Advanced Graphics Report

My aim for the scene was to create something inspired by the Dystopian Cyberpunk genre. I felt it would be a great setting to take full advantage of different lighting and reflection settings with Unreal Engine, as well as skeletal meshes for people and volumetric fog. While collecting references images, I was particularly struck by an image of a marketplace with a body of water precariously flowing through the middle of it, while skyscrapers loomed over the scene in the distance. I interpreted this as a marketplace built on top of flood planes following the effects of climate change. Other common themes I identified from gathering references was general bad weather and lots of fog as well as the presence of Japanese style signage coming out of the walls as demonstrated in figure 2. Finally, some of the reference art would centre a piece of architecture that loomed over the people in the scene to emphasise the class struggle such as in figure 3. While I didn't plan to or end up using a China Town style gate the large object taking a central focus in the scene was an idea I ran with.

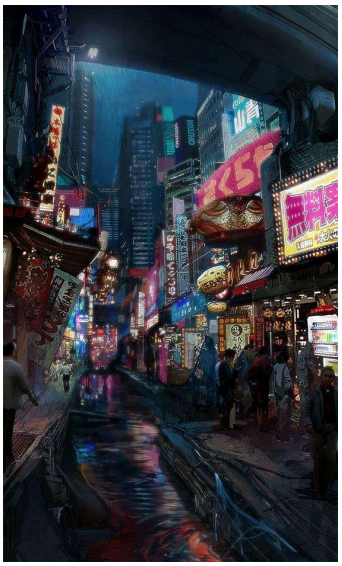


Figure 1 Night City from Neuromancer by Vincenzo Natali

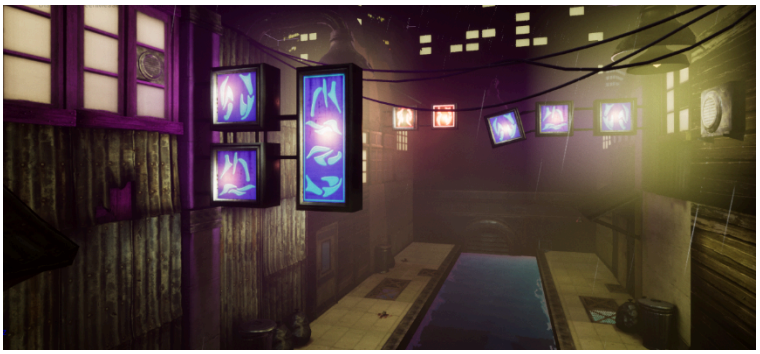
Since I started with the idea of something built on top of flooded land, I started with a terrain based on Huntington Beach, California a relatively low lying but developed area on the US west coast. I then used UE4's water plugin to create an ocean on top of it that would serve as the water source going through the scene and off into the distance. The main light source of the scene was then darkened to make the ocean look near black to both hide what was underwater and make it highly reflective while also implying some level of water pollution. The water combined with the implementation of Screen Space Reflections makes the scene feel more alive and dynamic while capturing the exact vibe I wanted from figure 1.

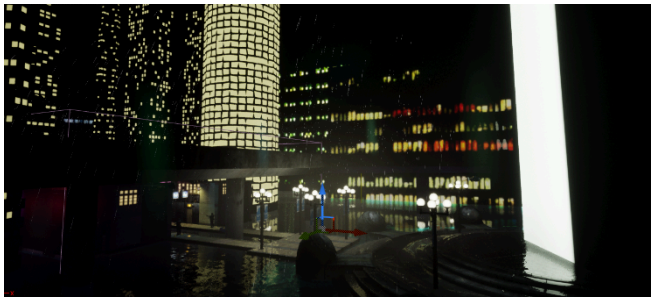
To get the scene looking and feeling wet I had two key

implementations. A particle system to simulate the rain and the use of more complex materials that are more reflective to simulate wetness. The “wet” materials, like the water itself, also interacts to great results with the screen space reflections present in project. To add to the feeling of bad weather, I also implemented various tarps throughout the scene that take advantage of vertex shaders to add a feeling of it also being very windy in the city.

Next I wanted to capture the way the references are mostly lit very artificially. Lots of point lights were combined with static meshes for the japanese style signs that also take advantage of high exposure and bloom to create lots of different coloured light sources making it similarly visually busy in the way some of the references were. Furthermore, the scene is littered with street lamps that give the scene an overall yellow tone, as yellow lighting has always felt especially artificial to me. The exposure and bloom from the signs also trigger a lens flare effect to give the scene a cinematic feeling. And to further add to the cinematic feeling the scene takes advantage of a light grain jitter effect and uses light chromatic aberration to add a sense of surreality. Finally, these lights are greater dispersed by volumetric fog effects which makes the colours more visibily fill up space. The fog also matches the weather from the references.

Figure 5 reflective materials on the floor5





Finally, to contrast with the poor slums that make up most of the scene there is a series of skyscrapers around the scene including a specifically futuristic looking skyscraper that acts as the center piece of the scene. The non-futuristic skyscrapers come out of the water, to imply they were built before the flooding and some are connected by an overpass. The overpass takes advantage of a blueprint which simulates a “spline mesh actor” to allow for it to curve. Two police officer skeletal meshes also serve as a physical barrier between the slums and the futuristic skyscraper and they take advantage of a pose asset.

PC Performance Analysis and Optimisation

For PC my optimization target was to hit a stable 60fps at a 1080p resolution on an upper-mid tier gaming desktop. Specifically, the system I ran the performance analysis on has an overclocked RTX 2070, a Ryzen 5 2600 6-core CPU which has been overclocked to 3900mhz and 16gb of DDR4 RAM. For the performance analysis system I mostly took advantage of

the Unreal Insights tool. I created a blueprint that iterates through cameras that covered areas of the scene I expected to have a high performance cost and traced a large list of commands for a couple seconds at each location based on a list of commands by from Epic Games.[1] The created trace asset was then be opened in Unreal Insights, where I primarily focused on analysing frame times, both overall and how long each individual aspect was taking.

For the first performance test, all raytracing capabilities were enabled as they were physically capable for the GPU in the system and would greatly enhance a scene that was so dependant on lighting and reflections for its aesthetic. This created a scene that performance analysis identified to be running at about 33ms on average but peaking at the more expensive camera at 43.3ms.

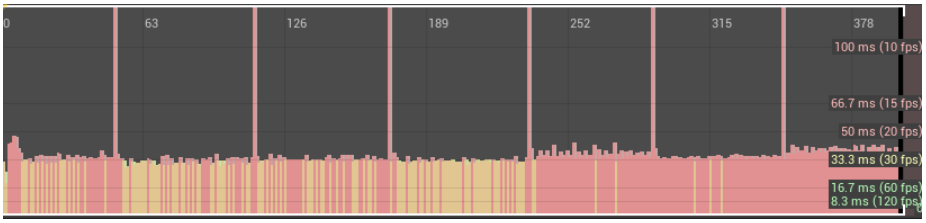
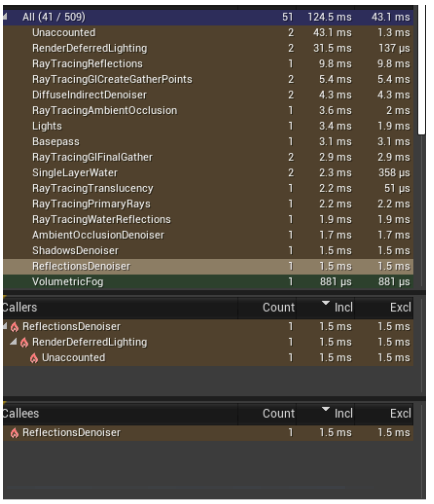


Figure 9 First Run frame time graph



This was due to a huge gpu bottleneck, primarily from the raytracing post process effects that were in use.

For the second run, I disabled all the raytraced settings, replacing ray traced reflections with screen space reflections and ray traced global illumination and ambient occlusion with their none ray traced equivalents. This resulted in scene that averaged out to the 23-24ms range but peaking as high as 29ms.

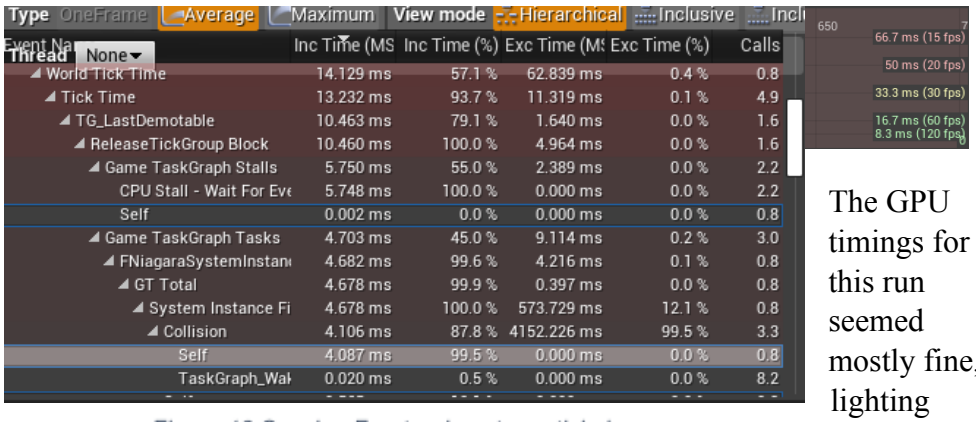


Figure 13 Session Frontend spots particle issue

The GPU timings for this run seemed mostly fine, lighting was expensive in parts caused by unintentionally overlapping stationary lights and Volumetric fog was a slight concern depending on the frame.

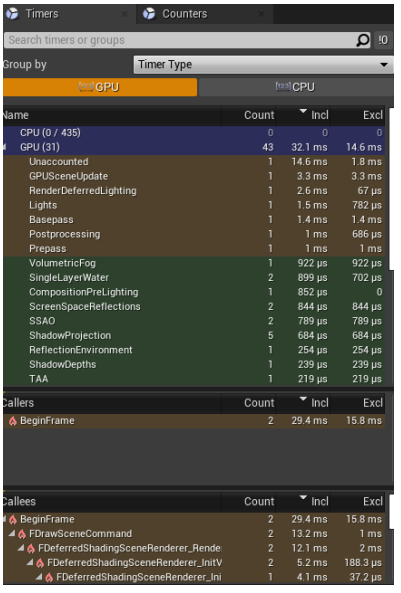


Figure 11 Second Run GPU times

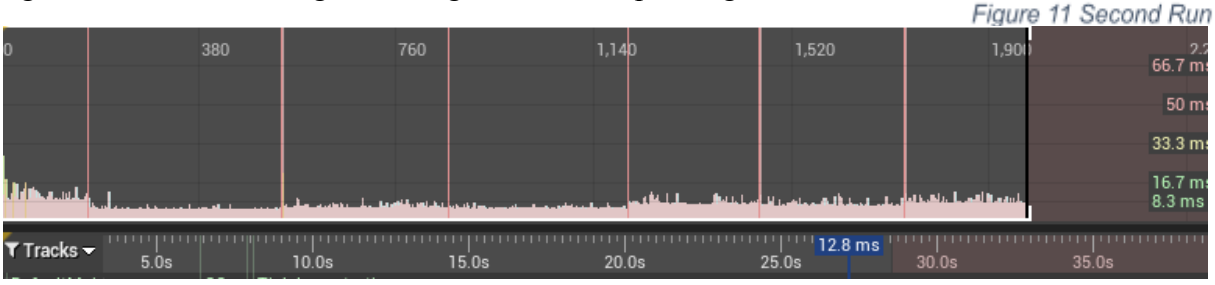


Figure 14 Third run, no particle System

However, the bigger issue was on the CPU side where it was spending large amounts of time waiting for tasks to complete. To help better diagnose this, I created a trace from just wandering around the scene that could be passed into the “Session Frontend” tool. Through this I identified an issue with the Niagara particle system that I was using for the rain causing a huge amount of problems. Upon entirely removing the particle system and replacing the stationary lights to static lights (as no objects in the scene accept the particles were moving) the next performance test returned times as low as 6ms (144fps) and peaking at 12.8ms which was still very comfortably under the 16.7ms target to hit 60fps.

Since rain was key to my visual aim for the scene I needed to find a more optimizaed way of implementing rain as a particle system. The initial rain system I used also spawned a splashing particle whenever a rain drop hit something and since there were thousands of rain particles being spawned a second I wondered if that was causing the performance issues, while adding very little to see the visually as they were hard to see.

After recreating just the rain part of the particle system and running a performance analysis the frame time fell mostly in the 8-10ms range but peaking at 14ms. This was still comfortably under the target for a stable 60fps.

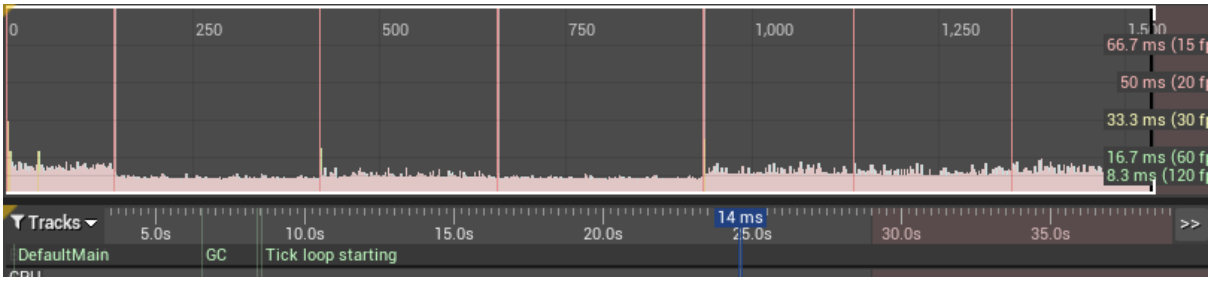
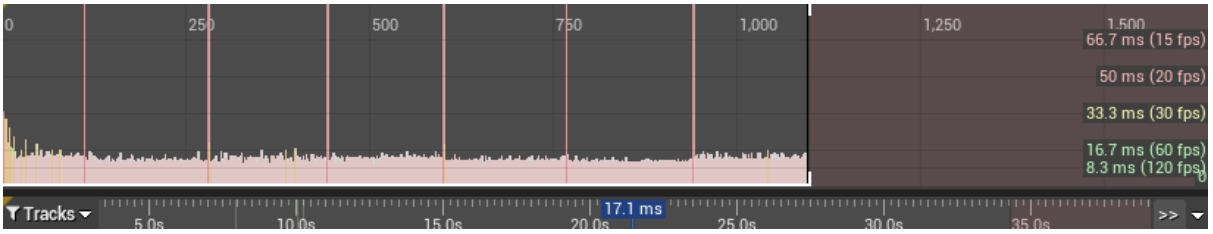


Figure 15 Fourth run, basic rain system

Figure 16 Fifth Run, RTAO enabled

Since I still had some room to up the graphical performance of the scene, I wondered if I could reimplement some raytracing, particularly reflections as I felt they would really enhance the scene. Unfortunately even without a performance it was clear ray traced reflections would be too expensive. However, I did try to see if Raytraced Ambient Occlusion (RTAO) would be fine as I remembered it cost about 2ms previously.



This resulted in a scene that had just a single frame higher than 16.7ms at 17.1ms which I felt was good enough to say I have hit my 60fps target.

Optimising for tablets

In my aim to optimise for mobile I took reference from a note in the unreal docs that mentioned 4th generation iPads and the original iPad Air could handle around 700 draw calls and 500,000 triangles[2] at 30fps as well as a page that listed what post process effects were supported by mobile devices.[3] Since that generation of tablet is over a decade old, modern

tablets such as the iPad 10[4] should be able to handle around that amount if not more than their 4th generation counter parts.[5] So I have aimed to get as close I can to those numbers.

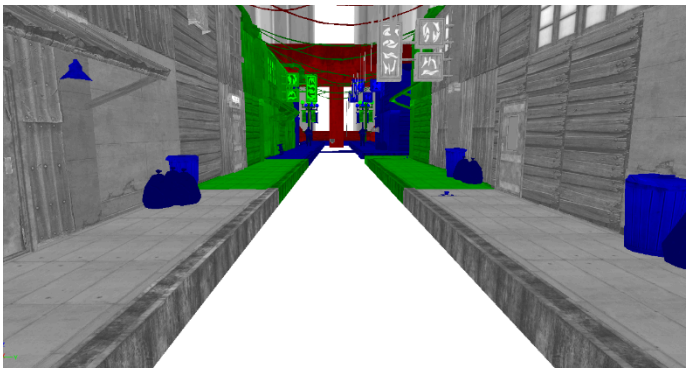


Figure 17 Mobile Scene in LOD view

The first thing I did is lowered the target resolution from 1080p to 720p as I felt that would be sufficient quality for the small screen of a mobile device, while greatly reducing the number of pixels to be rendered. Since my biggest concerns were triangles and draw calls I then naturally looked at implementing LODs to reduce the triangle count and merged the parts of modular meshes together to reduce the number of mesh drawcalls required. This included making the base LOD for all of these meshes have a lower triangle count than they would in the PC version of the game. Another big area for concern with draw calls is the material draw calls. As such I replaced the more complex reflective materials with simpler ones that looked dryer. Furthermore, the floor meshses that had transluscent materials for grates were replaced with meshes that don't to aid in this sense too.

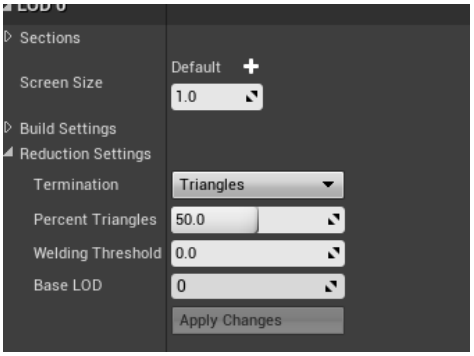


Figure 18 Example LOD0

Figure 19 Less reflective floors

My next big concern was in regards to post processing. Most of them are either incompatible are very costly for a mobile platform. As such all screen effects such as lens flare and vignette and grain jitter were removed alongside the more advanced features such as ambient occlusion, screen space reflections and global illumination. While reflections are still present due to a regular sphere reflection capture, the water source becomes significantly less interesting to look at. I had considered implementing planar reflections to retain some semblance of the high reflectivity of the water but the duplication of the materials drastically increased draw calls and triangles to a level that was untenable.

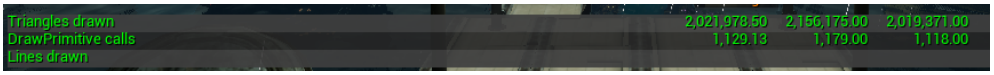


Figure 21 Triangle count and draw calls with planar

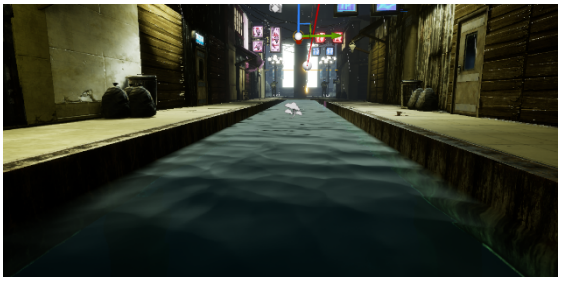
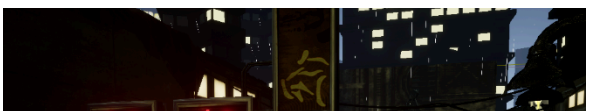


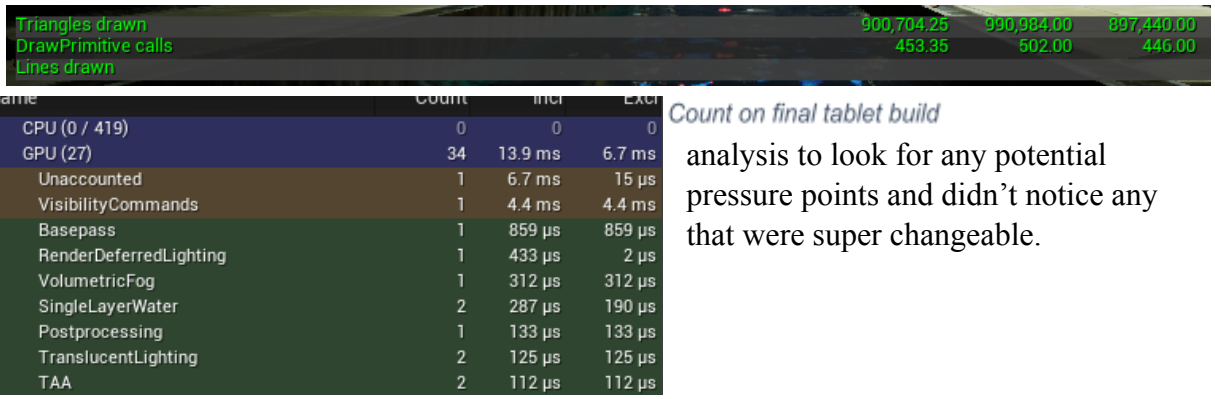
Figure 20 Water on Mobile

The fog's prevalence has been greatly reduced as it was quite expensive to run even on pc, it now primarily is an effect off in the distance. This is combined with many of the light sources, in particular those around the signs being removed causigtheir to be less scatter needing to be calculated in relation to the fog, while also reducing light render times.



All of these changes result in a scene that I think could hit 30fps on modern tablets but probably wouldn't be usable as you go back in generations and certainly not on more typical smartphones. The scene runs at it's worst analysis angle at ~900k triangles and ~500 draw calls but I can't test it on a tablet itself to be sure it would run ok. But for most views, escpecially if it doesn't feature the police officers taking up a large section of the screenit has around 650k triangles which feels more than low enough for modern tablets.

Finally I ran a performance analysis with unreal insights similar to my PC performance



analysis to look for any potential pressure points and didn't notice any that were super changeable.

Name	Count	Incl	Excl
CPU (0 / 419)	0	0	0
GPU (27)	34	13.9 ms	6.7 ms
Unaccounted	1	6.7 ms	15 µs
VisibilityCommands	1	4.4 ms	4.4 ms
Basepass	1	859 µs	859 µs
RenderDeferredLighting	1	433 µs	2 µs
VolumetricFog	1	312 µs	312 µs
SingleLayerWater	2	287 µs	190 µs
Postprocessing	1	133 µs	133 µs
TranslucentLighting	2	125 µs	125 µs
TAA	2	112 µs	112 µs

Figure 23 Final Performance Analysis

Final Thoughts in relation to references

Overall, I think the PC version of the scene does a relatively good job at capturing the vibe of the references. The lighting, reflections and fog really captured the same vibe. However, I think the scene would've been greatly approved by having more people or perhaps even flying cars around to make it feel livelier like the references do. I think the tablet version loses a fair bit of the charm that the references and the PC version have by losing the strong reflection on the water and the fog. The decreased number of rain particles present in the tablet version also take away from some of the chaotic feeling intended by the weather.

Video Report

https://youtu.be/72IyR8yWO1o

References

1. Epic Games (2020), Profiling with Unreal Insights, https://unrealcommunity.wiki/profiling-with-unreal-insights-ilad24y4
2. Epic Games, Performance Guidelines for Mobile Devices https://docs.unrealengine.com/4.27/en-US/SharingAndReleasing/Mobile/Performance/
3. Epic Games, Post Process Effects on Mobile Platforms https://docs.unrealengine.com/4.27/en-US/SharingAndReleasing/Mobile/PostProcessEffects/
4. GSMArena, iPad 10.2 specifications, https://www.gsmarena.com/samsung_galaxy_s10_lite-9917.php
5. GSMArena, iPad 4 specifications, https://www.gsmarena.com/apple_ipad_4_wi_fi_+_cellular-5071.php

Asset References

1. Epic Games (2018) [Assets], Soul: City, https://www.unrealengine.com/marketplace/en-US/product/soul-city
2. PurePolygons-Enviroments (2020) [Assets], Downtown West Modular Pack, https://www.unrealengine.com/marketplace/en-US/product/6bb93c7515e148a1a0a0ec263db67d5b
3. Quixel Megascans (2019) [Assets], Megascans – Cyberpunk Enviroment, https://www.unrealengine.com/marketplace/en-US/product/76e1fa1df6c04146bac76552266f20d4
4. Next Level 3D – Enviroments (2019), Modular Industrial Area, https://www.unrealengine.com/marketplace/en-US/product/modular-industrial-area
5. UNEASY(2018) [Blueprint], Good Sky, https://www.unrealengine.com/marketplace/en-US/product/good-sky
6. Epic Games(2020) [Assets], Showdown VR Demo, https://www.unrealengine.com/marketplace/en-US/product/showdown-demo
7. Crazy_8 (2020) [Assets], FuturisticSkyscrapers Pack, https://sketchfab.com/3d-models/futuristic-skyscrapers-pack-07d66785c56d4a56b930e60cfadc5320