The World's Largest - Developing Large Urban Designs into Virtual Reality.

AS224505
David Weir-McCall & Abdulrahman Hadi Hammoud
CallisonRTKL & MIT

Learning Objectives

- Identify and implement simple techniques to optimize your 3D Studio Max files for Virtual Reality.
- Understand the migration process from 3D Studio Max to Data smith and Unreal Studio.
- Learn how to effectively import & navigate around large models within Unreal
- Create using instances, models and entourage within Unreal for optimal performance in large scale Urban Projects.

Description

Often, we see Virtual Reality being used on a small scale, hotel rooms and small bungalows. It is not often we see a large urban plan being adopted for Virtual Reality. Looking at large scale models, such as models from Planning and Urban Design, the functionality and process to bring it to life is completely different. Focusing less on the small details, such as furniture types and kitchen appliances, and more towards the functionality and interactivity to suit the larger scale designs. In this Class we will look at a CallisonRTKL Case Study, showcasing the implementation of bringing some of the largest urban developments in the world into Virtual Reality and creating an interactive area over 1,000m wide, and over 1,000m high. We will focus on the process and optimization of your 3D Studio Max files, and integrating them into Unreal Engine, with some simple interactivity to help navigate through large scale models, finally looking at how to simply maintain these models through the design process.

Speakers & Authors

An Architect and Enthusiast for Design Technology. After spending several years working with Apple and teaching at Robert Gordon University in the UK, his passion finally took him abroad looking for more creative challenges and opportunities. He has worked with Architectural Engineering firms, taking their first BIM projects in the region off the ground, facilitating in the training and roll out of the emerging technology. This has Included creation of training materials and working towards company BIM Standards. Currently David works as a Senior Designer with the Design Firm CallisonRTKL, focusing on the company's development in Design Technologies across the different sectors, including BIM, A360 and VR. He is part of the global team in rolling out and training in Virtual Reality within the firm, working to set up company standards for integration with current workflows and looking at how emerging technologies can bring new ideas and opportunities to both clients and the company.

Originally from Lebanon, Abdulrahman studied architecture at the American University at Beirut. There he focused on design technology and particularly, visualization, eventually leading him to game engines and their integration in the design process. Upon graduation he moved to the UAE where he took a role as a designer with CallisonRTKL. Abdulrahman is currently a student at the Massachusetts Institute of Technology where he is studying Real Estate Investment and Finance. He continues to be involved with virtual reality through the communities around MIT and Boston.

Company

CallisonRTKL combines the legacy of two great design practices into an even stronger, more distinct voice that is characterized by the strength of our ideas, the spirit of our culture and the passion of our people to make the world a better place.

People are at the center of what we do. Focusing on our relationships, with clients and teams, helps us make the big seem small, more personal, and keeps us moving in the right direction. Our process centers on creativity, cross-pollinating ideas and talents across offices to deliver quality at all levels and in all places.

Contents

Introduction
Before you get started

Before commencing with any VR project, we are required to understand the challenges and scope of the project at hand. With larger urban environments this involves looking at several key components. Firstly, it is important to consider the value and purpose of the VR experience and Master Planning. Even though the plans can span kilometers, it is essential to start developing the project at the level of the human experience.

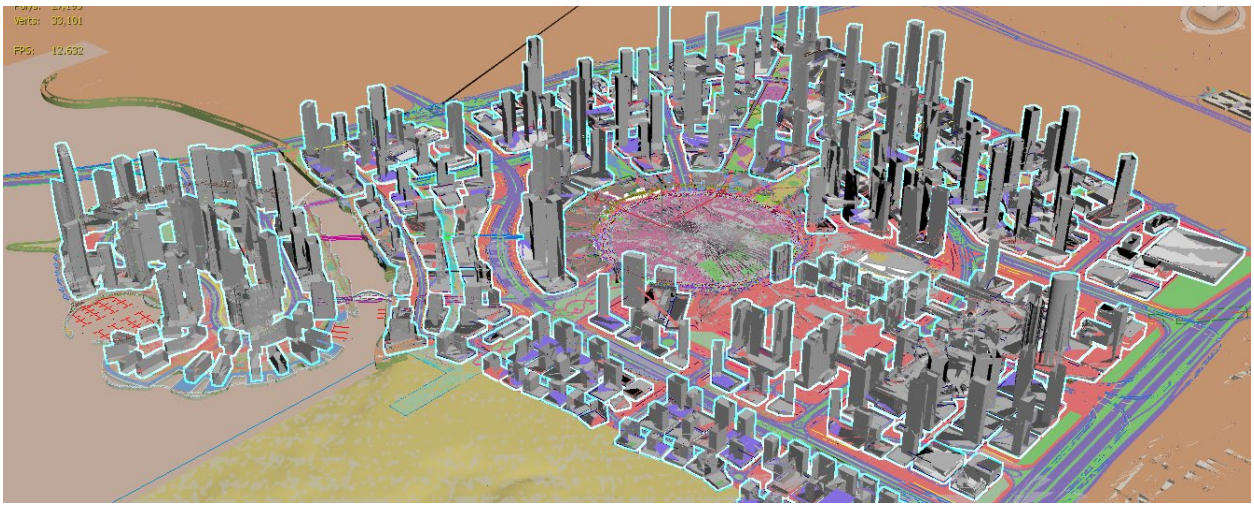
Getting Started
Start out the initial process by establishing the fixed components within your design. Ideally, if you're creating a VR experience you should be considering it from the beginning. This is to ensure
that most of your time can be spent creating fun experiences and troubleshooting, rather than re-modeling other people's work or designs.

Why Unreal?
With so many great products on the market the first question that always arises is; why do we choose to work with Unreal? Typically, there are no wrong choices when bringing your designs to VR. Yet, due to the scale and size of projects we are developing in CRTKL, it is the optimum choice to use the Unreal Engine to facilitate them. Additional benefits of this program are: It allows for a simple workflow with the preexisting architectural programs; its demonstrated its ability to handle and manage the models; and the level of detail that can be achieved. Lastly, one of the key deciding factors to work with Unreal is that it is a game engine, so allows you to narrate your own experience - as long as you have the skills and knowledge of how to do so.

What's your narrative?
The key benefit of using Unreal is its ability to craft a story and experience for its users. This gives it a massive advantage when we are considering master planning, as most common VR platforms do not currently provide this benefit. This capacity to personalize the users experience can be used in the navigation, the control, and even the quality. You control how and what your client sees. Imagine crafting a VR experience that is so intuitive that you don't even need to be in the room.

Learning Outcome 01

Identify and implement simple techniques to optimize your 3D Studio Max files for Virtual Reality.



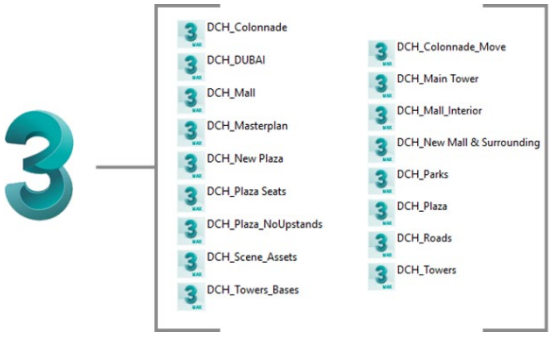
File Size

File size, though a basic element, is often overlooked when designing a VR project. With increasingly efficient methods of bringing our models into a VR environment, we fail to consider the complexity that the VR program must handle. The largest file for a master plan which we have dealt with was a 3D Studio Max file of over 11GB, we found this was too large a quantity of information for the VR program to support. So if you have a file anywhere over 100MB, you may want to reconsider the level of detail within it.

Split your Model

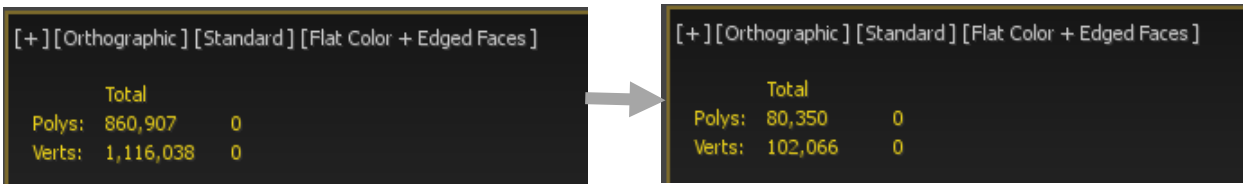
This initial step requires you to divide your model into smaller, more manageable segments. This is beneficial for numerous reasons.

1. Allows for easy identification of problematic areas (e.g. those that have a higher than recommended poly count)
2. Supports multiple users; allowing them to work simultaneously on different areas of the project, without interrupting workflow
3. Keeps the .3ds files small and manageable for average computers
4. Creates a library of components that can be easily updated and managed within Unreal Editor



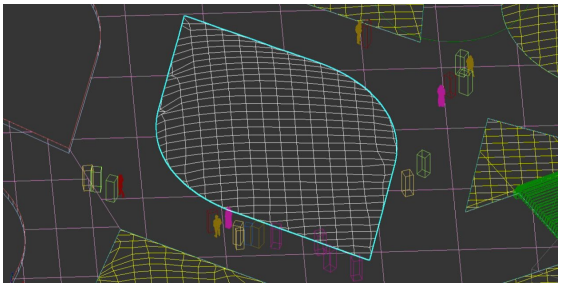
Poly count

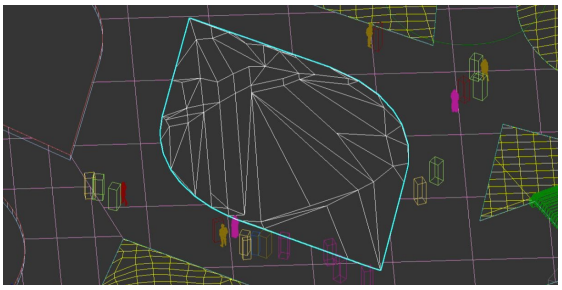
A seemingly minor part of your project that could surprise you is with how dense it is in geometry. Reasons for this could be because you have inherited it from another user or company. It is possible that the original use of the model was not intended for VR - models created for renders are typically heavy in geometry. Why is this important? Unreal and VR is a live rendering tool, meaning it renders every frame of every second that you are in there. When it has to cope with millions and millions of Polys and complex geometry, it becomes harder for the computer and the software to keep up. So, let's look at some tools and tips that will help us reduce the size of our models.

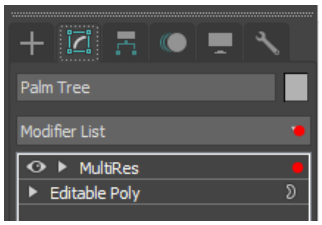


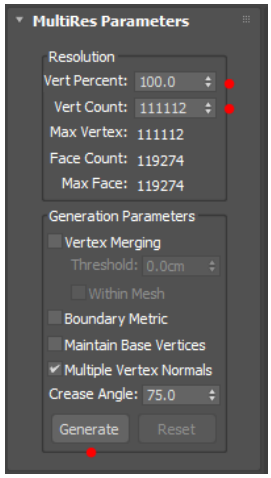
MultiRes (Modifier)

The MultiRes modifier reduces the memory overhead needed to render models by decreasing the number of vertices and polygons. This is useful not only within 3ds Max, but for content creators who export models for use outside of 3ds Max, such as Unreal. MultiRes offers several advantages over the Optimize modifier, including faster operation and the ability to specify reduction as an exact percentage or vertex count.



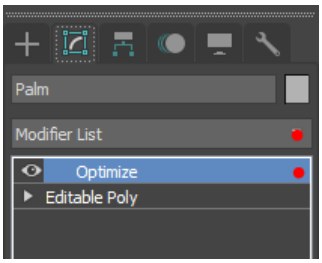


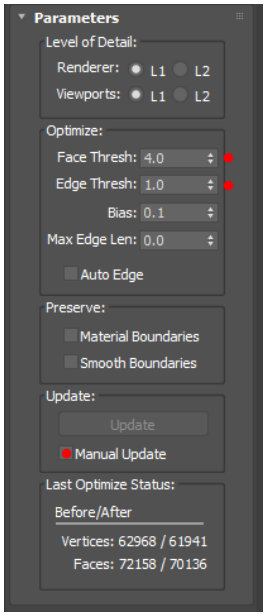




Optimize (Modifier)

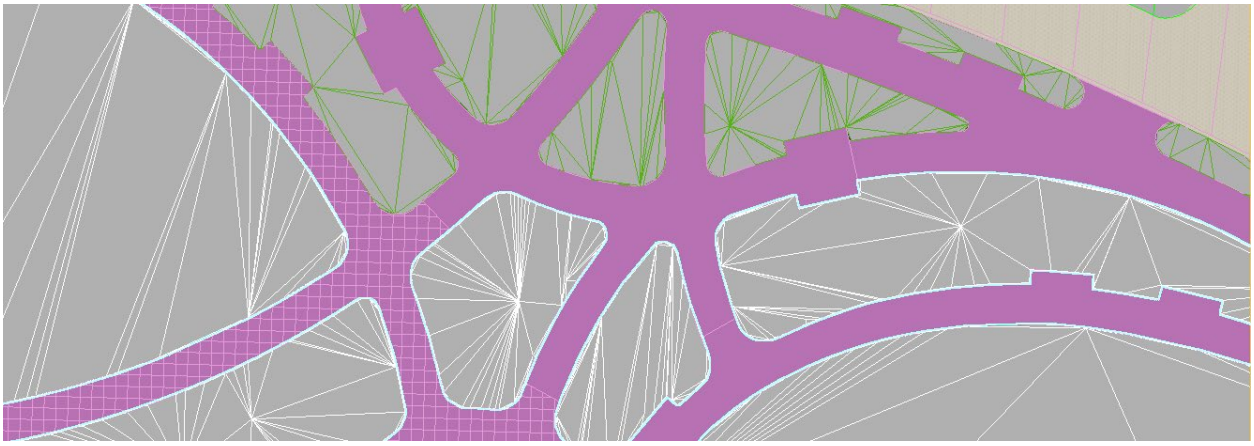
The Optimize modifier lets you reduce the number of faces and vertices in an object based on angle. This simplifies the geometry and speeds up rendering while maintaining an acceptable image. A Before/After readout gives you exact feedback on the reduction as you make each change.





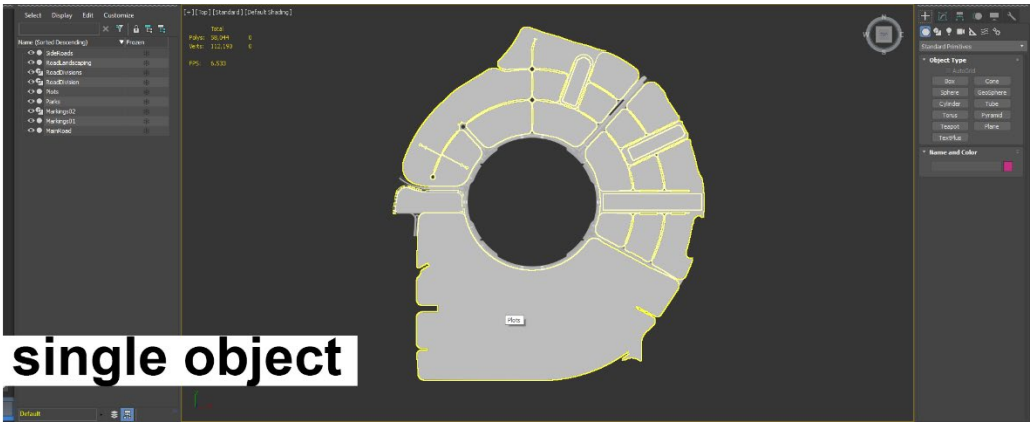
Edit Polylines (Manually)

Regardless of whether it's your model or not always review your model with the wireframe view port as sometimes even flat surfaces have unnecessary edges that can be removed to reduce poly count.



Grouping/Ungrouping

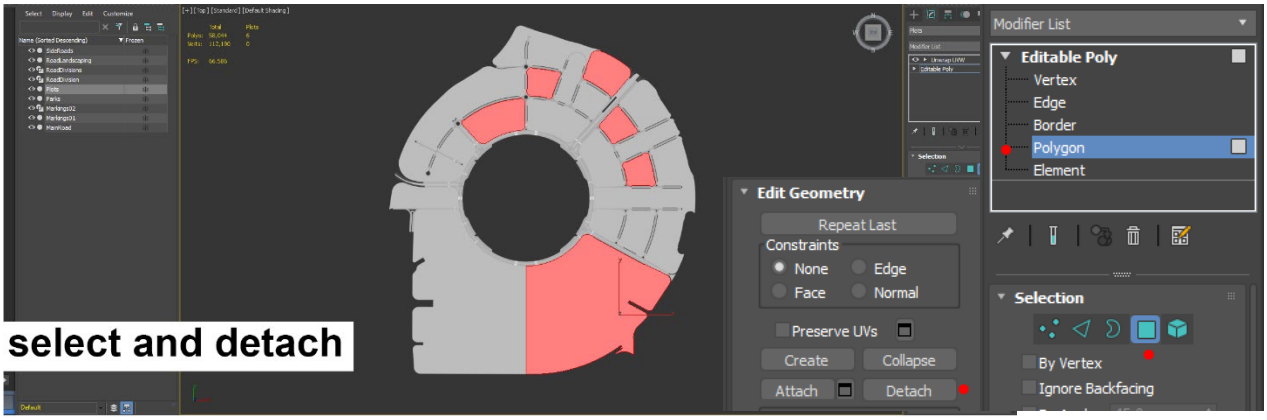
Unreal engine renders what it can see, whereas what it can't see demands little power from your computer. Certain VR experiences can be very slow and drop frame rate, because there is too much needless information in the scene being processed. In the case of large urban environments, a large grouped component can cause issues later for the project; such as roads, pavements, parks, and even towers. Why? Because even if a small part of the group is showing in the corner of your vision, as it's created in the form of a single component, the entire segment is consequently rendered in your scene. Let's see an example:



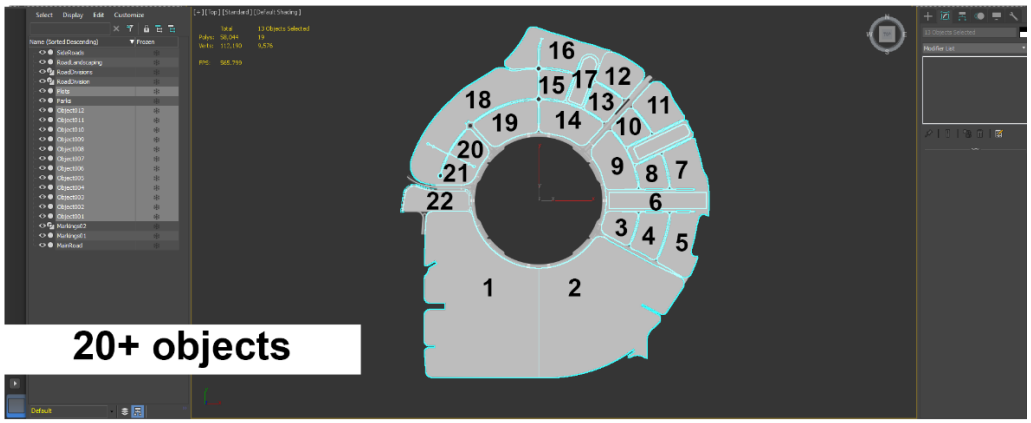
Here we have a plot map for a segment of a masterplan. On it are roads and the grounds around the developments, with the object you see above is almost 2,000m wide. Currently it is one object, where ever you stand it brings in the entire object.

Detach

A simple way to handle this is to break the model up with the detach tool. Within the edit poly command, you can select the faces or objects and detach them as separate components.

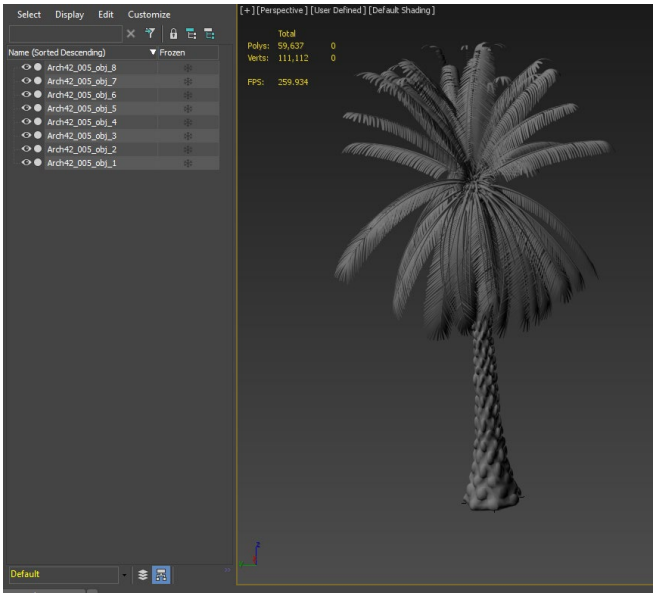


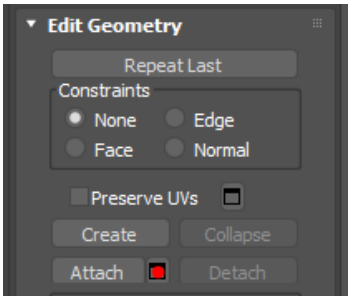
Once completed you will have numerous objects scattered across your masterplan. This will avoid additional and sometimes never seen parts of the model being rendered unnecessarily.



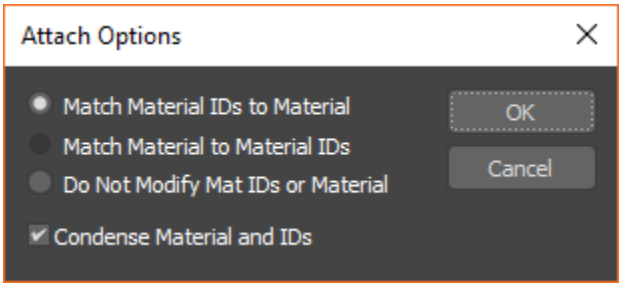
Attach

There are also occasions where the opposite effect is desired, accordingly the Attach Tool can be utilized. This usually applies to the smaller elements within your urban masterplan. As all individual objects enter into Unreal as separate elements, you are going to have plenty of objects in your scene to work with. Minimizing the number will make management of these objects much simpler. In this case, we would look to minimize the number of elements by using the attach tool. See below example:





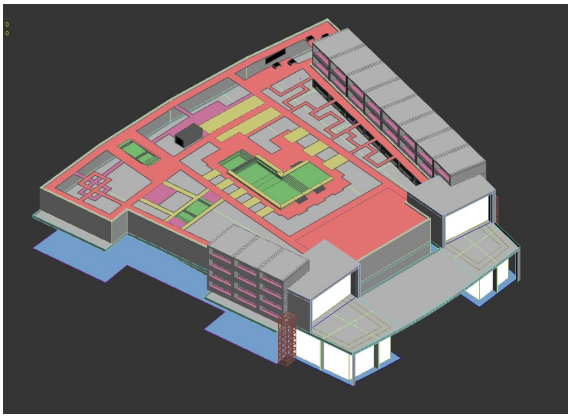
This single tree element has 9 Separate objects. In a master plan with over 300 trees the aim is to make management of these as simple as possible.

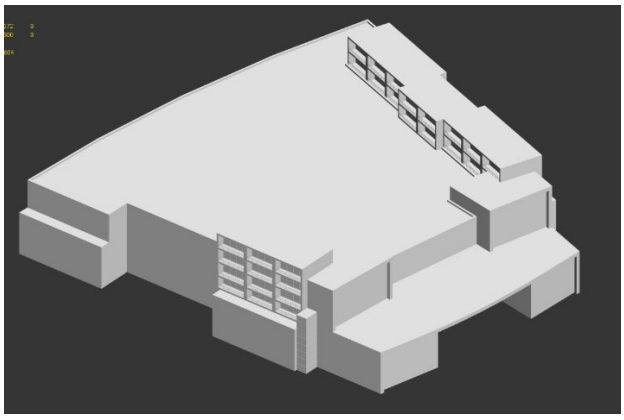


Depending on how you've set up your materials and material IDs you will select a different option. There is no optimum choice here, but you will have to ensure that your materials appear correctly before exporting to Unreal

Clean your files

Finally, when preparing your models for VR it's important to only retain in them features which you are absolutely going to see. Once you know your narrative this task becomes more straightforward; for example, towers that appear in the distance need minimal levels of detail, as they won't be entered, their internal layouts are unnecessary. The aim is to be selective with what you keep, as managing this and adding extra details in later, is far easier than trying to find new ways to optimize your 3D model.



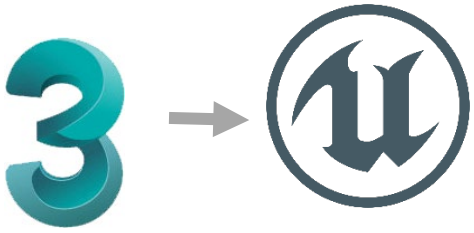


Learning Outcome 02

Understand the migration process from 3D Studio Max to Datasmith and Unreal Studio.

Now that you have your refined model, you are ready to start bringing this into Unreal and playing around with it. Prior to this we need to migrate our models from one platform to another. A few years ago, this would be achieved with exporting scenes or elements out as FBX files which came with a series of challenges. It would have to be manually assembled effectively forcing you to rebuild the entire scene and aspects such as materials, lighting, and cameras would be missing. This would require a lot of work in bringing it all together in Unreal.

With the introduction of Datasmith and Unreal Studio this process has become very streamlined. Once this data is loaded into the Engine, the data created by the importer is ultimately the same and all assets become standard Unreal Engine assets, like Actors in a level, Static Meshes, Materials, Textures, and Lights. Epic games are currently working with Chaos Group on a Vray plugin for Unreal that replaces the default rendering engine with Vray Realtime.



Datasmith

What exactly does Datasmith do for us? When you save your 3D Studio Max scene into an Unreal Datasmith file, all its assets are converted from 3DS Max components to Unreal Engine Assets. This includes:

- Object Instances
- Pivot Locations

- Scene Hierarchy and Layers
- Material and Physical Based Rendering (PBR) Characteristics and Textures
- Light Positions, Colors, Sizes, and Intensities
- Camera Properties
- Meta Information and Custom Attributes
- Unit Conversion

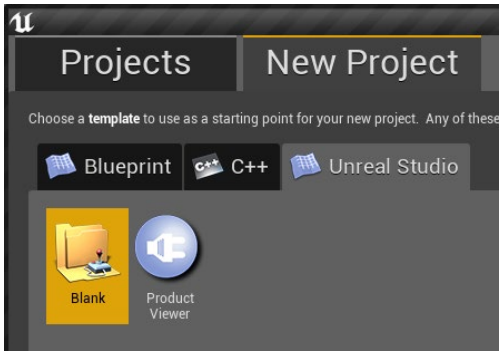
The importer also automates the generation and translation of time-consuming tasks like:

- Generating and packing a Lightmap UV
- Converting image formats to ones supported by Unreal Engine
- Converting Bump Maps to Normal Maps

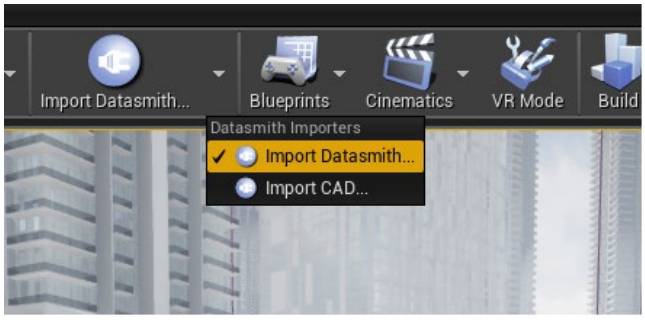
Some if not all these properties are useful when dealing with large 3DS max files or projects.

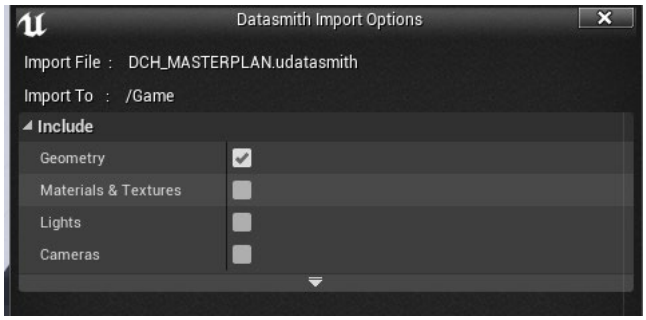
Exporting

Now that we have made all the necessary modifications it's just a case of exporting your model. With the Datasmith plugin you will receive an additional option when you export, allowing you to export to a .uDatasmith file type. In this contains all the information that Unreal will need to handle your model.



Within Unreal the Datasmith importer can ONLY be utilized when selecting the Unreal Studio template. Once opened along the top menu bar you will find the following icon allowing you to import your Datasmith file.





You will then be given the option to import certain elements into Unreal.

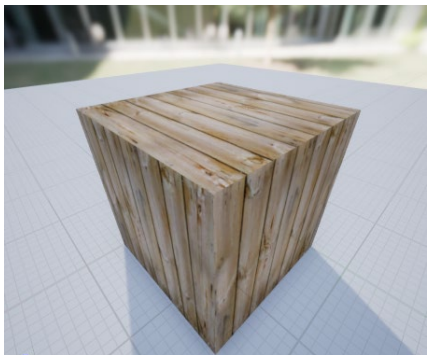
Lights and Cameras

3D studio max files can contain a lot of information that you don't wish to bring through. The default reaction is to make sure all these options are checked and bring through everything. However this can result in duplicate skylights, cameras, and lighting sources to your scene. Unless you have specifically set these up, keep them unchecked. At the same time if you wish to use your own lights in the scene be sure to remove any existing ones
created by Unreal.

Materials & Textures

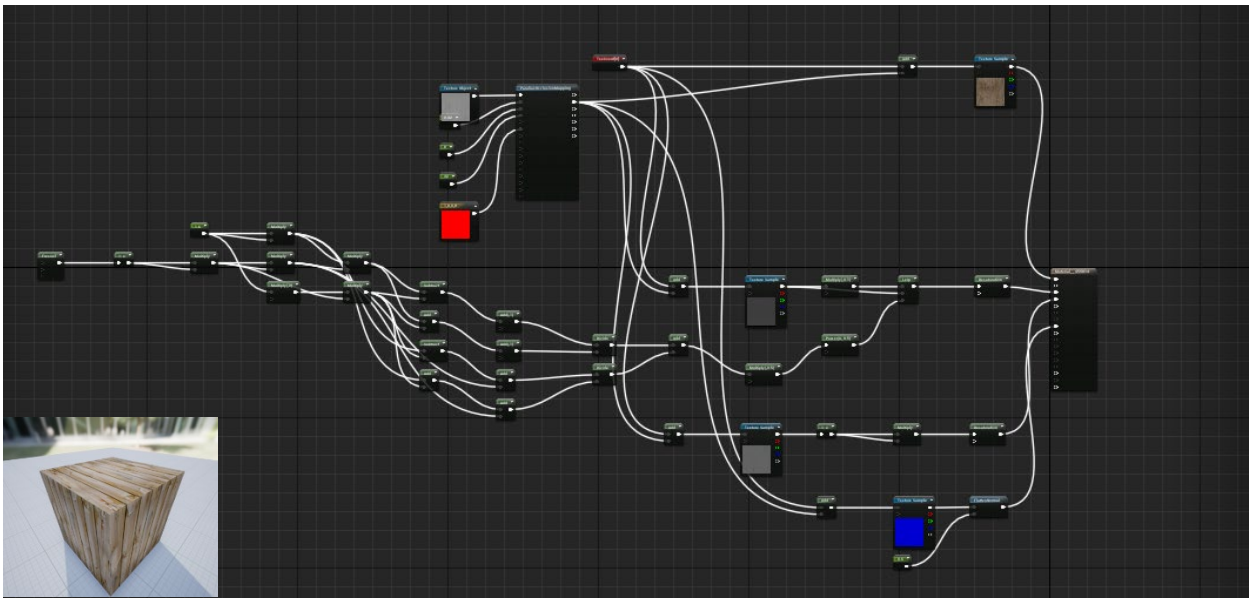
Materials with Datasmith can be a lifesaver in cases when you have customized your materials within 3D Studio Max, as Datasmith permits with ease the conversion from one software to the other.

However, this process is still being refined. Currently simple textures from Max can come across into Unreal with a lot of extra code than you might not have expected, especially if you are using a physics-based renderer like Vray that uses complex algorithms to replicate real world conditions.

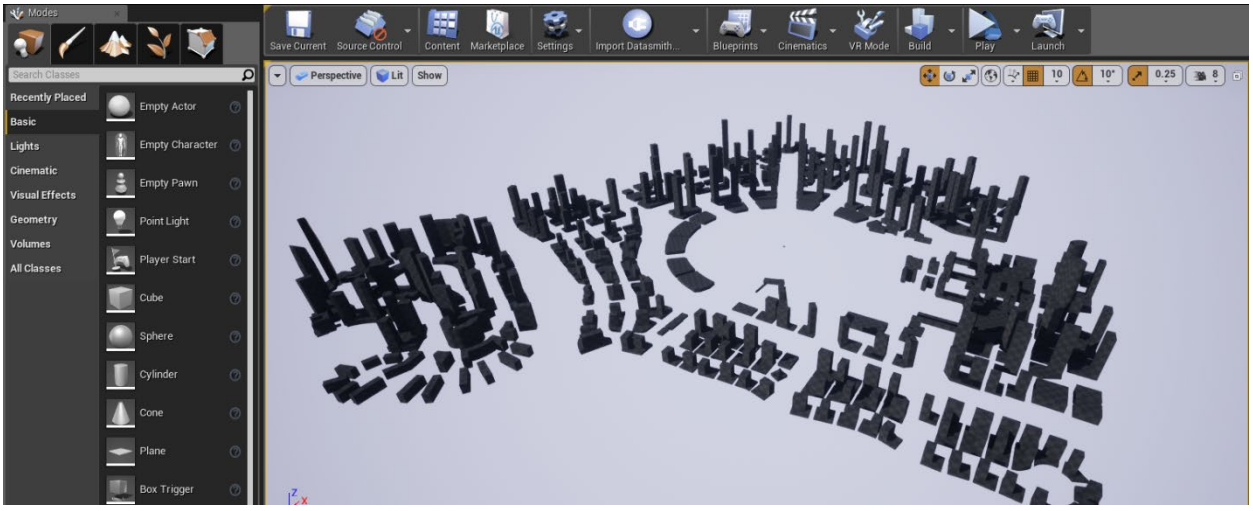


This is a simple textured wood material, from 3D Studio Max. This material has been created, altered, and adjusted slightly in Max. Upon importation into Unreal, our new

operating program, commences to break down each element in Max's material editor as a separate adjustment. As a consequence, you are left with large chunks of code. In the long run these avoidable large chunks of code can add up and will slow down your experience.



In instances where you have not given your textures the attention required in 3D Studio max, you can always apply them directly within Unreal with a little time and effort. If you chose to uncheck this option, then the models will come through without any materials attached.
In large projects, with numerous Datasmith imports and file management, these options are sometimes better left unchecked. This will allow you to control the materials within Unreal, and also to avoid having multiple material types, in your library, which creates confusion to the material selection.



Managing and Updating

Due to the ever-evolving nature of the assignments within architecture and urban planning, being settled in this profession is not an option, for it can fast lead to a becoming stagnant. Yet with multiple design options, changes to existing plans, and alterations, it can be challenging to stay on top of your Unreal model. Whilst Datasmith does help manage such changes, it is still imperative to be careful during your application phase. For example, choosing incorrectly can result in having to reapply textures, materials, or even commencing collision mapping back from the beginning.

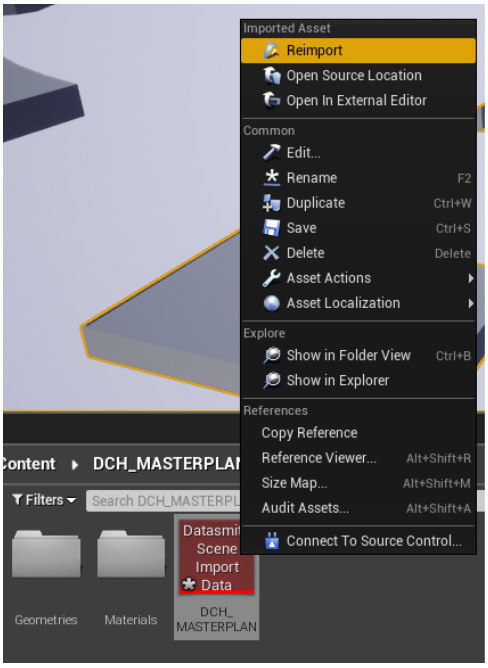
Fortunately, the management of these updates has been made considerably easier since the introduction of Datasmith, which now allows for a simplified integration process.

Full Scene Re-import Vs Asset Reimport

The two options available when updating an existing Datasmith file are a Full Import, or a Partial Import. Depending on the scope of change this will dictate which action would be most appropriate.

Full Scene Re-Imports are advisable under the following circumstances:

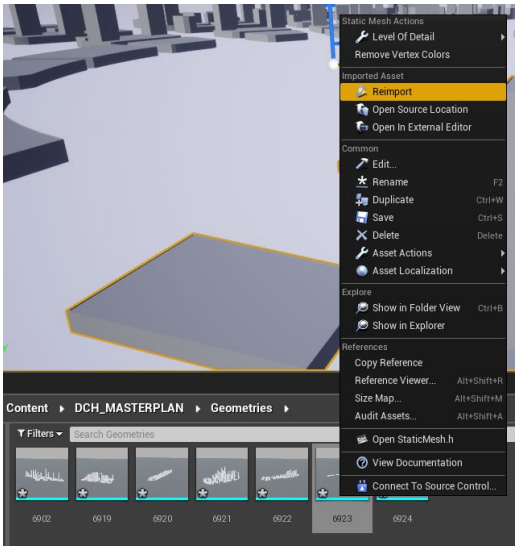
- You need to bring in new objects or materials from your source scene which did not exist previously in Unreal. Such as adding additional details *e.g. pavements, planters, road markings in the 'Roads' Max file.*
- The layout of the scene objects in 3D space has changed, or objects have new parenting relationships in the scene hierarchy that you want to be reflected in Unreal. Eg. *Relocating buildings and arrangements in a scene.*
- Or, you cannot re-import your assets individually. This may be because there are too many objects altered in your source file to work with them one-by-one, or because the assets altered are not specified? *E.g. Client has altered or changed the entire masterplan*.



The Partial Import option alternative is recommended when your project is developing smoothly, for example:

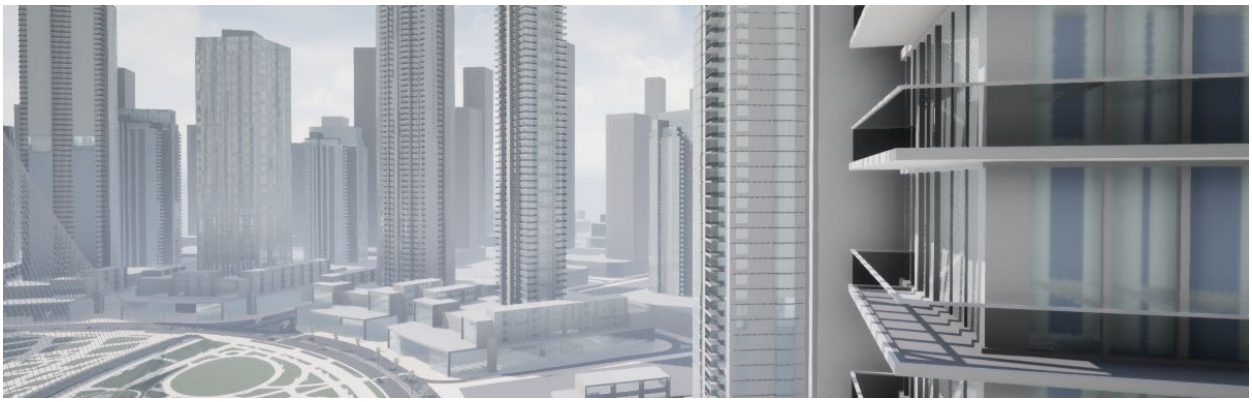
- When you want to import changes to a small number of known assets but bringing in the whole Datasmith scene would result in changes to several other finished assets, and/or alteration in scene hierarchy. *E.g. adjusting the road width, material, or shape.*

- You wish to apply different Datasmith import settings to particular assets. Rather than having the original ones, that were established in the first importation of the whole Datasmith scene. *For example, adding materials in Max later during process.*



Run Performance Check on Your Model

Now we reach a very crucial part of the process, the performance checking of the scene. This will consume approximately 70% of your projects time, due to its importance in ensuring our overall aim. This aim is to be able to run the VR experience, without an entourage or additional components, as smoothly as possible with a consistent frame rate. This can be achieved in several different ways.

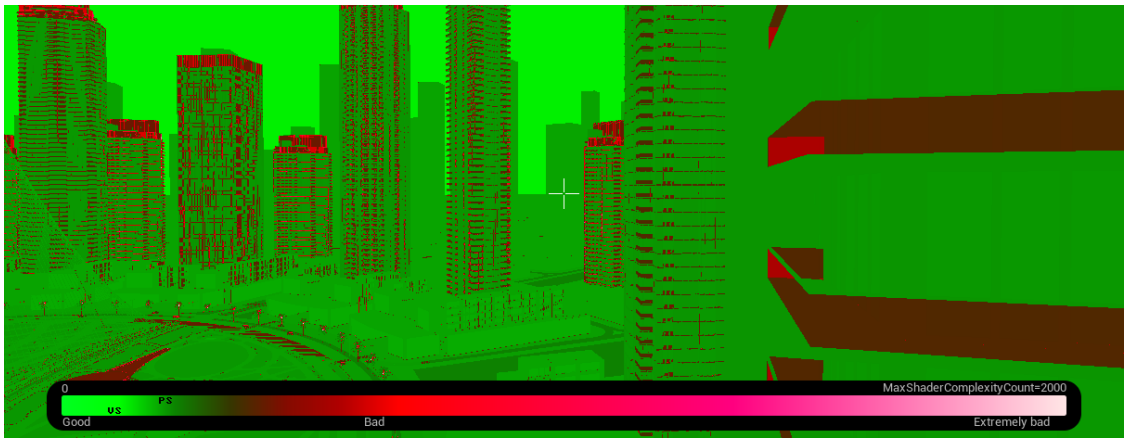


Shade Complexity (Alt + 8)

This view allows you to see the number of shaders being calculated per pixel in your current view; with green being the lowest, whereas red and white rate as the highest (most expensive to your graphical process).

It is preferable, before moving forward with your project to have this scene shine a nice shade of green is preferred. The areas seen below in red are one of the biggest hits to

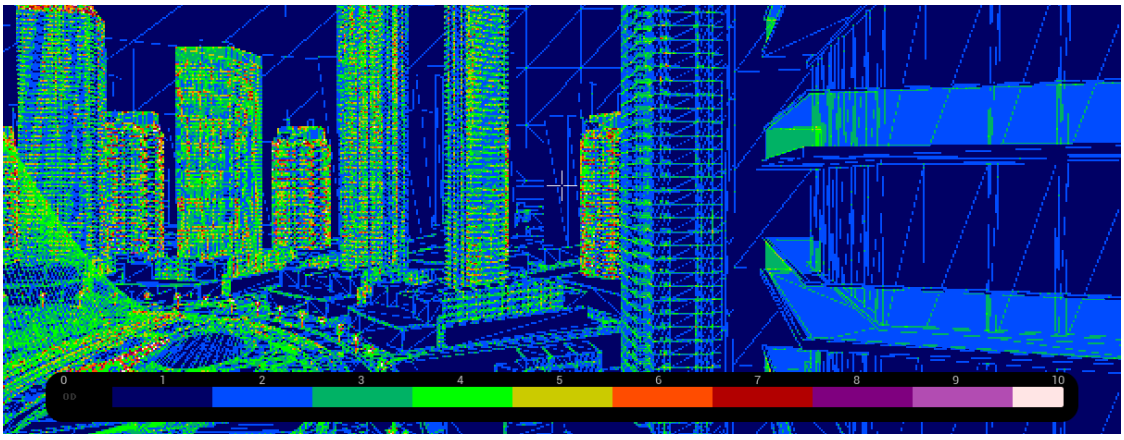
performance, with opacity, such as balustrades and glass being some of the most frequent and common culprits. In addition to foliage and particle systems. This will end up being a tradeoff, judging what is important enough to keep and areas that don't really require it. In order to reduce the complexity of your shaders the number of materials, and the number of texture lookups within those materials, must be reduced.



Quad Overdraw

This is an ideal view to identity the locations of the heavy elements within your model. When multiple polygons are overlapping and being rendered within the same 'quad' on the graphics processing unit, the performance of the scene will suffer; as these renders are calculated and then unused.

This issue is created by the complexity of the meshes and the distance they are from the camera. The smaller the visible triangles of a mesh, the more that are present within a quad. As with shader complexity, opacity is a major performance hit and causes a lot of overdraw. There are some ways to reduce the overdraw: using LODs (refer to Learning Outcome 04) to reduce the poly count of models at a distance; or further reduce the complexity of the model (refer to Learning Outcome 01); and reduce or remove transparencies in particle systems and foliage.



Frame Rate (`stat FPS)

Frames per second (FPS) is the best way to judge the performance of your model at this stage in its development. Epic requires something in the region of 90FPS, in a VR experience game, for an ideal aim rate. Also, be very cautious of FPS readings, as they can differ between the editor and the actual VR game play. It is also worth monitoring FPS in the views that you or the client are intending on seeing. Being frustrated that when you view every single component in your model at an angle no one will ever see reduces frame rate below the recommendation, will have you running in circles. Typically, in large open environments of masterplans we have a less strict FPS goal in mind, due to it being more informal and not to be published as an official game. However, at this stage of the process we need to aim high as the majority of heavy processing elements will come in the next couple of Learning Outcomes.

Check the stats (`stat Unit)

Another useful tool to be aware of is the Stat Unit viewer. In short, these set of statistics are there to tell you exactly how long it takes to process or render each frame. Refer to the explanation below:

- **Frame** is how long it took your frame to do everything.
- **Game** is how long it took the game logic (this encompasses C++, blueprints and the engine itself.)
- **Draw** is how long it took to draw your scene (what you see in your viewport).
- **GPU** is how long it took your GPU to do everything that was sent to it.

The recommendation with this tool is to bring it up whilst you are testing your views and experience in VR. If this starts indicating red or yellow, then this tells you that the end user is not going to have a smooth experience.

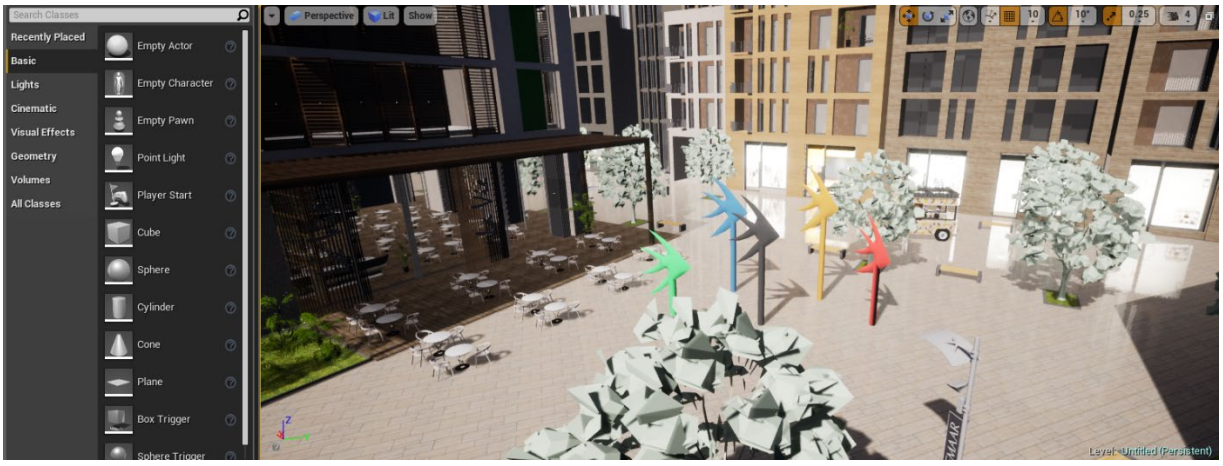
This ties everything in the performance section together, you'll typically notice that depending on what you are looking at these figures will change, understanding what you want the client to see and experience will help you focus on what you need to reduce or work on.

The important thing about this is you will never get it right first time round, it requires a lot of optimization and fine tuning. The aim, however, is to get this section working smoothly with a goal rate of 100-120FPS; understand that as we start adding people, cars, plants, and materials this rate shall increase.

*****Repeat Learning Outcome 01 and 02 to refine to optimize the model before moving on*****

Learning Outcome 03

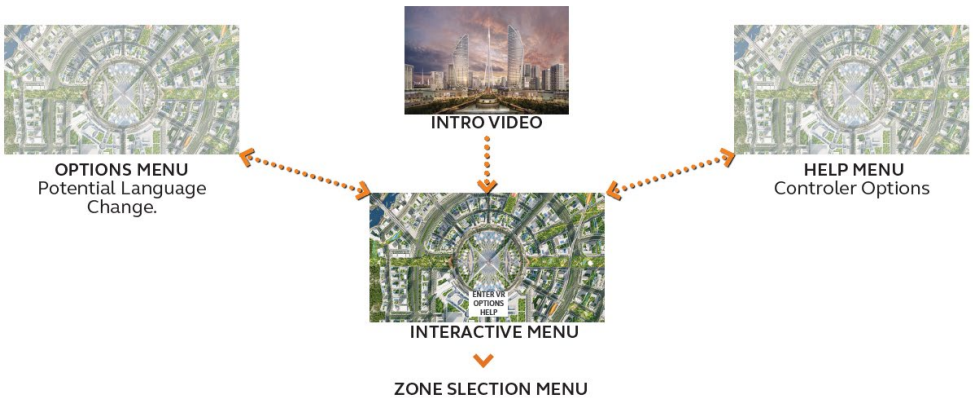
Learn how to effectively import & navigate around large models within Unreal.



We've used Datasmith to import our model, and now we have our huge complex urban environment to navigate.
So, what are the differences between navigating through a typical small sized VR project compared to a massive 2km 3D model? How do you begin to craft that user experience? Aiming to have the client understand the project, and even have the ability to navigate it themselves. It all starts with narrating your own experience, whilst keeping in mind your target audience. What is important to them? What is important to your scheme?

Narrate your story

Every project is unique. Therefore, it is important to tailor make each VR experience to suit its use. First, you must identify what the needs are that you're dealing with: whether the purpose is to show a customer's journey; provide a site evaluation study; to be a model for environmental impact, feasibility, or special awareness; or even to demonstrate work in progress. Each VR experience that we have created has simple principles in mind which help user navigation through the space in Unreal. Starting with how the user enters the experience.

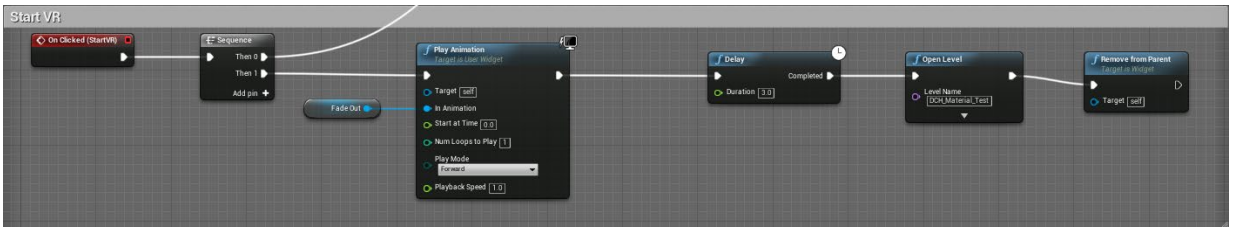


Set up a Menu & Zone Selection

Similar to computer games, DVDs or even Netflix, which commence with a welcome screen and menu, we do the same in VR. This provides for the user, an important sense of familiarity as it is instantly recognizable. This then gives them confidence in their ability to engage with the technology, even though it may be their first-time using VR. Standard introductions include promotional video or a company logo, followed by a welcome menu. Menus providing simple options, such as:

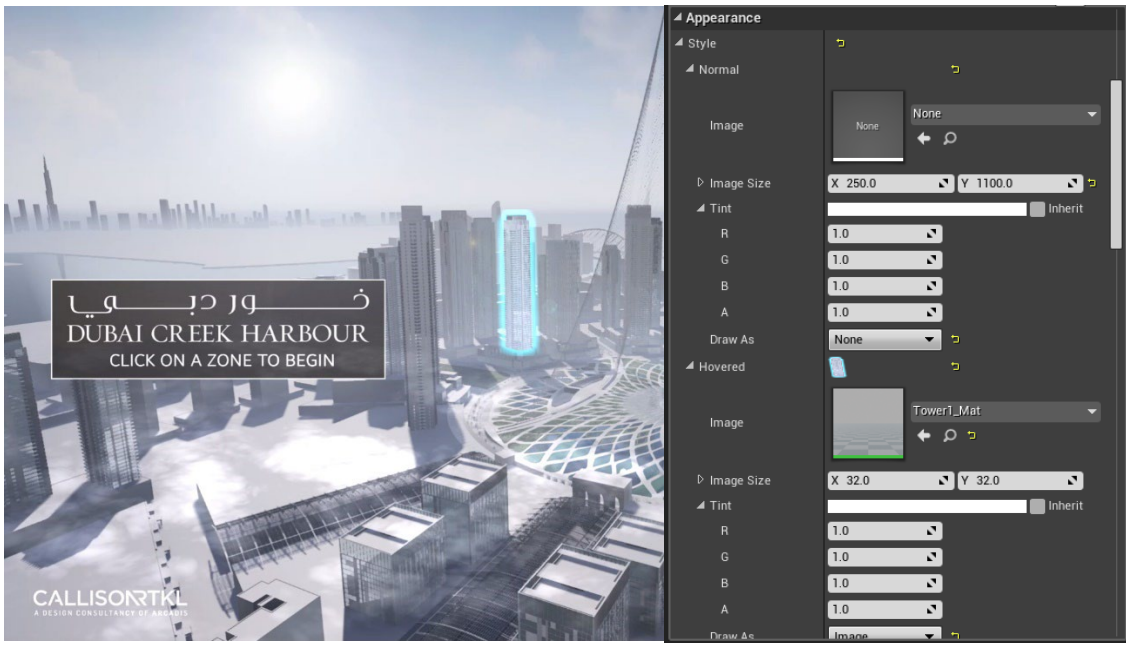
- Begin
  Clicking this option will start your VR experience
- Controls
  If you are not about to help guide the user a controller option is always advised.
- Options
  This is optional if you wish to have language features or resolution settings.
- Quit
  Unreal's default is to take over the desktop screen, having a quit button allows users to leave the experience, if required.

This is achieved very simply with a widget in a new level, a few buttons, and some small lines of code. For a customized experience, this widget also allows you to animate buttons on command.

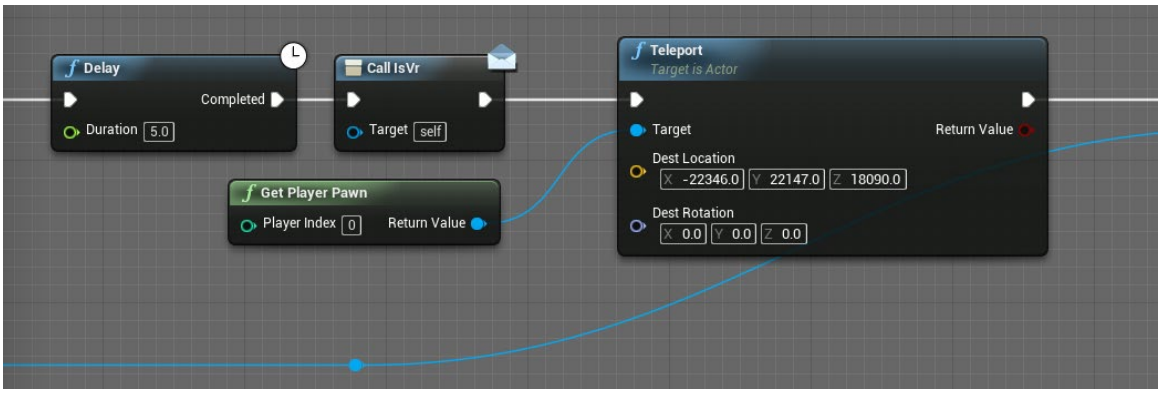


Zone Selection

In larger masterplans, when the client clicks ‘Begin’, we want to center them in an instantly recognizable zone. An ideal starting place is for them to see the overall Masterplan, and the potential areas within. However, there are some complicated aspects in the setting up of the Zone Selection menu. Yet our aim remains, to make navigation as simple as possible for the user. In the below example, we do not enter the VR until the zone is selected. The reason? Simplicity for the client. A prime example of simple zone selection is having the buildings and areas react to the movement of scrolling your mouse over them. It immediately informs users, that these areas can be chosen. In this instance, we have used the images as buttons.

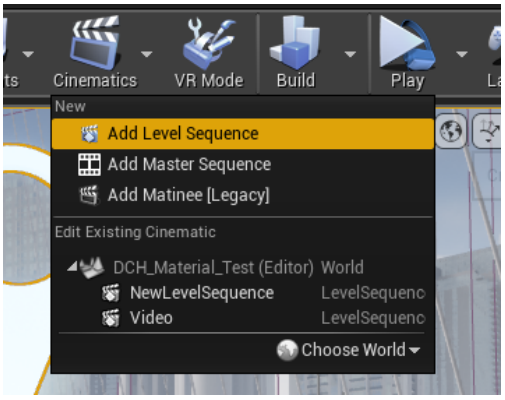


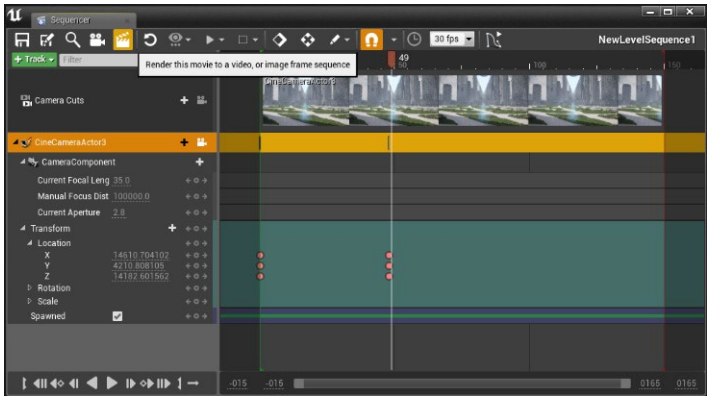
Next, we want to program each button. So that upon selection it take us to a predesignated location. In this way the user, whether it’s you or the client, can jump to their desired location or viewing point. This can be achieved either with Multiple Player start options situated around the level, or by simply by receiving the exact co-ordinates in the zone you wish the player to start. In the case below, we use the teleport option.



It is recommended that you dedicate time to effectively preparing your menus. Similarly to Main Menu you can construct the zone selection menu and make it as complex or as simple as you like. However, as it is the first point of contact for the client, to the experience, an easy 'fade-in-and-out' animation can make a lot of difference to the presentation of the menu.

In our design example, we animated videos in and out of the scene, and employed them as 'cut scenes' during the transitional stage. How? Amazingly fast and effortlessly using the cinematic option. Creating a camera and setting up a series of 'position' key frames, will easily allow you to render out a full HD animation clip in a matter of minutes.



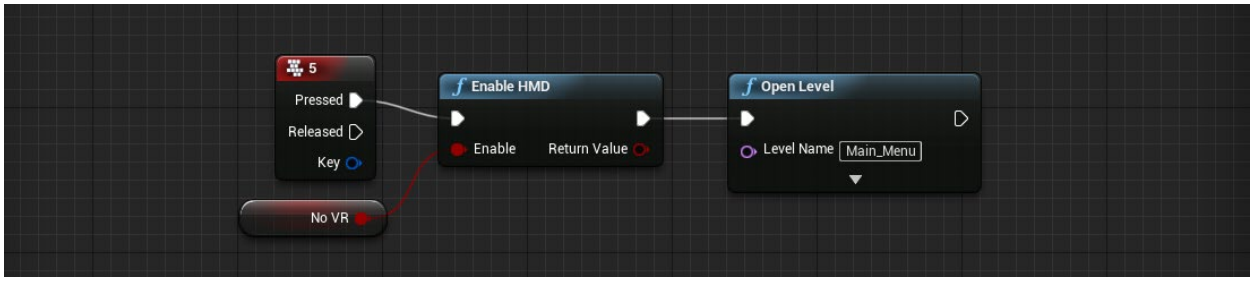


This option promotes one of Unreal's greatest aspects. Which is its ability to grab fully rendered videos, with the click of a few buttons. There are no render farms, and no longer the need to leave it overnight. Once the model is there, it is yours to extract what you need.

Simple Keyboard Commands

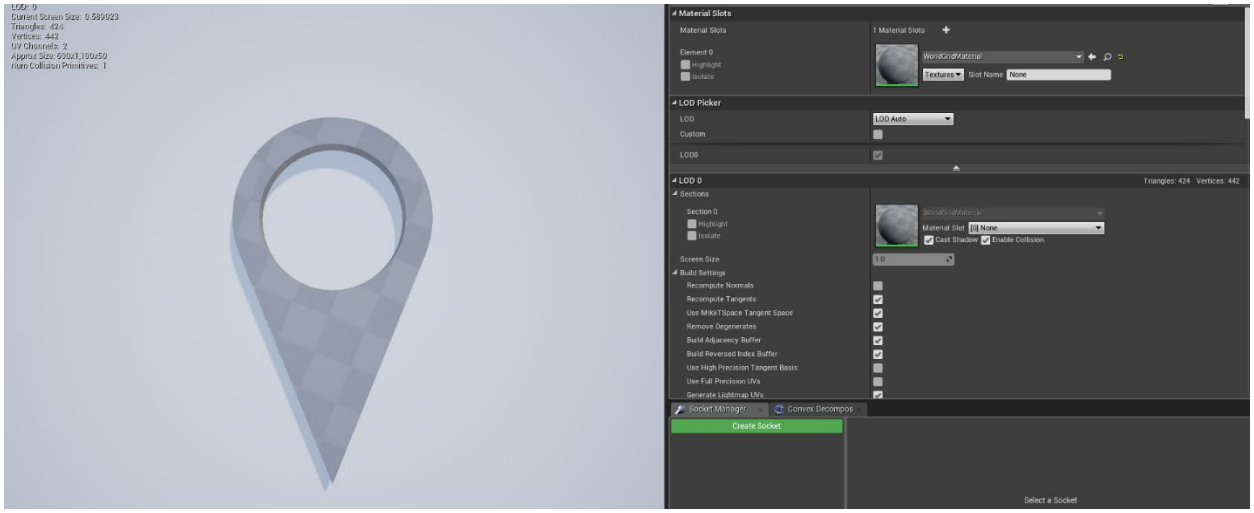
Short keyboard commands are an important tool to rapidly reset and return to the beginning. They allow users access to menu options should they require, or if a new player enters, and they wish to switch zones. In these instances, this is a quick and

effective option. Alternatively, you can program a button on the motion controller to achieve a similar goal. In this design, we use the ‘Level Blue Print’ to facilitate this option.



Teleporting Icons

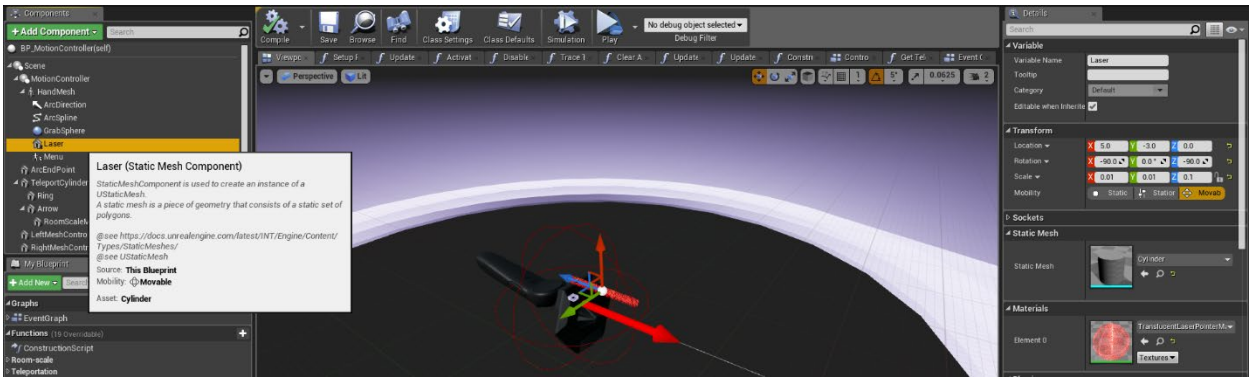
What if we don’t want to return to the menu to select our next destination? What if it is a short 100m jump away, that we want to teleport to it? Or, another floor with a better view? We’ve experimented with numerous options to develop a way to signify to users, that they can teleport to a location. In the end, we settled on needing to make it large enough to see from a distance, as well as a recognizable logo. So, we looked to the classic directional pin for inspiration.



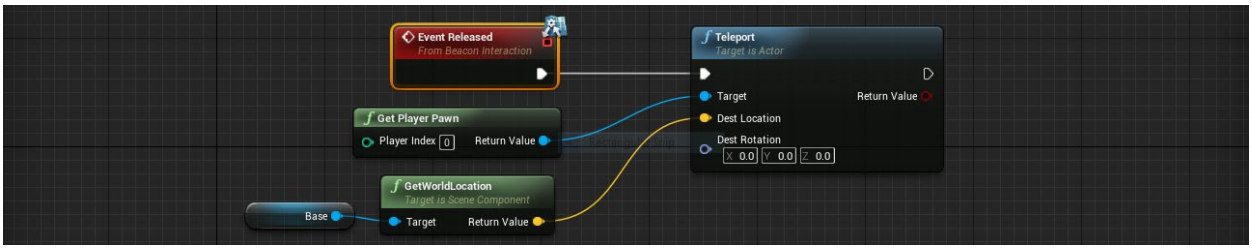
With these icons dotted around the zones, it permits users to quickly navigate large expanses without having to walk or slowly teleport towards it. They also signify to users that there is a point of interest in the distance, such as balcony views or a vantage point.

Teleport to Pin

Part of what we want is to direct a pointer at the icon and for it teleport us to the location of the pin. This starts with creating a laser in our Motion Controller Blue Print, and then having it interact with the icon.



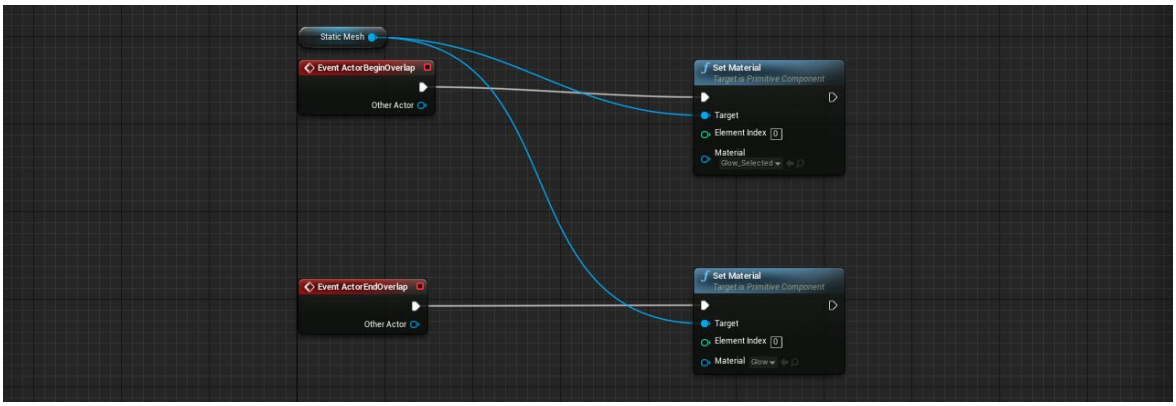
Once set up, it is then a point to have the beacon react to the laser, when it is pressed and released. Then teleports the user to a predesignated base, which is loaded into the blueprint.



Animate or glow

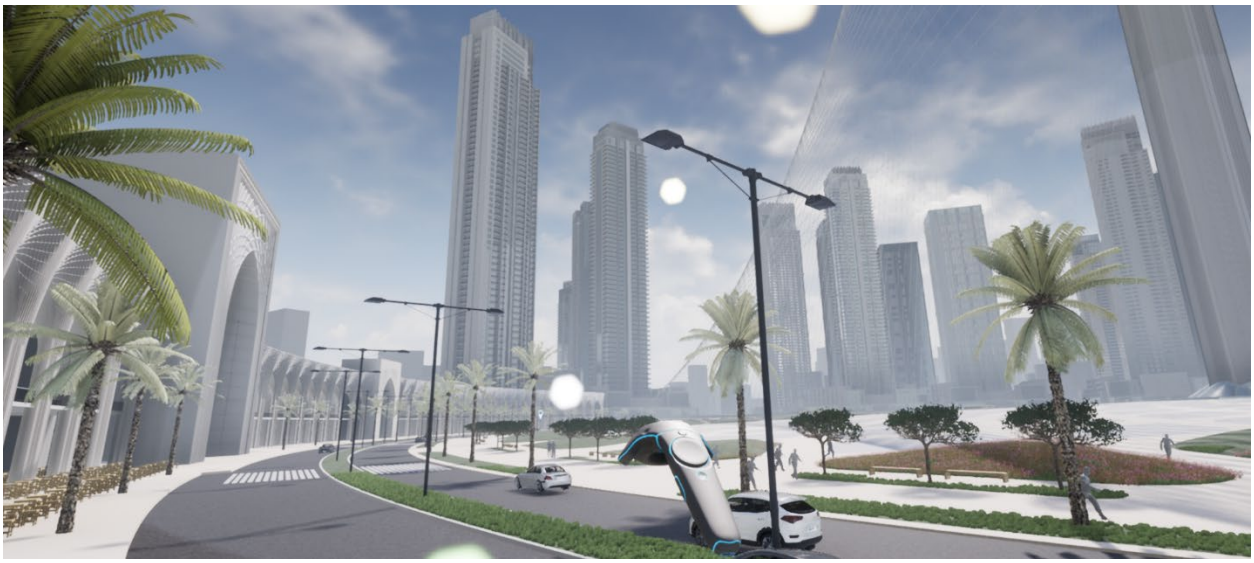
Additionally, we found it helpfully if your object is easily distinguishable from your scene. This can be achieved simply by choosing to add a special property to the static mesh, such as a glow or a noise. In the illustration below, you can see we created our icon with a blue glow, what's not shown is it was also spinning slowly.

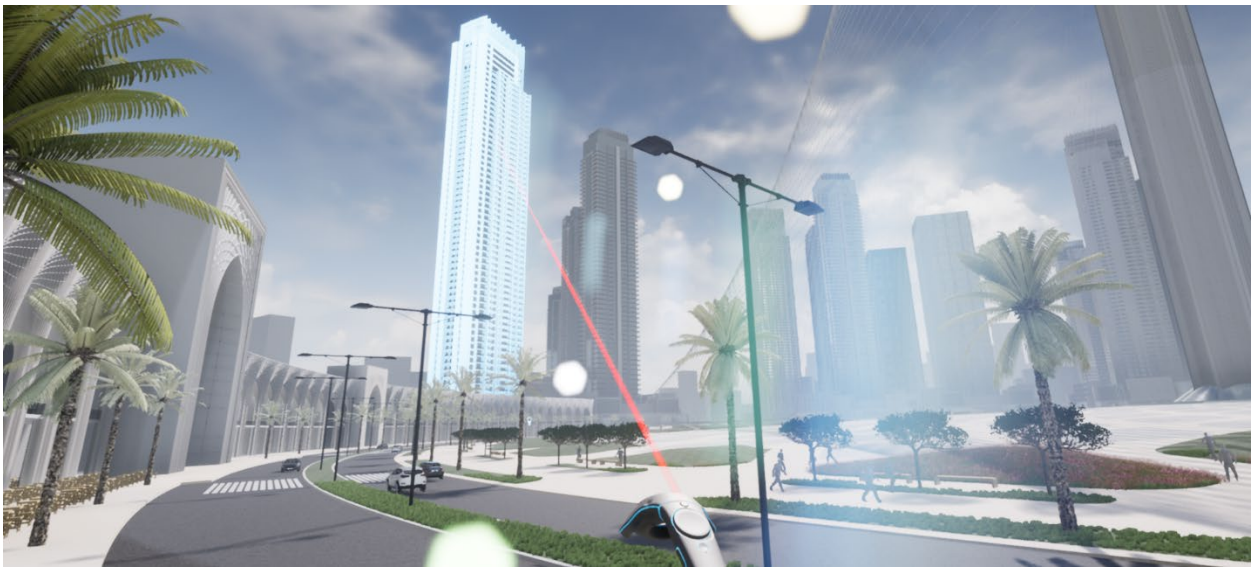
With our laser pointed sorted, the next task was to have users clearly identify when the icon is targeted by the laser. As accuracy with the motion controllers, can be troublesome for some, the easiest way to show this, is a simple material change upon overlapping with the laser. See Below:



Move around your masterplan seamlessly.

A complication we discovered when trying to show certain points of interest to the client, is the ability to achieve the exact desired view. For in VR it is all too easy to wander in a wrong direction, or to look at something from a poor vantage point. Another challenge is, although the location markers work for short range teleportation, they aren't the ideal option to show the client a view of a tower in the distance. When neither returning to the zone selection or flying over, are appropriate options; we looked at using entire buildings as hotspots for teleportation instead.

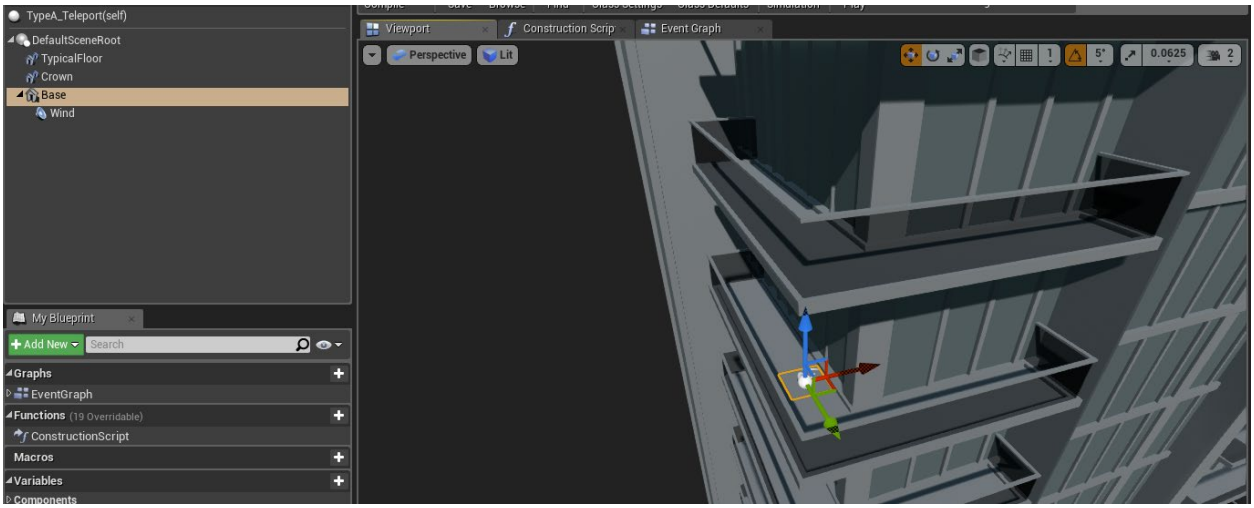




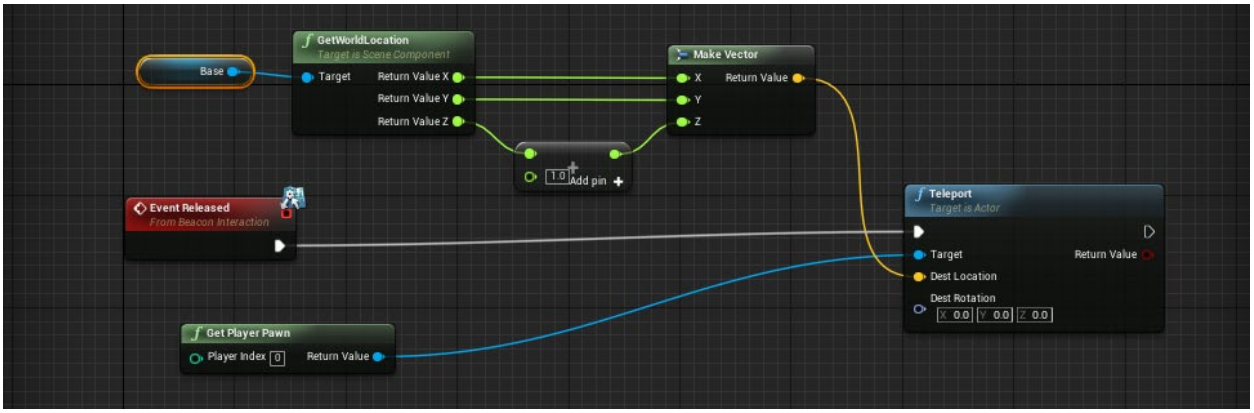
This process involves firing an indicative red laser out of the motion controller and directing it at the surrounding area. As we only want it to interact with certain elements, this laser does nothing if accidentally pressed or pointed at undesirable objects. However, when hovering over a building (our point of interest) it lights up, thus letting the user know there's something worthy of attention there. Releasing the trigger will then teleport them, to the exact position that we want them to view from that tower. This is like the teleport pin discussed prior, with activating static meshes to interact with the laser, and also setting up a reaction when the trigger for the laser is released.

Teleport Script

A tower is massive, with numerous views and vantage points and it would be complex and confusing for users to make their own way up. So, this idea is rooted in the original discussion of refining the user experience and narration of your story. In this instance, we wish the client to view something that is very important to them, which is the view and visibility from the balcony, over the main space. For this there is only one place, that we really need the client to go. Within our blueprint we create a simple static mesh plane and place it on the floor we want to teleport to; let's say it is on the 39th floor.



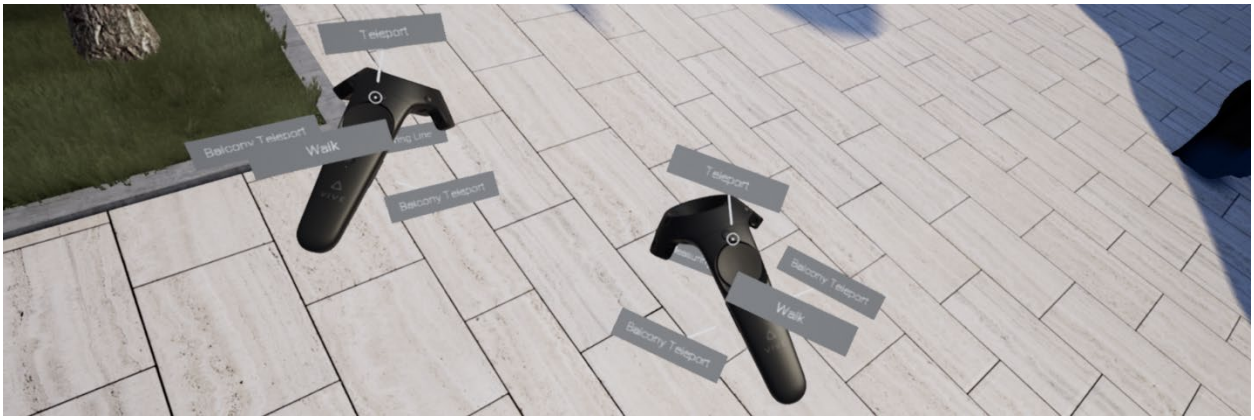
Now similar to the Pin, we interact with the laser and upon release of the trigger button (our designated laser button), it teleports us directly to the base point that we specified.



Building Glow

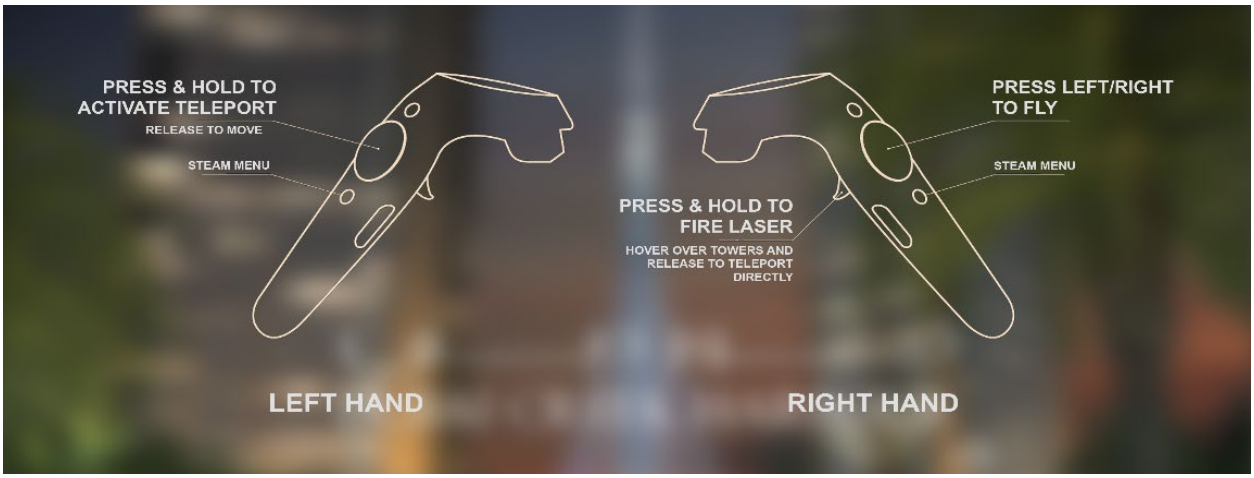
There are several methods to demonstrate points of interest in landmarks, around your masterplan. We find the glowing effect to be a great solution, once again grounded in what is commonly experienced in games. The glowing is the least intrusive way, to alert users to the fact the object they are selecting is significant. We have found that having messages appear on the screen, or programming a series of buttons to teleport across the scene is an unnecessary burden on users concentration and ultimately detracts from the experience.

Overly complicated functions and applications results in the experience being less fun and less intuitive. In one simulation that we tried, we used the series of buttons on the motion controller to guide the user to different views. However, those who have tried to explain to new users where the ‘grip buttons’ are on an HTC Vive are, will know the frustration of trying to tell a seamless story.



*wrong

Therefore, we keep the controls as straightforward as possible for the users. It’s rare in master planning VR, that you’re picking up and throwing furniture about, opening doors or pressing buttons. So, keeping it simple to navigate, will make the experience a lot more consistent which ultimately aids your goal of the VR experience being easy and educational.

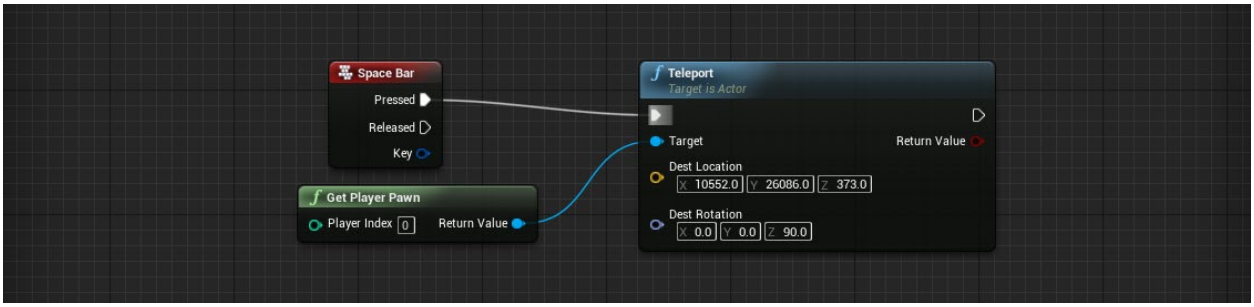


In returning to our menu, we created a simple sheet, which highlights the functions the user will need. For example, short range teleport and laser.

So, keep it simple. Remember not everyone is a gamer, or even comfortable with technology.

Always have a Backup Plan

And finally, regardless of how skillfully you programmed the user experience, and how aptly you controlled the story and narrative, sometimes things go wrong. The client will probably find that one place to get themselves trapped or stuck. Always have a backup to return the user to a recognizable point in the map.



With these navigation tips, you will be able to craft an experience over vast distances and areas. Keep in mind that the users may be unfamiliar with the project, or inexperienced with VR. We ran numerous tests, by involving random people from our office, in each aspect of our experience. If we found they struggled to understand an element, or to figure it out, we would go back and reevaluate our approach.

*****Repeat Learning Outcome 03 until your story/narrative is complete*****

Learning Outcome 04

Create using instances, models and entourage within Unreal for optimal performance in large scale Urban Projects.

Finally, we arrive with a fully optimized model with a well narrated story through your Masterplan. Now we want to go one step further and add some life into your model. As one of the main benefits in unreal over other VR plugin software's is its ability to handle animations and movement and the quality and detail you can achieve. For this learning outcome we are going to break it down into a number of different areas.

Entourage & Levels of detail (LODS)

In large master planning models, it's essential to note that you cannot load unlimited elements into your scene, without it having a direct effect on the frame rate and experience. You are required to be very selective with the entourage that you chose to put in your scene. Customarily we look more into simple entourage elements, such as benches, lamp posts, and a small scattering of trees.

Realism Vs Performance

This is one of the hardest things to weigh up when creating a good VR experience, as we want to populate the experience with as much lifestyle as possible, however want a smooth and crisp experience for the user.

White Model Massing
Level Size 1000mx1000m
Basic Landscape, Animated People and Cars.
Avg Frame Rate: 90-100fps

Full Textured Model
Fully loaded scene, Landscape, Animated People with Skin and cars
Avg Frame Rate: 30-50fps

The above examples feature two contrasting experiences. Remember to use your judgement of what best achieves your desired narrative. However, in most examples we have found a minimalistic model that runs smoothly has a greater effect on the experiences compared to one that has a lower frame rate but with greater detail. In master planning and architecture there are more momentous objectives to convey, so don't detract focus by filling your experiences with expendable details.

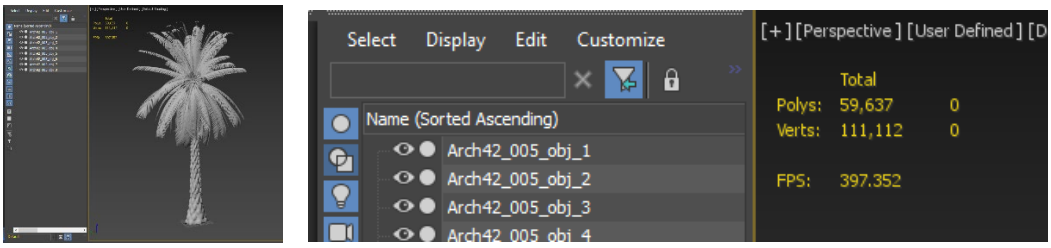
LOD and VR scaling Issues

What is LOD for a start? It stands for Level of Detail. The gamers amongst us might have noticed this happening in games, but most of us are oblivious to the changes occurring around us in games. What we're referring to is when a model's complexity and detail changes, depending on the user proximity. The closer you are to the model, the clearer it appears; step further and the detail will diminish. What this achieves is a reduction to the load on the model when you see multiple objects in the distance. In Master Planning this function is highly necessary as you can frequently see for many miles into the distance.

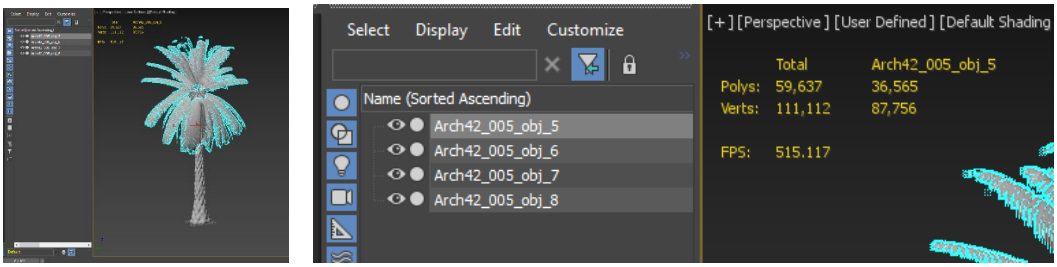
In this instance we wish to have entourage in our scene, let's look how we can bring the weight of the model down but still retain a good amount of realism within the models. In this example we will look at Trees. In particular, we will look at our Palm trees in our Dubai Project.

Create LOD Models

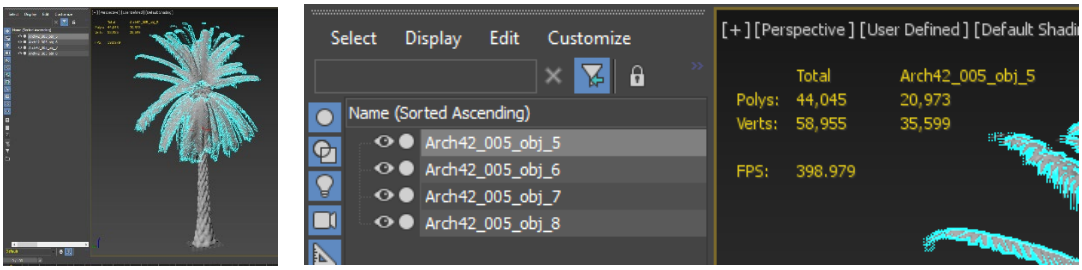
How do we do this? There are two ways to create LOD Models, one is within 3D Studio Max the other is within Unreal. This involves creating multiple versions of your component at different scales of complexity. Firstly, we will look at 3DS Max:



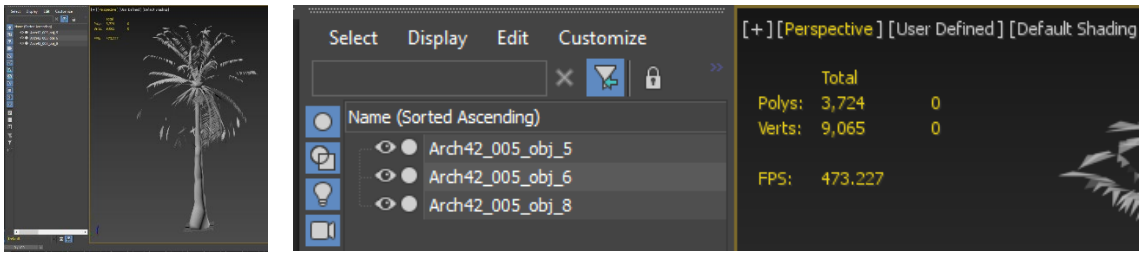
Here we have a Palm Tree model. Now, models you download or buy from render packs typically have a high poly count. Why? they're built for quality renders not for optimization. The above example has a whopping 60,000polys Per Tree. In our model if we have 50 Trees, in view, that's 3,000,000 polys of just trees. On inspection we can identify two things we are going to fix.



To start with there are multiple objects within this tree component. With LOD modeling we want the entire model to be a single object. Secondly, there's a drastic difference between these elements in weight. The palm branches are heavier than the trunk at 36,000 polys. Since we won't ever be positioned directly next to this tree, within the experience, we can immediately reduce its count. Our intention with LODS is to create gradual steps to simplicity, so they change slowly.



By applying a simple Multires Modifier, as seen in LO1, we can see that the polys in the leaves have been reduced by almost half. However, upon inspection there's only been a slight change to the quality of the model. Depending on the model's proximity to the user we may want a less dramatic drop, but we can save this in this instance as 'LOD 02'. Before saving we apply the same effect to the other parts of the tree.

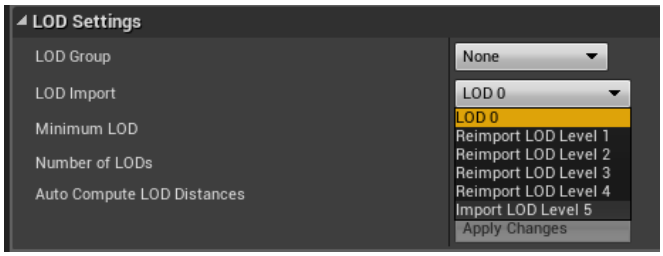


Let's keep going to LOD 04. By this point the tree leaves are minimal but still visible. These will be over 200m away and therefore not very visible. The size of LOD 04 is now just under 4000 polys, meaning that a model with 50 Trees is now 200,000 polys. This is a great improvement to the optimization of our scene. Before we export these out we combine the objects.

Manage LOD

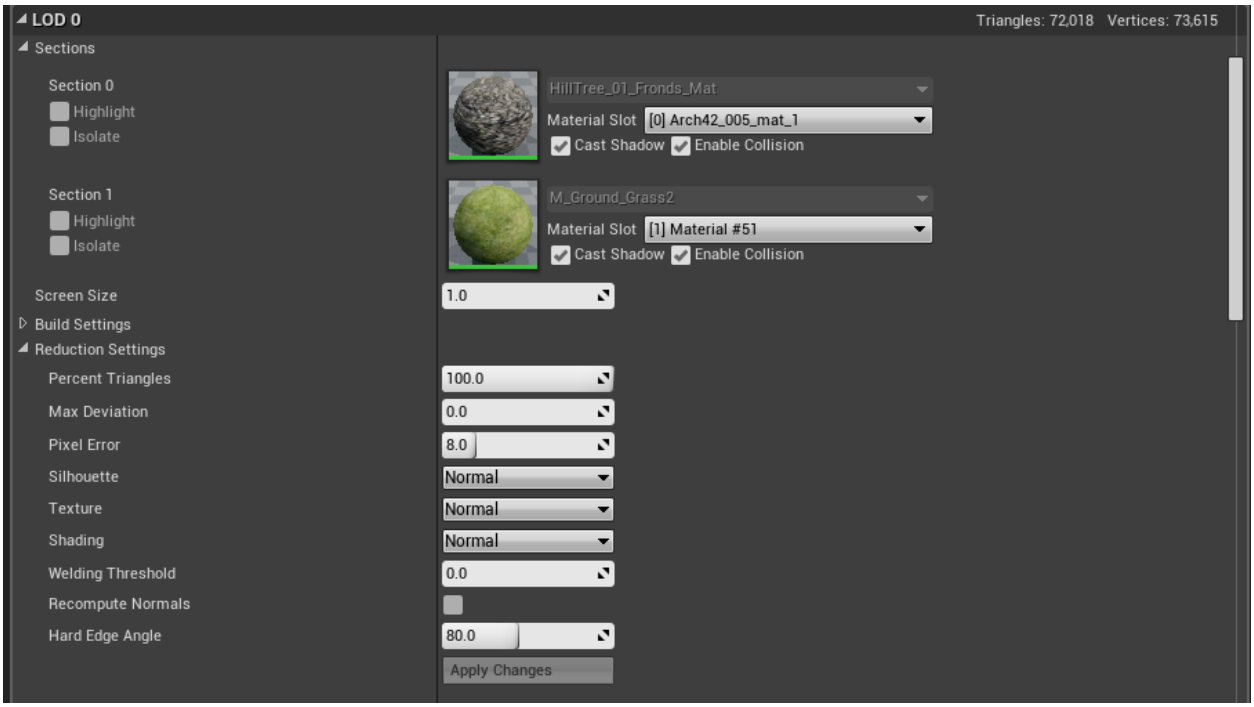
Now we have our models we jump into Unreal. Being single objects, these can be imported as .fbx files for simplicity. Below we can see our LOD trees, ranging from high level detail of 70,000 Polys down to 4,500 Polys.

First of all, we bring in our LOD01 Model, and within the static Mesh editor in our detail panel we can see our LOD Settings. By clicking on the drop down of 'LOD Import' you will see an option to Import LOD. Now we locate the file on our drive and add it to the list. We now rinse and repeat with our other models.

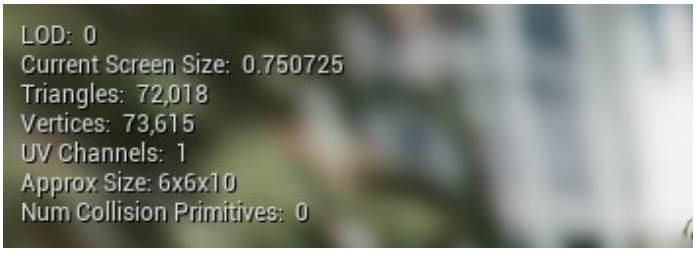


The important part now is the option at the bottom of the Settings which says 'Compute LOD Distance'. By checking this option, Unreal will automatically compute the distances that the LOD models will change from the user in the experience. What we would

recommend is viewing this very carefully as sometimes (especially in VR) these distances can be dramatically off. When we preview it in Game Mode, they disappear as they should; however, in VR you can be 5 foot away and our LOD4 model will be the one visible.



To manually control this under the LOD type there is an option called Screen size. Altering this will help you adjust how far away each LOD will be when it changes. Use this in conjunction with the view port information however always remember to view it in VR mode before progressing.



Landscaping & Culling Distances

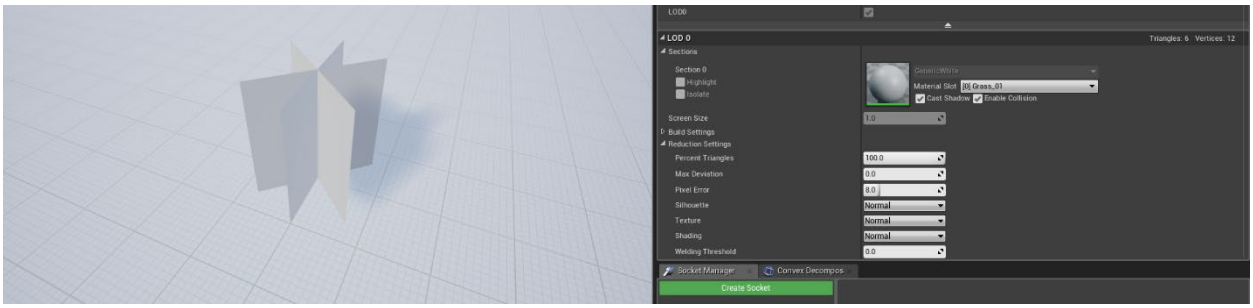
As mentioned in previous Learning Outcomes the best way to optimize your model is to remove foliage due to its high opacity rates and complex geometry. However, one of the best ways to add a bit of flare to your scene is with a little foliage. So, where's the middle ground? In our experience we do a couple of things. Firstly, avoid using certain landscape packages on the internet. The majority are built to provide for small scenes and with a heavy intent on making it

as realistic as possible. They will add to your poly count, and consequently subtract from your performance. Next, in instances where you do want some grass, or heather, or perhaps shrubs. Make your own. This tactic works well in models with less detail. Although in fully textured models it's not advisable, for they are not modeled to the standard of the other components.

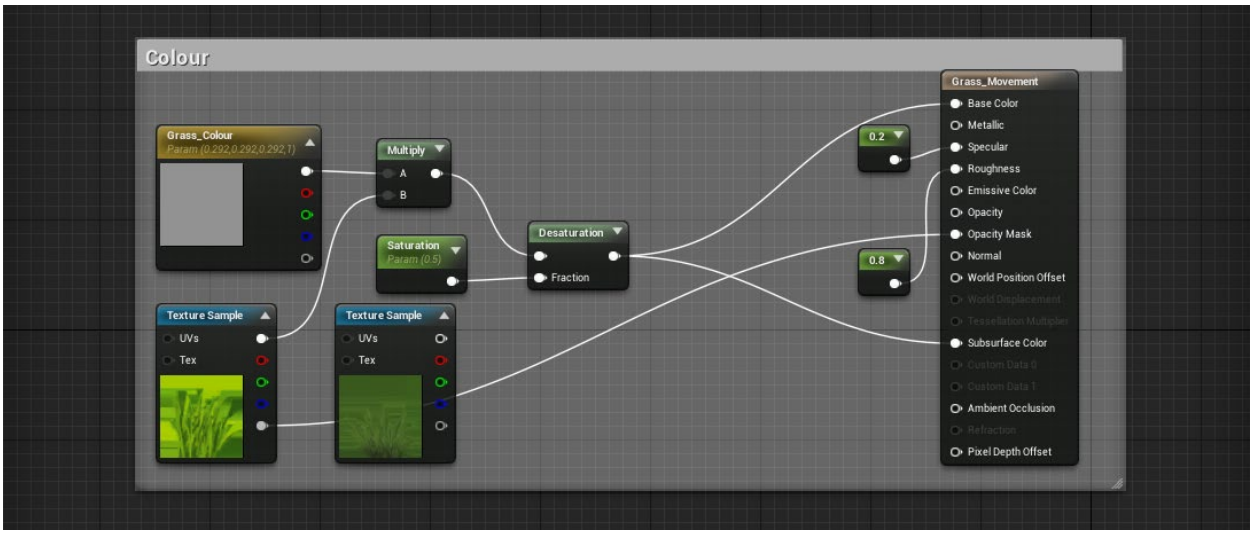
Grass & Greenery

When aiming to optimize your model it is possible to add more detail and maintain performance. We can add landscaping and grass, with minimal effect on the overall performance, by dropping its quality. Whether you're using a created grass or a low poly image, remember that the grass is far away, and not a main focus, so don't be afraid to reduce the quality.

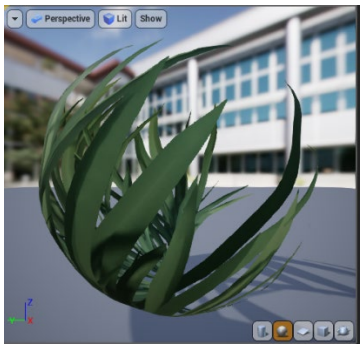
this is an 8kb grass png. It wont win any awards for most photogenic grass but it'll do for this example. Second is to create a series of planes intersecting in Max and bring it into Unreal as per below. This is to give the grass a 3D feel as it will be printed on a series of faces so the grass image will be seen from all angles.

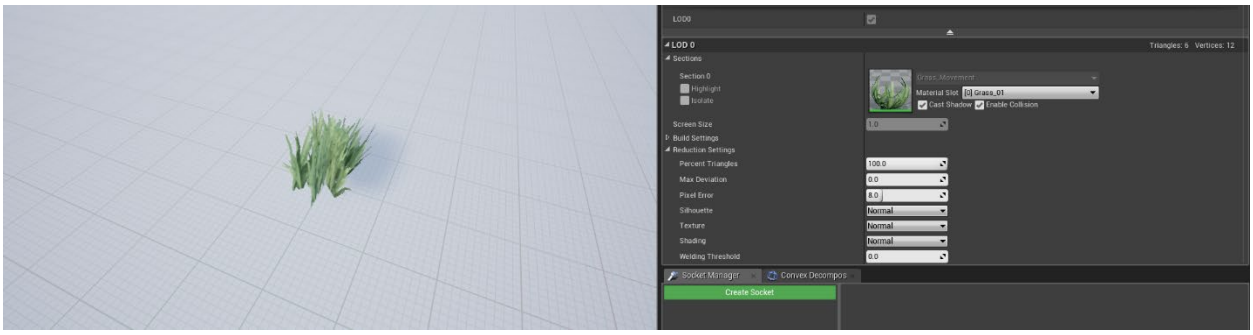


Next, we import our image as a texture and apply some simple modifications to have it the color and saturation we want. The saturation is there to remove the bright colored glare that images tend to think grass has, in Dubai this is not the case. We also use the image as an opacity mask to allow the background to disappear.

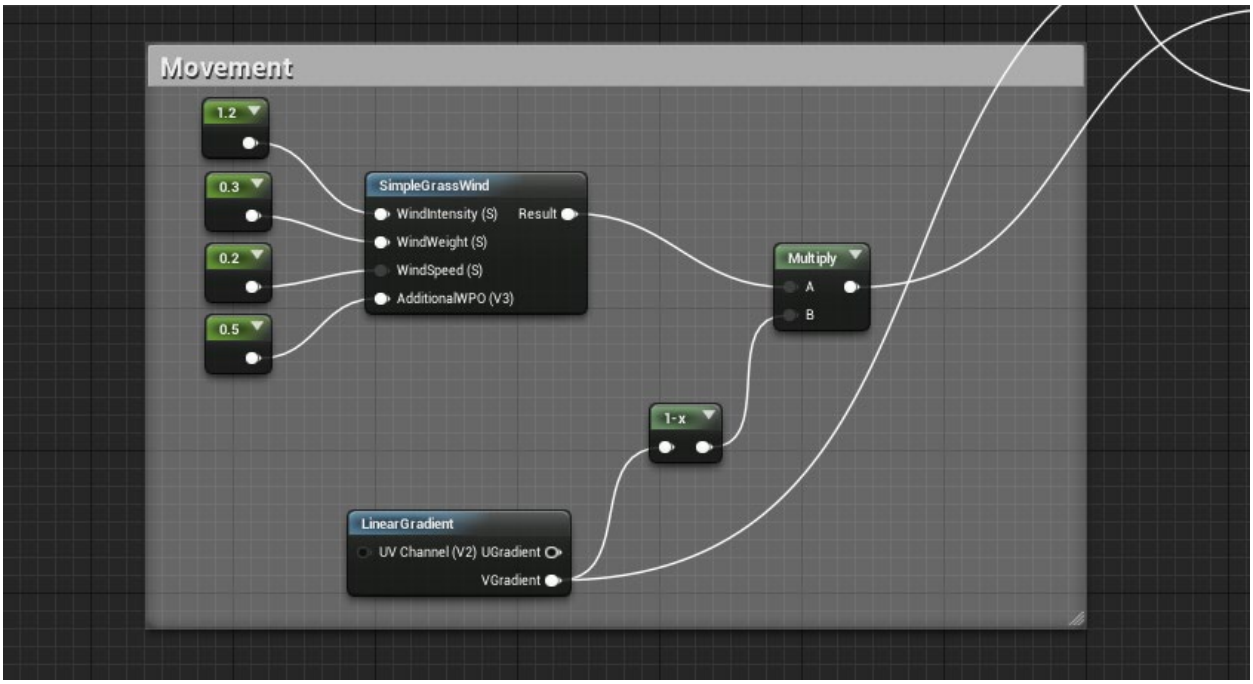


And now we have a simple grass texture





This can be applied to all the faces of the plane object we created earlier, and we now have a small patch of grass. There are other minor things you can do to make it slightly more realistic, such as give it movement in the wind. This is a basic modification which involves making the grass move on a gradient (to keep the base still whilst having the tips blow in the wind). This however is not completely necessary, and any animations can be a draw on performance. Although once we start culling you'll find this is an expense we can afford.



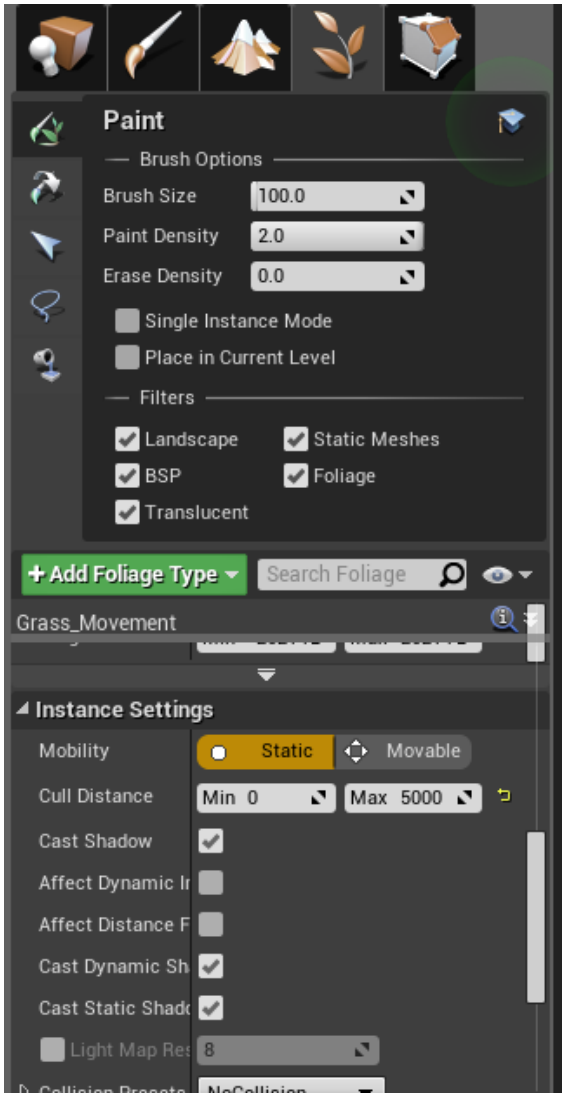
The same techniques can be adapted for a variety of different foliage types, small bushes, heather or ferns.

Culling Distances

For those who are unaware what Culling is, again it may be something gamers amongst us will have seen before but never knew the correct term.

It is an amazing tool for improving performances. Linking back to the concept that Unreal only draws what it sees; if it's not in sight, then it won't be processed. But what we want to achieve is a subtle disappearance, without the user noticing. This is accomplished by setting a distance for which the objects will disappear.

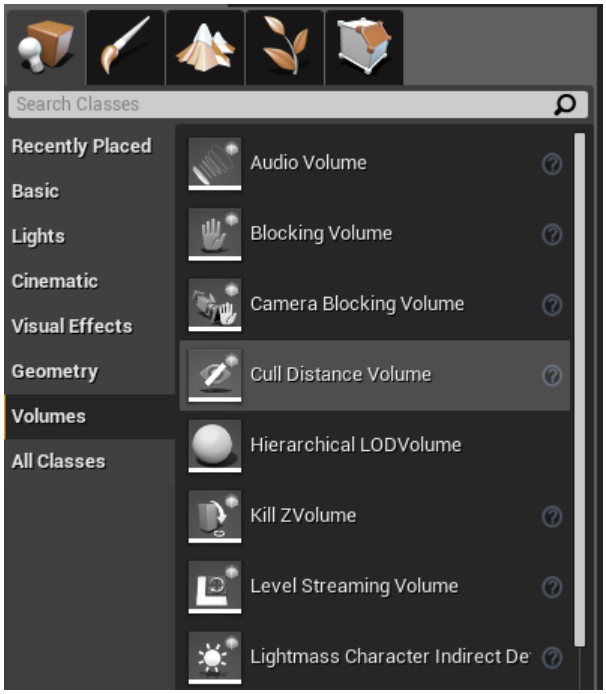
In the instance above you can see that the grass textures disappear in the distance. So, lets first look at landscaping culling since we've just created some lovely grass. For Landscaping Culling is easy, each Landscape element has parameters you can play with that will affect all instances of it in the project. In the landscape drop down menu you will see very clearly under instance settings an option called Cull Distance. Which allows you to set both a min and maximum value. Similar to the LODS this has to be tested in VR to find the distance that works.



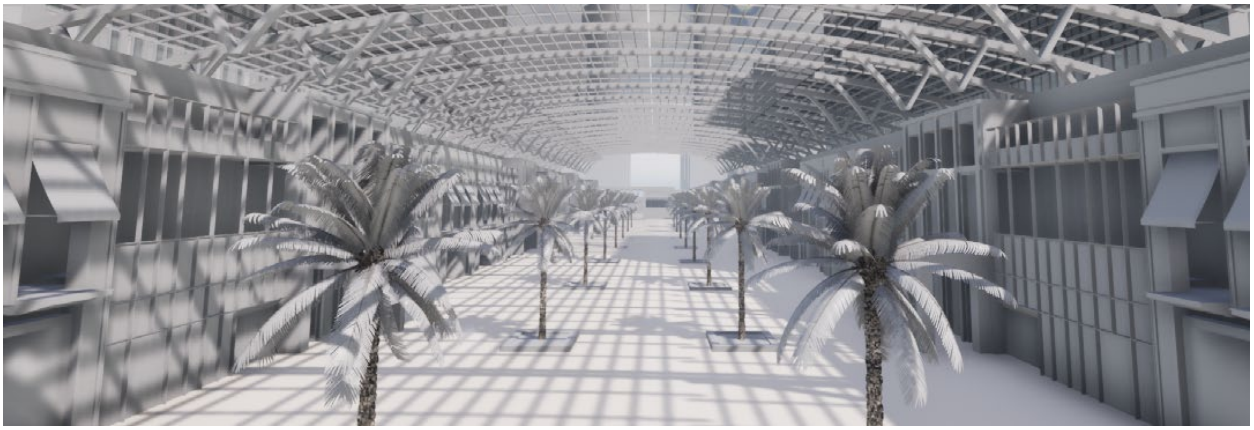
Culling Volumes

There's a second way we can work with Culling and that's through the use of Culling Volumes. This can work with buildings, towers, facades, and even benches and dust bin entourage. Located in your Modes Panel under volumes is Cull Distance Volume. Much like the other volumes in this set this only works on the

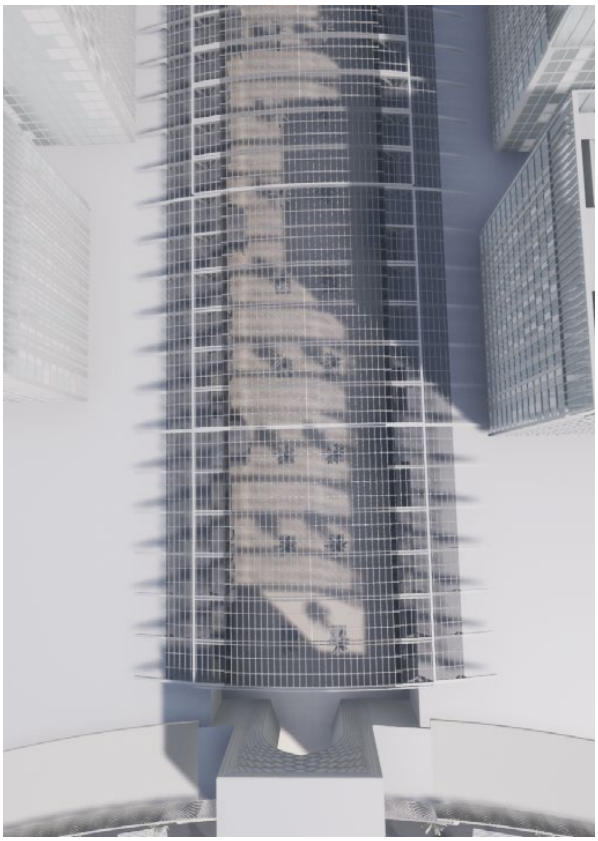
objects within the volumes, meaning that it will only Cull Actors and meshes within its bounds.

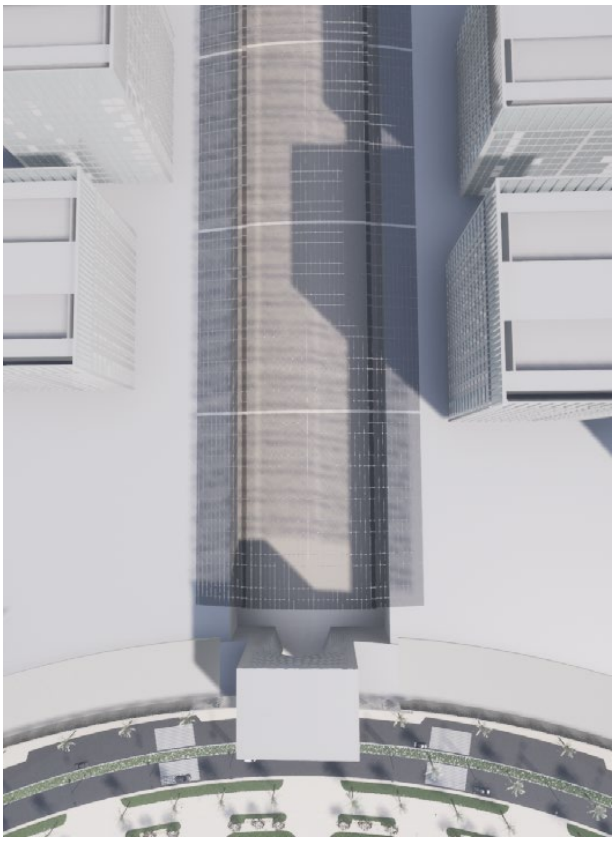


Our masterplan features a mall. For this we have brought in a generic mall face, and from certain ground planes it will be visible. However, you will notice it is heavy in geometry. So, when I am a suitable distance away, I do not want it to render this.



Thus we encase this entire mall section in a cull volume and establish the cull distances. In this illustration, I will keep it simple; starting with a rule by setting 1 array with a cull distance of zero, with a hypothetical size larger than all your elements. This is a requirement to make the cull volume work. Secondly, provide a distance and size of model that will be affected by that cull distance. I implemented two such distances, one for the palms and another for the facades.





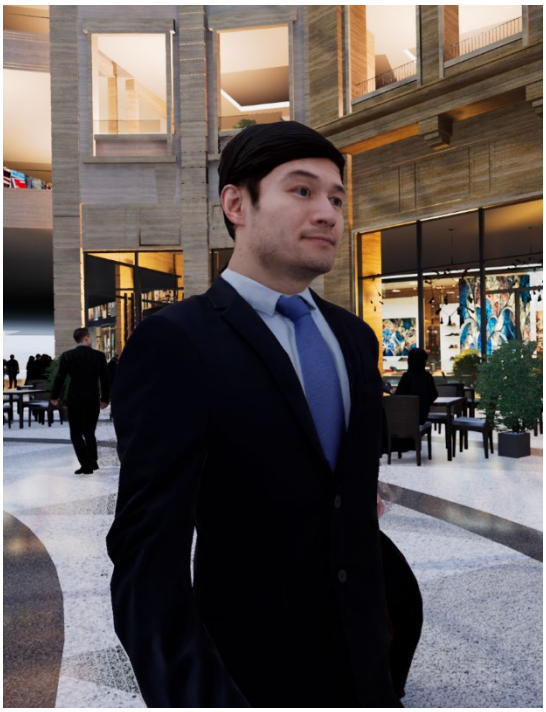
In the examples above you'll see the camera offset at two distances, and how everything internally within the Mall disappears from afar. This is the result of what culling does. This is helpful in a variety of uses in Masterplans and Urban Planning VR such as:

- Buildings far in the distance don't need to be visible
- Interiors and denser areas with more detail can be controlled by distance proxy.
- You choose what the client sees from each vantage point. This can include cars, people, skylights, interiors and landscape elements.

  In short, it's a productive tool for making sure your model runs smoother with a higher level of detail and modeling in your scene.

AI People

One of the favorable aspects of Unreal over other products on the marketplace such as Enscape or Lumion, is the ability to animate the scene, and bring life to an otherwise abandoned empty experience. A.I. people also are a great mechanism to emphasis scale in VR, a reference point that otherwise could seem skewed. Similar to what we've seen previously, we had to decide in the early planning stages between fully textured people or basic.

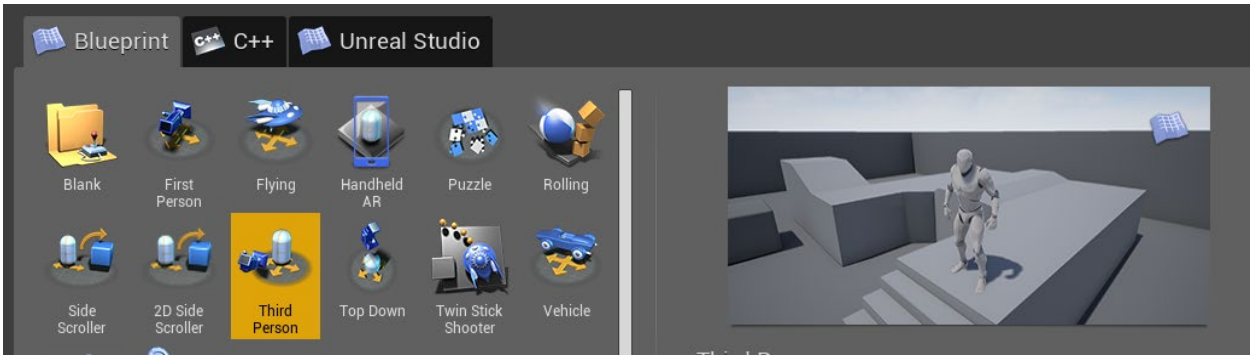


The above examples the one of the left showcase an Unreal 3DS Max plugin called Anima. In which you can input a variety of characters fully textured, and animate them doing various

different things such as walking, standing, sitting or laughing. The other is a typical 3DS Max skin, that we brought into unreal and created its movements ourselves. Both of these have been used in different projects and received a variety of feedback and results. We generally favor utilizing the white massing models of people, for a number of reasons:

- We can animate them using very simple steps in Unreal;
- Users reaction to fully textured is varied, from favorable to alarming;
- It keeps focus on the elements of the model you want to showcase;
- We can control the poly counts on our own models and even apply LOD's to our characters.
- Downloaded content can be very heavy in addition to the movements being very static and on a loop.

If you are new to Unreal, or unfamiliar with Third Person Movement and animation, there is a great template to assist you to develop the knowledge required.

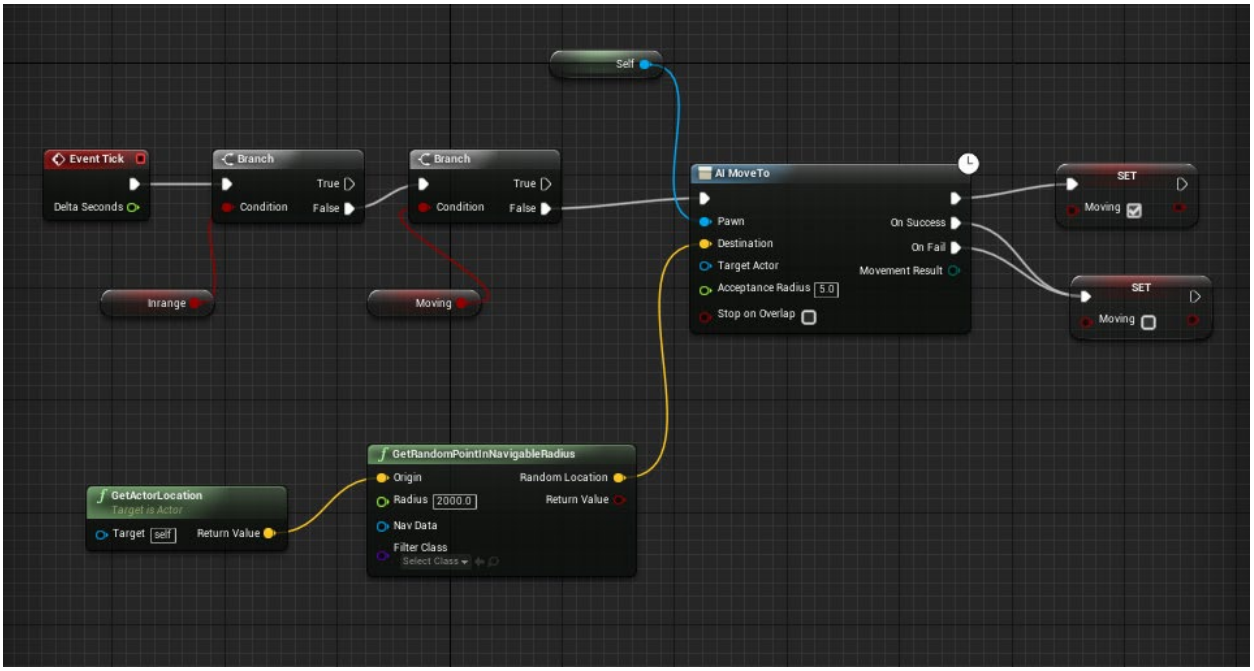


From this point on, we require some basic effort to start in character animation. For the skins template provides you access to Unreals robot skin, which is a great character skin. However, when your scene has hundreds of these robots roaming around, it can be a little like something from iRobot, instead of an Urban Masterplan.

Simple Roaming Animation

As previously stated, the purpose of people in our animation is to accord the experience an aspect of life. They permit us an understanding of the size of the spaces, and enable the user to comprehend the scale within the VR. In this instance we will look to animate for the purpose of scale and lifestyle instead of dedicated paths or reacting off data sets.

We want the character not to walk on a loop, from point A to Point B, but instead wander randomly about our scene. This allows us to install them, and then leave them to their own devices, as we focus on other aspects.



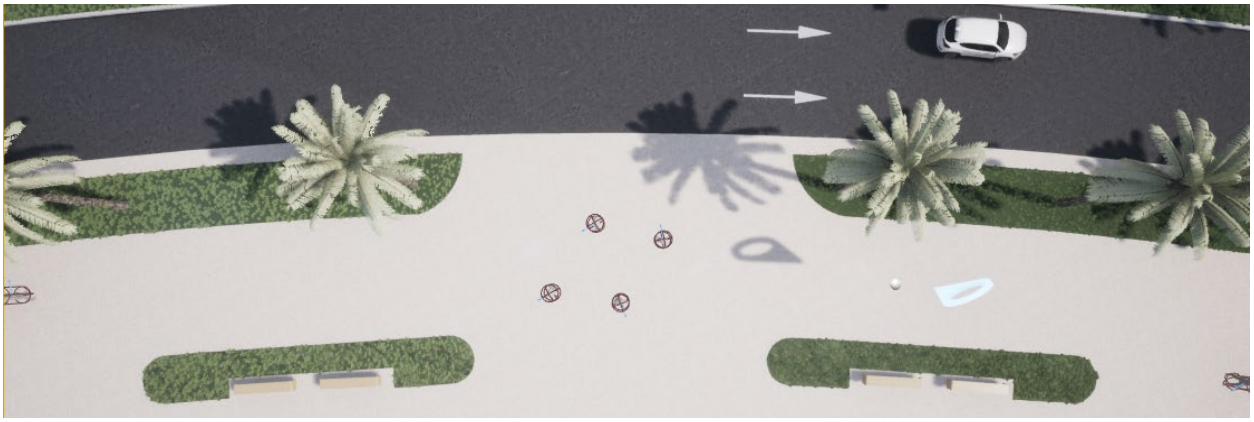
In this character blue print above, we are setting up for the character to draw a radius of a set distance. In this instance we used the value of 2000, which may be a little high. What we are telling the character is to take a random point, from within this radius, and simply move to that location. Upon arriving at the point, the character is told to simply do the same thing again and so the process continues indefinitely. If you add a collision to

your character and to yourself, in the situation when an animated person closely approaches you, they'll stop and commence the process anew. The result is very organic, with a large number of people in an open space, appearing to stop, engage, and examine their environment.

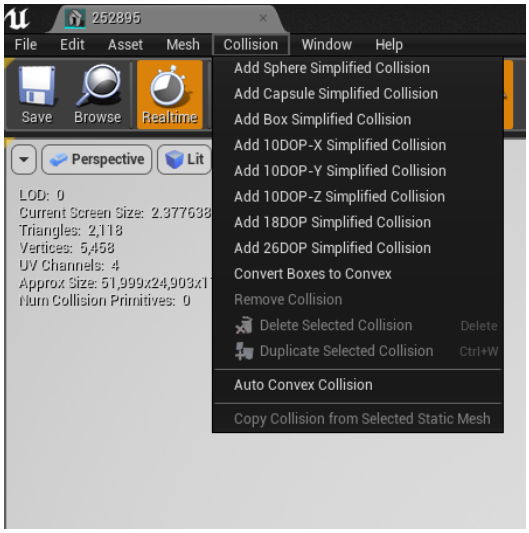
Collisions.

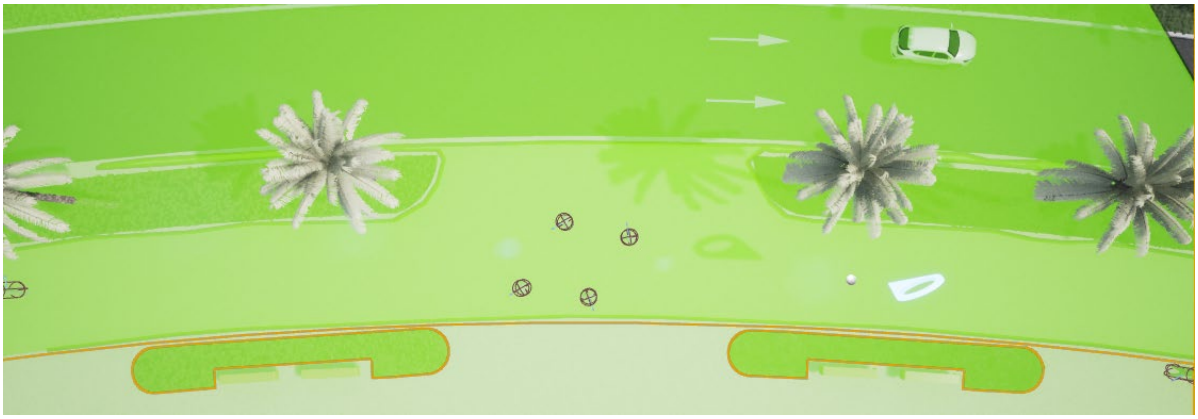
This requires you to be attentive and set up boundaries. For you do not want your people to be overflowing on the roads or walking across flowerbeds. This requires a great deal of alterations to the collision map and bounding boxes. Let's examine some examples of this.

Within a typical static mesh, we created for the pavement areas or walkable areas we can go into the settings and set up some basic collisions



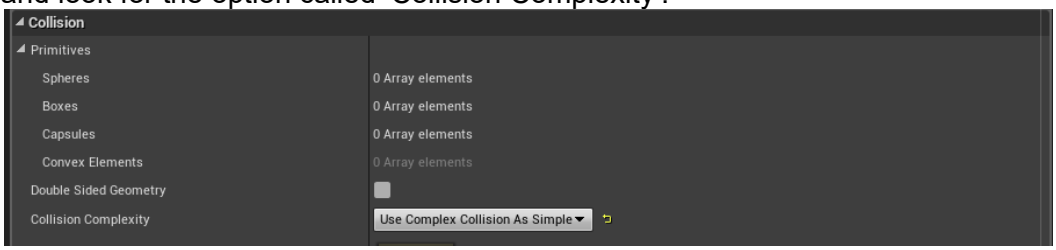
Here we see options for to add collision maps to our objects. However, if we apply a typical sphere or a box collision to our model it makes everything that intersects with that a surface too.



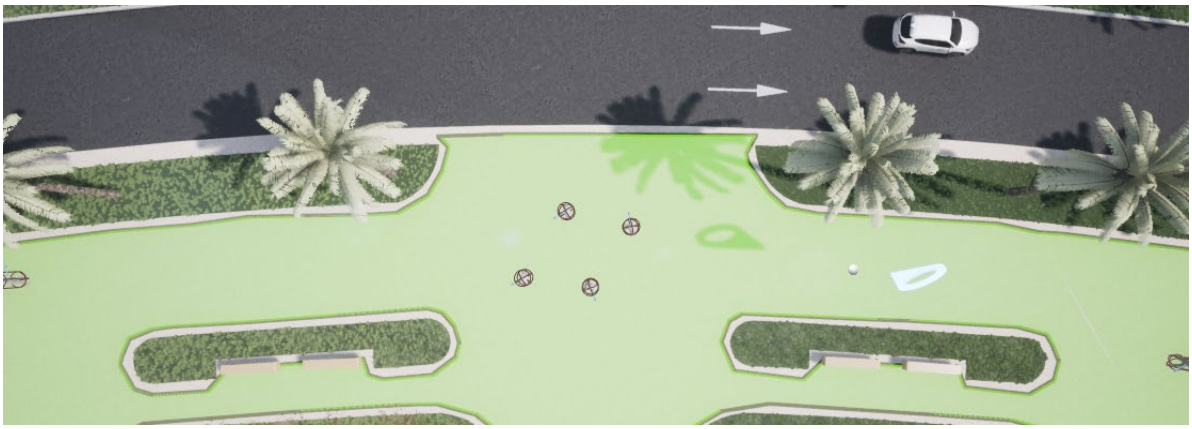


In our scene if we hit ‘P’ it’ll show us our collision map within the bounding box. Because we created a simple sphere collision its not highlighting everything within the sphere regardless if it’s a separate static mesh.

Instead what we do is within the Static mesh editor located in the collision drop down and look for the option called ‘Collision Complexity’.



In the drop down, menu select the option of ‘Use Complex Collision as Simple. What this does is draws a more accurate Collision map based on the geometry.



Now what we see is a Collision map that wraps around our geometry and will now stop our character's from wandering onto the roads or into our plants.

This also requires us to go back into our 3D Max model and detach objects based on its uses, so we can create objects based on character movement. Its at this point we refer back to Learning Outcome 02 about the tips and tricks to import your model and update it via Datasmith.

There are so many different things you can add to your scene to help it feel more alive, however always refer to your narrative and the client and project to help you steer it in the right direction. Many times, we can get carried away with all the things we can do and never stop and think what we should do. As we saw before this is true in the controls in VR, but it is especially true in the entourage and landscape elements within LO4.

*****Repeat until you have an amazing Urban Masterplan VR Experience*****

Summary

Developing Large Urban Designs into Virtual Reality.

The developmental differences of small-scale projects compared to large scale VR projects should now be clear. These distinctions emerge at the start of a project and should be acknowledged before the design even enters the VR space. Devoting time on your model to optimize it from the beginning, will save you at the later stages from being restricted or overloading your program. This comes with care and attention when modeling within 3D Studio Max, utilizing the optimization tools within and managing your team's workflow and files.

Narrative is also essential to your project. The story will be there to guide the user/client within your finalized experience. Its objective is to enable them to find the experience as easy as possible, but also to maximize the benefits of having developed the project in VR. You are able to showcase, to your audience, the best of your urban planning in size, scale, and views. This is where the Unreal Engine's use comes into play, with its ability to customize every aspect of the design, and the experience is what will help guide you to the right experience with your client.

Lastly, in populating the scene you are able to surpass their expectations. You can create for the user an experience which they will remember. A major recommendation and advice is to test your model and experience with multiple people and scenarios. Experience is the key and from every interaction you should be developing and working on making the user experience friendlier.

The components we have covered in this class, have emphasized the major processes you can take to optimize the experience and the model for urban design. I encourage you to continue to explore new ways to adapt and work with your models, for there are many ways to achieve these objectives. If you approach each aspect with the same attention and consideration, from the cleaning of your model to importing, narrating and populating, what you can accomplish in VR has no limits. As your experience improves, you will be able to branch out and examine multiple design options, integrate date sets, and solar studies.

But for now, have fun and explore your urban masterplan.