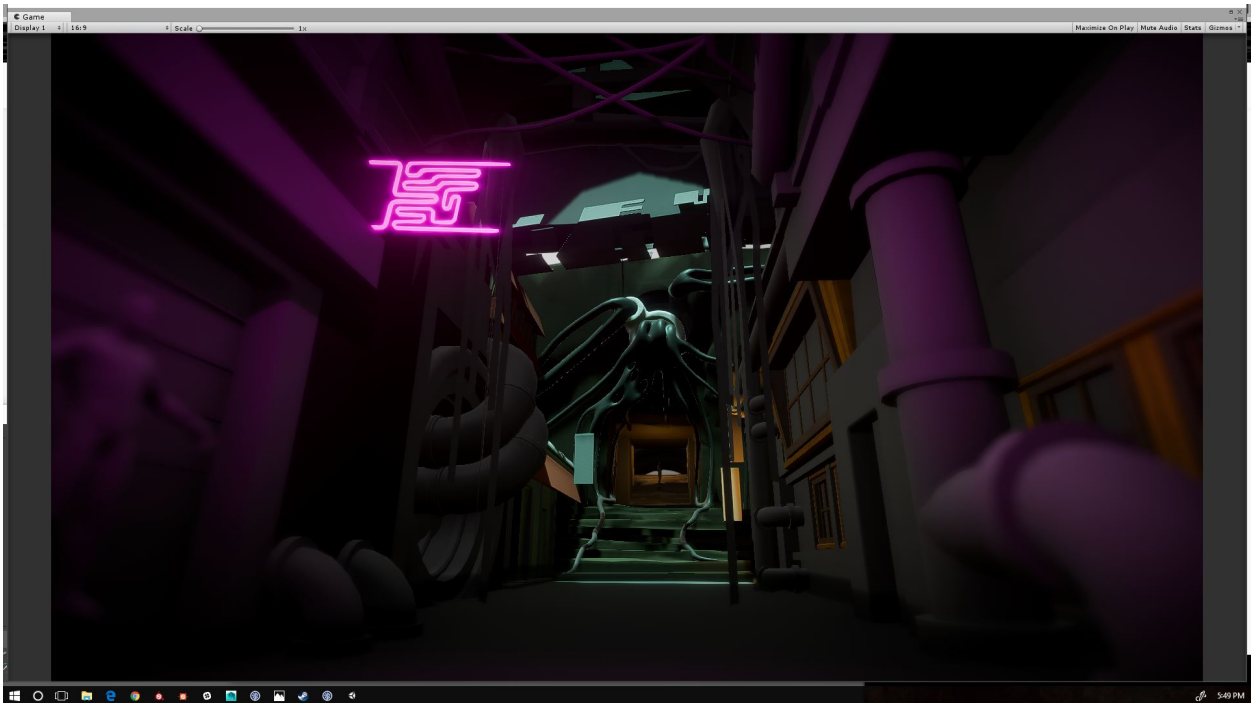


Now, I set up a camera in Maya and reorient assets to compose something more interesting. From here I save the file as an .MA, which Unity can read, in order to work quickly between Maya and Unity. The far wall with the weird growth was made using Oculus Medium and remeshed in ZBrush*. I wanted to capture a

cool set piece that had an organic cyber punk fusion. The stairs are also made in Medium, and reflect my initial sculpt.

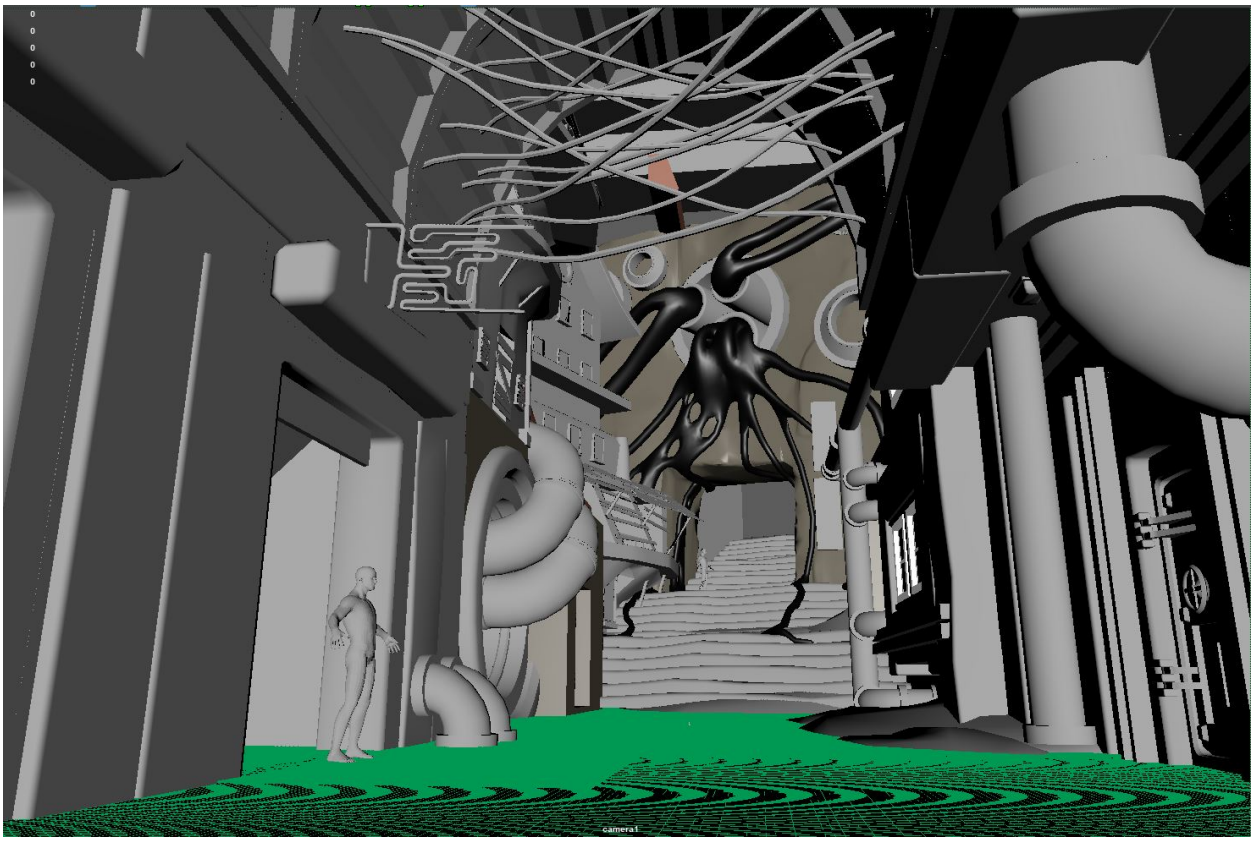


Here I have the Maya file imported in order to test lighting early.

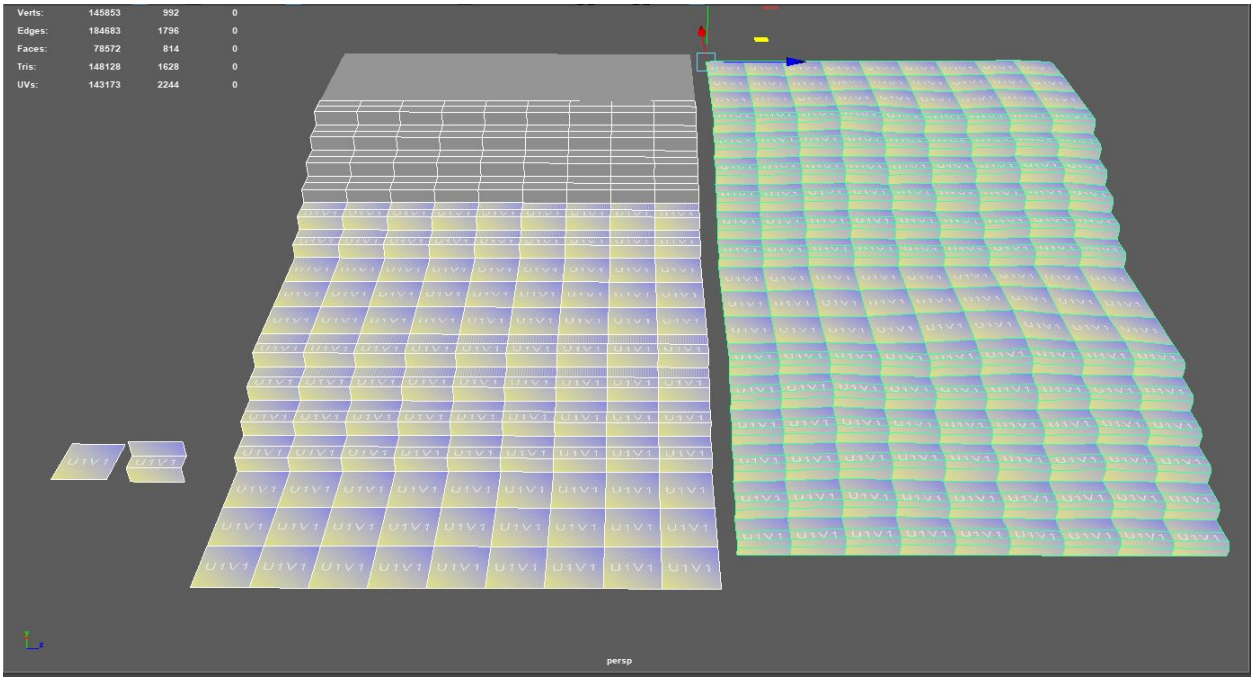


Getting the scene filled out a bit more.

It's a little hard to tell, but I overlaid my original Oculus sculpt into this scene. I absolutely love this. This is where the initial payoff comes full circle. Now I can see with fresh eyes and notice a sense of rigidity that I must break up.



Now I have offset the wall to an angle and added some elements to create some more organic motion as eyes look through the scene. The key addition is the stairs, which have a unique workflow that I like using.



To easily get everything tiling perfectly I use two planes, shown on the left. I weld them all together and we have it tiling easily. Next, I add some bevels to remove hard edges. Lastly, I use Maya's sculpt tool to add in noise to the geo, to break up the rigidity. This is the final stage of getting usable modules unwrapped. I prefer to use Nightshade UV, which makes things much easier than using the default tools.

I then export my scene in chunks as FBX files and begin assigning materials in the engine, finding values that work well with my lighting while also adjusting the lighting. My current step is the hardest part, polish. From here on it's on to the grind: high poly meshes, trims and tiles, and shaders to add variation control. Then, as a final stage we go on to optimization, to get this scene running in VR. Much of my look dev post effects wouldn't be viable. Also, I have yet to have the right shaders come online for the final look and style but that's okay, because I have the mood I'm looking for. When optimizing environments the three major drawbacks are:

- Geo: Simplifying the geometry and edge flow.
- Textures: Lowering the resolution; finding balance between lowering different maps such as roughness, masks, and ambient occlusion but keeping normal maps at set size.
- Shaders: Keeping the nodes short and sweet. Blend only up to three full materials within one shader at a time and utilize masks and constants to do the rest.

Note: sometimes it can be outside art causing performance issues (such as characters, scripts, or post effects), so be sure to check everything else first. Optimization is another puzzle and is tricky to get right, but modular workflows help make it a bit easier with fewer assets to optimize, and the possibility of reworking areas if need be.

Last Words and Advice

Projects meet a serious halt for not being well organized and/or too much polish on one asset too early, and not enough overall progress early on. It is a hard lesson learned. Do not feel the need to be good enough too soon; it's an iterative process. The hardest part about being an artist is knowing how to take feedback positively and build up self-worth when starting out. Sometimes it's grind, grind, grind, and it goes by all too quickly. Other times, problems abound and you are in a lull. In the end, have a goal. Know where you want to be. It's easy to get lost in your work and feel aimless or that you need to do more to get better.

The best advice I ever received was the only thing that truly matters is trying to enjoy the ride. Once you stop having fun a project can die quickly. Or worse, you could wake up when you are 30 and wonder where your years, grinding, went. Sometimes you just need a little more time to get better at the individual steps. Learning how to use software is the same as with any tool. The more you use it, the faster you get. You pick up tricks along the way and ideally never stop learning. Create an environment or project that focuses on a goal or something you want to learn or improve on. Don't aspire to just make something cool. That is not a goal and you will likely never be finished.

Modularity is one of the core pillars of game development for environment artists. Once you understand the basic fundamentals, you can begin to break scenes down into more efficient and approachable methods for environment creation. The most valuable asset an artist has is the sense of community within art production. It really is a glass door community made open and accessible, thanks to ArtStation*, Polycount, YouTube*, Twitch*, and more publishers of articles that focus on art production. Artists are finally in the limelight and you know who the rock stars are. Learn from these people. Follow them on ArtStation and aspire to meet your goals as you watch your peers meet theirs. Keep your head above water, stay inspired at every step of the way, and strive to see things from different angles. The rest will come in time.

Useful Links

Modular Design & Workflow: http://wiki.polycount.com/wiki/Modular_environments

Thiago Klafke's breakdown of building with modularity: http://www.thiagoklafke.com/modularenvironments.html

Step by step tutorials for various materials: http://www.philipk.net/tutorials.html

Substance Designer Tutorials/mentorship: https://gumroad.com/artofjoshlynch

Substance YouTube Channel: https://www.youtube.com/user/Allegorithmic

Fee collection of user-made substance materials: https://share.allegorithmic.com/

Dinusty (Senior Environment Artist/Streamer): https://www.youtube.com/channel/UCKfTBvcsO9Kw0b5EVuf4CJQ

Jacob Norris's Modular Building on Polycount: http://polycount.com/discussion/144838/ue4-modular-building-set-breakdown/p1

Batch Export Script: http://thiagoklafke.com/batchexport.html

Draw Calls Visual Guide. (This site also has a lot of great breakdowns and articles of game art tricks): https://simonschreibt.de/gat/renderhell/

Jacob Norris Polycount Forum on Modularity Construction:
http://polycount.com/discussion/144838/ue4-modular-building-set-breakdown

ChamferZone - modeling videos:
https://www.youtube.com/channel/UCx6FRkGCUg3NXut5qGOdgYg

Creating Floaters: http://www.bs3d.com/

Creating Crease Sets: https://www.youtube.com/channel/UCx6FRkGCUg3NXut5qGOdgYg

Texel Density and normal map baking: http://leonano.com/portfolio/category/tutorial/

Tech Art on Star Citizen:
https://www.reddit.com/r/starcitizen/comments/3ogi3o/im_an_tech_artist_in_the_industry_and_id_love_to/

Texture Sheet method: https://80.lv/articles/deus-ex-modular-one-texture-assembly-line-library-tutorial/

Nightshade UV Editor Maya Plugin: https://www.creativecrash.com/maya/script/nightshade-uv-editor

The Order 1886 - Material Pipeline: http://blog.selfshadow.com/publications/s2013-shading-course/rad/s2013_pbs_rad_slides.pdf

The Order 1886 - Environment Art and Lighting Pipelines:
https://readyatdawn.sharefile.com/share

Fallout 4 Modular pipe GDC: https://www.youtube.com/watch?v=QBAM27YbKZg

Environmental Storytelling:
https://www.gamasutra.com/view/feature/131594/environmental_storytelling_.php?page=2

Notices

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.