

# **Neotropic: Creation of a Modular Environment**

Xavier Marín Solà

Universitat Politècnica de Catalunya - CITM  
Bachelor's degree in Video Game Design and Development

David Masana Lafuente

September 16th, 2022



UNIVERSITAT POLITÈCNICA DE CATALUNYA  
BARCELONATECH

Centre de la Imatge i la Tecnologia Multimèdia

## Table of Contents

<b>Table of Contents</b>	<b>2</b>
<b>1. Introduction</b>	<b>6</b>
1.1 Abstract	6
1.2 Keywords	6
1.3 Acknowledgements	6
1.4 Links	7
1.5 Preface	7
1.5.1 Motivation	7
1.5.2 The Problem	7
1.5.3 The Goals	8
1.5.4 Scope of the Project	9
1.6 List of Tables	9
1.7 List of Figures	9
1.8 Glossary	12
<b>2. State of the Art</b>	<b>14</b>
2.1 Theoretical Framework	14
2.1.1 Stylized Cyberpunk Environment	14
2.1.2 Modular Asset Pack	16
2.2 Current Workflow	17
2.2.1 Planning Stage	18
2.2.2 Blocking Stage	18
2.2.3 Modeling Stage	19
2.2.4 Texturing Stage	19
2.2.5 Building Stage	20
2.3 Market Research	21
<b>3. Project Management</b>	<b>22</b>
3.1 Procedure and Tools for Project Monitoring	22
3.1.1 GANTT Chart	22
3.1.2 Kanban Board	23

3.1.3 GitHub Repository and Version Control Tools	24
3.2 Evaluation tools	24
3.3. SWOT Analysis	24
3.4. Risks and Contingency Plans	25
3.5. Initial Cost Analysis	26
<b>4. Methodology</b>	<b>28</b>
4.1 Phases of Development	28
4.1.1 Pre-Production	28
4.1.2 Production	29
4.1.3 Post-Production	29
<b>5. Development of the Project</b>	<b>30</b>
5.1 Modular Environment Workflow in Game Industry	30
5.1.2 Modeling for Coordinate-Based Snapping	30
5.1.2.1 Preparation	31
5.1.2.2 Workflow	32
5.1.3 Texturing with Tiling Textures and Trim Sheets	33
5.1.3.1 Preparation	34
5.1.3.2 Workflow	34
5.2 Solutions for Modular Environment Art Fatigue	36
5.2.1 Lighting Solution	37
5.2.2 Hero Asset Solution	38
5.2.3 Decals Solution	39
5.2.4 Extra Props Solution	40
5.3 Breakdown of the Project	41
5.3.1 Game References	41
5.3.2 Asset List	43
5.3.3 Blockout	45
5.3.4 Main Features	45
5.3.4.1 Modular and Tileable Buildings, Walls and Railings	45
5.3.4.2 Custom Decoration	49

5.3.4.3 Interior Cubemaps	50
5.3.4.4 Colorizable Decals	52
5.3.4.5 Dynamic and Emissive Signs	53
5.3.4.5 Toon Shader	54
5.3.4 Demo Scene	56
5.3.5 Submit Content in UE Marketplace	58
5.3.5.1 Folder Organization	58
5.3.5.2 File Nomenclature	59
<b>6. Conclusions</b>	<b>60</b>
<b>7. References</b>	<b>62</b>

## 1. Introduction

### 1.1 Abstract

As time goes by, things evolve, usually for the better, and video games are no different. If we take a look at the history of video games, we can see that environments are becoming more detailed, more innovative and better in terms of visual quality and optimisation than before. That is why environment artists must understand how to best combine visual fidelity with performance and high-volume productivity.

Modularity or modular architecture is a very useful technique in environment art, giving the artist the solution to avoid art fatigue and production delays during the development of a video game. In addition, teams benefit from the flexibility and effectiveness of working with modular assets.

Although there is not much documentation on this topic, I will talk about modularity so that anyone with little to no knowledge of this topic can be introduced to the concept in depth. I have also created *Neotropic*, a 3D modular game environment to explain the workflow used and how the modular architecture can be applied.

Working on an environment with modular techniques proved beneficial in terms of speed and flexibility when creating the design of the level. Otherwise, when poorly designed, it fails to achieve the intended level of optimisation.

### 1.2 Keywords

3D Art, Environment Art, Asset Pack, Modular Environment, Level Design.

### 1.3 Acknowledgements

First, I would like to acknowledge everyone that has helped me to get here, not only academically but also on a personal level. Thanks for the support and inspiration to all those who are part of my life such as my family and friends. You are a part of this project as well.

Also, I would like to thank my thesis tutor David Masana Lafuente for the guidance during these months. It has been a pleasure for me to have you as a mentor. Always attentive to me, being a source of advice and knowledge when I needed it most.

## 1.4 Links

- Video: [Link](#)
- Github Repository: [Link](#)
- Project Download: [Link](#)

## 1.5 Preface

### 1.5.1 Motivation

Creating an environment able to immerse the player in the game experience is something that has always amazed me. I think that having a good atmosphere when playing a game or watching a movie is very important and gives a lot of sense to the development of the story.

I came up with this idea when Joan Barduena and I were developing a *Third-Person Shooter* game in *Unreal Engine 4* where we had to design a level from scratch. I was in charge of creating the environment and it took me a lot of time to place all the assets, so I wondered if there was another way to do it which was more comfortable. It was at that moment when I first read about the concept of modularity.

### 1.5.2 The Problem

Creating a whole environment is not an easy task and requires a lot of time. There are a lot of small projects that cannot afford hiring more than one artist, so they have to be as efficient as possible to reach the objectives in each milestone.

Modular architecture is a technique which has always been present in environment creation. Although it is quite known, it is poorly documented and a lot of people avoid it due to the difficulty of planning or inexperience in this field. However, nowadays modern technology has given us a lot of freedom and artists have been able to refine their workflows to create new forms of modular asset creation that overcome the drawbacks of traditional techniques.

### 1.5.3 The Goals

I have identified two general goals for this thesis:

- **Provide a modular environment** for game makers ready to be downloaded in the UE Marketplace. They may use it to portray their level design idea into a game engine, to use it in game jams or academic projects, to test the asset pack and make sure they understand the process of creating a modular environment exposed in this thesis, etc.
- **Document the thesis** where the whole workflow carried out to get this modular environment done is explained, as well as more things such as modular environment theory, how to manage the project, methodology used, some useful analysis (risks, SWOT, cost, etc.), final conclusions, references, etc.

There are also specific goals that will help the general objective to be fulfilled more easily:

- **Optimal in terms of performance:** Video games have always required certain optimisation. It is true that as the years go by, game engines are able to support more tons of memory but this does not mean that optimisation is no longer important. So, one of the most important specific objectives is to make a stable pack so that the game's performance is not affected.
- **Understandable memory:** Have a totally understandable and well explained thesis in order to facilitate people, even with little experience in the field, to comprehend the concept and the whole process to create this.
- **Good-looking and flexible assets:** Try to achieve the high quality standards of a professional 3D artist in terms of aesthetics and functionality, so that everyone can be able to use it easily.
- **Free download:** There are not many free asset packs in the market, so I want to provide one to the people who cannot afford it or have a purpose which is not worth paying such an amount of money.

### 1.5.4 Scope of the Project

The project is aimed to be posted in the UE Marketplace or Unity store. Everyone will be able to use the asset pack at their convenience. So, for this project I do not have a specific and clear target, it can be used by anyone who wants to have a final and polished environment for his/her level design idea, for game jams purposes, academic projects for students that may need a free tool to build their game environment, etc.

Additionally, I will document the creation process of a modular 3D environment for a game engine so that a reader who is interested in this topic can be introduced to the world of 3D art in video games.

### 1.6 List of Tables

**Table 1:** SWOT Analysis

**Table 2:** Risks and Contingency Plans

**Table 3:** Initial Costs Analysis - Total Costs

**Table 3.1:** Initial Costs Analysis - Hardware

**Table 3.2:** Initial Costs Analysis - Software

**Table 3.3:** Initial Costs Analysis - Salary and Living Costs

**Table 4:** General Assets Information

### 1.7 List of Figures

**Figure 1:** Cyberpunk Environment by Helio Frazao

**Figure 2:** Cyberpunk Environment by Denys Rutkovskiy

**Figure 3:** Stylized Environment from Ratchet & Clank

**Figure 4:** Stylized Environment from Fortnite

**Figure 5:** 3D Vikings Asset Pack by Polygon

**Figure 6:** Modular Building Example by Alt3d

**Figure 7:** Workflow Stages

**Figure 8:** Building Moodboard Example

**Figure 9:** Environment Blockout Example

**Figure 10:** 3D Modular Building Example

**Figure 11:** Trim and Tile Textures Examples

**Figure 12:** Final Result of the Environment in the Engine



**Figure 13:** Modular Cyberpunk City Asset Pack by HpdizzoArt

**Figure 14:** Modular Cyberpunk Environment by Tris Games

**Figure 15:** Cyberpunk Environment Megapack by Leartes Studios

**Figure 16:** Gantt Chart

**Figure 17:** Kanban Board

**Figure 18:** Phases of Development

**Figure 19:** Set of Modular Elements prepared for Coordinate-Based Snapping

**Figure 20:** Example of Blockout for a Modular Environment

**Figure 21:** Example of Trim Sheet Texture

**Figure 22:** Example of Tiling Texture

**Figure 23:** Room of The Elder Scrolls IV: Oblivion by Bethesda Softworks

**Figure 24:** Scenario full of Modular Elements in The Last of Us: Part II by Naughty Dog

**Figure 25:** Modular Environment with a Hero Asset by Kobus Viljoen

**Figure 26:** Example of Decals (Graffiti)

**Figure 27:** Extra Props Moodboard of my Cyberpunk Environment

**Figure 28:** Reference Game: Borderlands 3 Environment

**Figure 29:** Reference Game: Borderlands 3 Environment

**Figure 30:** Moodboard of Neotropic Environment

**Figure 31:** Blockout Renders of the Environment

**Figure 32:** Building resized with a Non-Tileable Texture

**Figure 33:** Building resized with a Tileable Texture

**Figure 34:** Blueprint of the Tileable Master Material

**Figure 35:** Details of a Tileable Material Instance

**Figure 36:** Ceiling Blueprints

**Figure 37:** Vending Machines Blueprints

**Figure 38:** Cubemap Image Template

**Figure 39:** Blueprint of the Cubemap Master Material

**Figure 40:** Details of a Cubemap Material Instance

**Figure 41:** Blend Configuration of the Decal Master Material

**Figure 42:** Blueprint of the Emissive Master Material

**Figure 43:** Array storing the Toon Shader Materials

**Figure 44:** Comparison: Without Toon Shader vs With Toon Shader

**Figure 45:** Render of the Demo Scene

**Figure 46:** Render of the Demo Scene

**Figure 47:** Render of the Demo Scene

**Figure 48:** Render of the Demo Scene

**Figure 49:** 'Content' folder of the Project

**Figure 50:** 'CyberPunk\_AssetPack' folder of the Project

**Figure 51:** Nomenclature Example

**Figure 52:** Comparison: *Borderlands* vs *Neotropic*

## 1.8 Glossary

### 3D

Three-dimensional. Something that has width, height and depth (length).

### Alpha

Measures the opacity of a pixel in an image.

### Asset

Everything that can help in the development and visual set-up of a game. Models, textures, UI, audio are some examples of game assets.

### Baking

The process where the software gets information from a 3D model and creates a new texture file.

### Blockout

Is a process where you use primitive geometric shapes (cubes, spheres, cylinders, planes, etc.) to block-in your level designs, game environments and game art assets.

### Game Engine

Software framework primarily designed for the development of video games

### Geometry

Is the combination of 2D shapes (polygons) that form a 3D model.

### Material

Is the combination of different textures that can be applied to an object/mesh to define the visual look of an object.

### PBR

PBR stands for Physical Based Rendering. Is a computer graphics approach that seeks to render images in a way that models the flow of light in the real world.

### Texture Map

Is an image applied to the surface of a shape or polygon that gives a specific type of information when we create the material with the combination of some of them.

The types of PBR texture maps are:

- **Albedo**  
Is the basis of your entire material. They are a flat light image of the pattern you want to work with.
- **Normal**  
Give your textures depth. This uses complex calculations to fake the way light interacts with the surface of the material.
- **Roughness**  
Define how light is scattered across the surface. It is a grayscale map.
- **Metalness**  
Used to define if your material is bare metal. It is a grayscale map too.
- **Ambient Occlusion**  
Defines how it reacts to light by combining with albedo. It is also a grayscale map.
- **Emissive**  
Used to make your material seemingly emit their own light so they are still visible in dark areas.
- **Opacity**  
It allows you to make parts of your material transparent. Opacity maps are grayscale where white is fully opaque and black is transparent.

### **UV Mapping**

Is the 3D modeling process of projecting a 2D image to a 3D model's surface for texture mapping.

## 2. State of the Art

### 2.1 Theoretical Framework

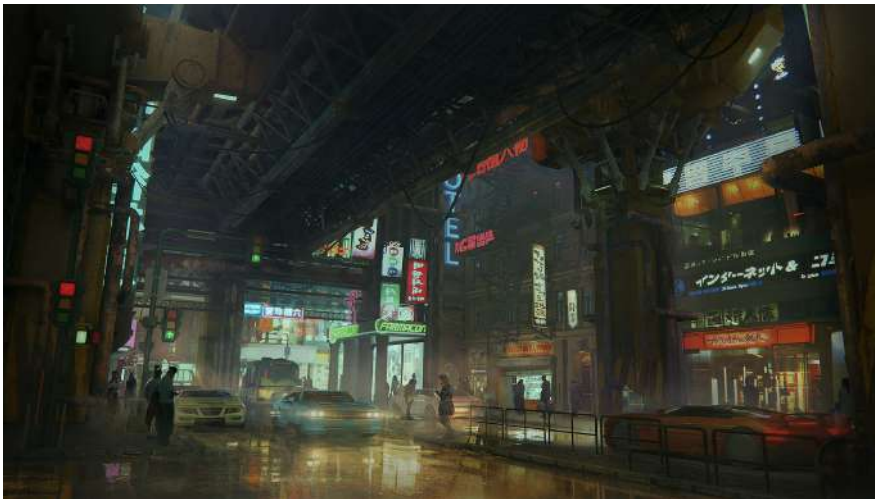
#### 2.1.1 Stylized Cyberpunk Environment

I think that the best way to explain this concept is by defining both separately:

- **Cyberpunk:** Is a subgenre of science fiction in a dystopian future setting that tends to focus on a “combination of lowlife and high tech”, featuring futuristic technological and scientific achievements, such as artificial intelligence and cybernetics, juxtaposed with societal collapse or decay.

**Figure 1**

Cyberpunk Environment. Concept art by *Helio Frazao*



**Figure 2**

Cyberpunk Environment. Image from the asset pack by *Denys Rutkovskiy*





- **Stylization:** The opposite of realism. Is a form of art in which you are free to play with the shapes and colors, exaggerate or remove details to enhance the look and feel in any direction. Doing so with realism would break the illusion of reality as it would not be viewed as what we perceive to be as 'realistic', it would not belong in our world.

**Figure 3**

Stylized environment from *Ratchet & Clank (PS4)* by *Insomniac Games*



**Figure 4**

Stylized environment from *Fortnite* by *Epic Games*

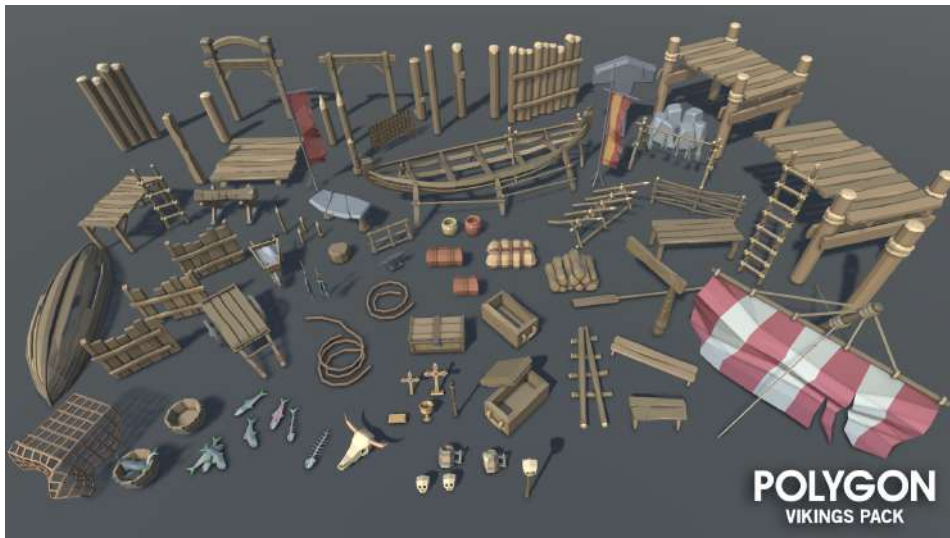


### 2.1.2 Modular Asset Pack

Breaking down the concept into two parts again, will make it easier to understand:

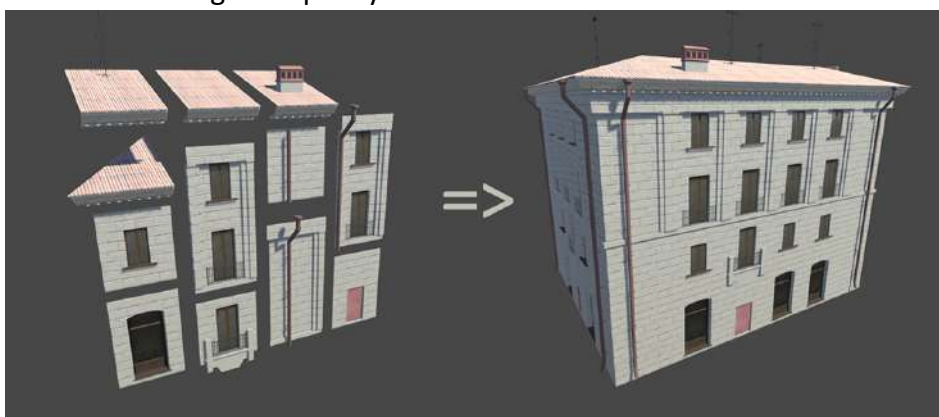
- **Asset Pack:** Is a set of assets, in this case 3D models, which belong to the same theme and are used to create scenarios. A clear example of asset pack is this one below:

**Figure 5**  
3D Vikings Asset Pack by Polygon



- **Modularity:** Is the degree to which a system's components may be separated and recombined, often with the benefit of flexibility and variety in use. But how is it applied to 3D environments in video games? With modularity you can organize groups of assets into reusable, inter-linkable modules to form larger structures. The core idea is reusing as much work as possible in order to save memory and improve performance.

**Figure 6**  
Modular Building Example by Alt3dt



## 2.2 Current Workflow

There is not a single and unique way to create a modular environment. Usually, all the steps of the workflow follow a logic and optimal order but each process can be carried out with many different softwares which are chosen by artists according to their preferences.

In order to explain the workflow I will be taking as a reference a project made by *Nathan Alderson*, a 3D Environment Artist who wrote an article in *80.lv* called '*Making a Stylized Cyberpunk Scene in Blender and UE4*'.

I do not know what difficulties and obstacles I may encounter during my journey creating a modular environment for the first time since every project is different but I want to emphasize something *Garth Travis* said which I completely agree with and will apply to the development of the project:

*"I think it is important to analyze other people's technical work and to brainstorm how it may have been done and then find the knowledge on how to replicate and practice these techniques."* - *Garth Travis, Terrain Environment Artist at Starbreeze Studios.*

The process is divided in five different stages:

1. **Planning Stage:** Set-up the project.
2. **Blocking Stage:** Prepare a blockout scene.
3. **Modeling Stage:** Create the 3D assets needed according to your plan.
4. **Texture Stage:** Create the textures for the assets you have done.
5. **Building Stage:** Build the whole environment in the engine.

**Figure 7**  
Workflow Stages





### 2.2.1 Planning Stage

This stage is based on gathering information and images you want to take as a reference. They can be whatever you believe may be useful for you; screenshots from other projects, google maps, handmade sketches, etc. Then, some moodboards are created with all these resources. It is important to always bear in mind what kind of environment is going to be created in order to maintain the consistency in the artstyle.

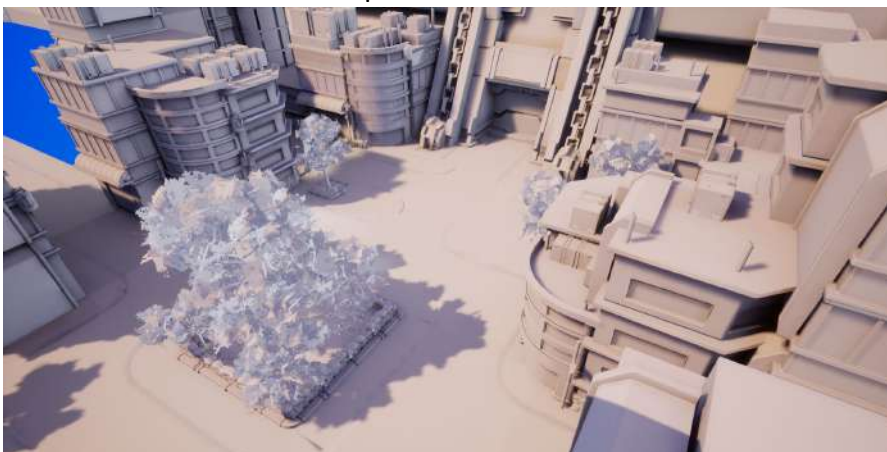
**Figure 8**  
Building Moodboard Example



### 2.2.2 Blocking Stage

At this stage, a blockout of the whole scenario is created in the game engine with primitive shapes. It is easy to underestimate how much needs to be made, so this stage is essential because it gives you a scope of it. It also helps to figure out what are the assets needed and which of them have to be modular so that they can be reused.

**Figure 9**  
Environment Blockout Example



### 2.2.3 Modeling Stage

This stage is mainly dedicated to modeling all the assets that are gonna be in the environment. Several software exist to do this, but some of the most used in the industry are *3Ds Max*, *Maya*, *Blender* and *ZBrush*.

**Figure 10**  
3D Modular Building Example



*\*Modeling and texturing stages can be interspersed depending if the artist prefers to model everything and then texture everything or model and texture one asset at a time and then another. It is up to the person.*

### 2.2.4 Texturing Stage

Basically aimed to texture all the 3D assets. This stage obviously involves unwrapping the UVs and baking the maps of the models from highpoly to lowpoly. Sometimes, instead of doing bakes, we can use trim sheets in order to optimize and speed up the process. Just as in modeling, there are also many software for texturing but the most common ones are *Substance Painter* and *Substance Designer*.

**Figure 11**  
Trim and Tile Textures Examples



### 2.2.5 Building Stage

Last but not least, on this stage everything is wrapped up in the engine. The final environment is built up by placing the final assets following the blackout we did previously and setting up the light. Finally, the whole scene will be checked and polished.

#### Figure 12

Final Result of the Environment in the Engine

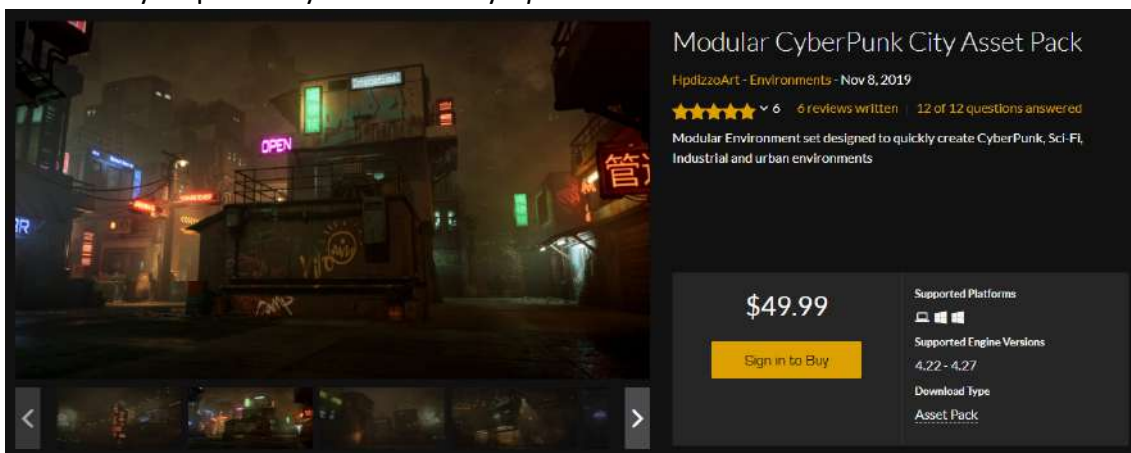


## 2.3 Market Research

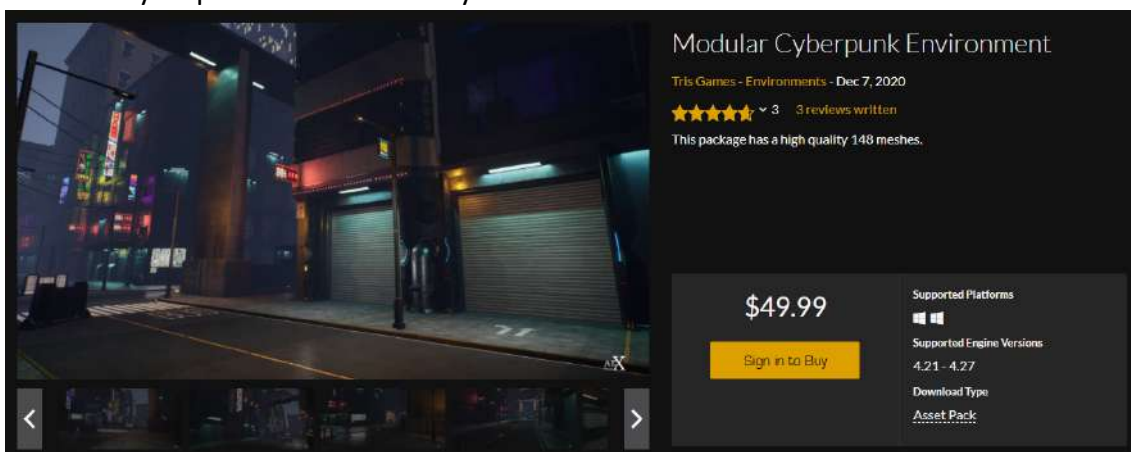
The market is full of free 3D assets, there are many websites where you can find tons of models of whatever you are looking for. However, it comes different when talking about a specific themed modular scenario.

When searching for 'cyberpunk environment' in *Unreal Engine Marketplace* there is a huge amount of available packs. Here are some examples of those which can approach the most to my project:

**Figure 13**  
Modular Cyberpunk City Asset Pack by *HpdizzoArt*

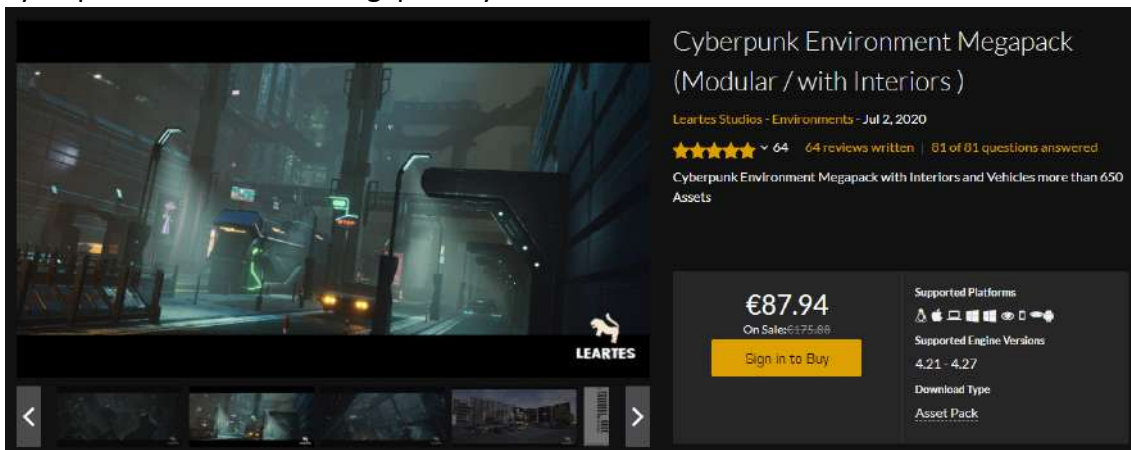


**Figure 14**  
Modular Cyberpunk Environment by *Tris Games*





**Figure 15**  
Cyberpunk Environment Megapack by *Leartes Studios*



As you have seen, there are so many options to choose from but there is a problem, none of them is free. Definitely, this is the gap I want to take profit of.

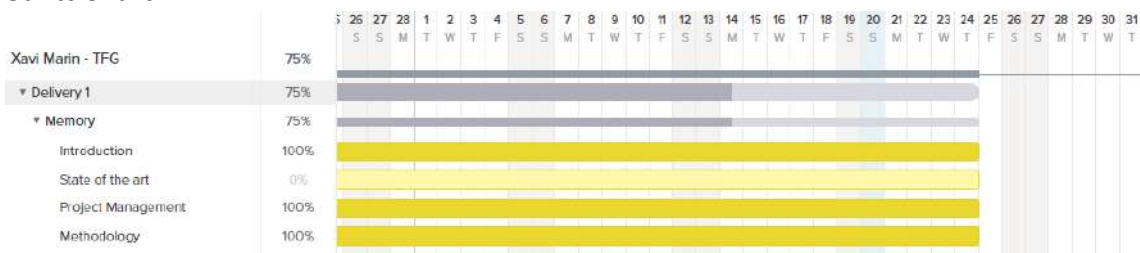
### 3. Project Management

#### 3.1 Procedure and Tools for Project Monitoring

##### 3.1.1 GANTT Chart

I have created a Gantt chart using *teammantt* which contains all the tasks for every development phase and it is useful to keep the thesis tracked in terms of pacing.

**Figure 16**  
Gantt Chart



A Gantt chart, commonly used in project management, is one of the most popular and useful ways of showing tasks displayed against time. On the left of the chart there is a list of the activities and along the top is a suitable time scale. Each activity is represented by a bar; the position and length of the bar reflects the start date, duration and end date of the activity.

### 3.1.2 Kanban Board

I have created a Kanban board using *Trello* which indicates the status of all tasks of the current phase I am working on. I will be able to easily manage tasks in little self-contained blocks and compare it with the Gantt chart to be aware of my current progress.

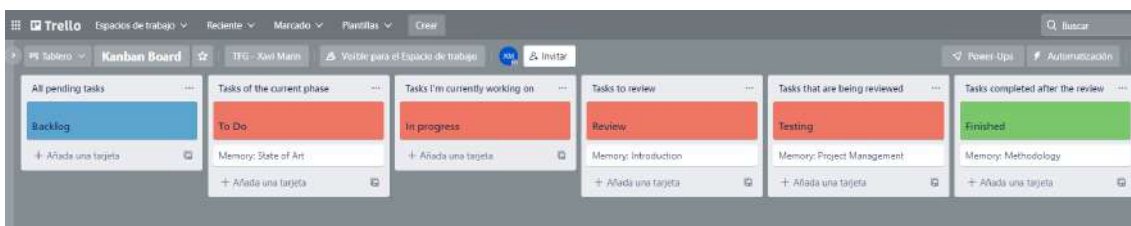
I have separated the board in six different sections to move every task depending on its current status. The sections are the following ones (sorted from first to last phase):

- **Backlog:** Tasks that are pending.
- **To Do:** Tasks I have to do for my current phase of development.
- **In Progress:** Tasks that I am currently working on.
- **Review:** Tasks that I need to review if they fulfill all desired requirements.
- **Testing:** Tasks that are currently being reviewed and almost finished.
- **Finished:** Tasks that have been reviewed and I consider them finished.

The board also features priority tags which will be useful to identify which task must be done first and which one can be done later. These tags are the following ones:

- **Green:** Low prio
- **Yellow:** Medium prio
- **Orange:** High prio
- **Red:** ASAP prio

**Figure 17**  
Kanban Board



A kanban board is an agile project management tool designed to help visualize work, limit work-in-progress, and maximize efficiency. It can help teams establish order in their daily work.

### 3.1.3 GitHub Repository and Version Control Tools

GitHub will be the software which will host my project. It is easy to keep track of all changes and control everything since all versions are stored there.

In my case, I will only use the master branch to push all the changes done in the project. The repository will remain private until the development is finished.

### 3.2 Evaluation tools

The main source of evaluation will be my mentor, David Masana. I will be able to evaluate my approach with his feedback which will be very useful to further improve any aspect of the project, as well as solve the doubts I may have during the development of the environment.

### 3.3. SWOT Analysis

**Table 1**  
 SWOT Analysis

	Positive	Negative
Internal	<p style="text-align: center;"><b>Strengths</b></p> <ul style="list-style-type: none"> <li>- Experienced in the workflow of 3D asset creation.</li> <li>- The project does not require a great monetary cost because of the software's student versions.</li> <li>- Can be used by everyone who has basic knowledge about any game engine.</li> </ul>	<p style="text-align: center;"><b>Weaknesses</b></p> <ul style="list-style-type: none"> <li>- Inexperienced in modular architecture.</li> <li>- Procedural environment generation is even more flexible and is expected to be the way used in the future.</li> </ul>
External	<p style="text-align: center;"><b>Opportunities</b></p> <ul style="list-style-type: none"> <li>- The project is aimed to be sold for free.</li> <li>- My mentor is an environment artist in a AAA game.</li> <li>- Is the common workflow used in game development.</li> </ul>	<p style="text-align: center;"><b>Threats</b></p> <ul style="list-style-type: none"> <li>- My job might reduce my available amount of time.</li> <li>- There are already many modular asset packs in the market done by professional teams.</li> </ul>

### 3.4. Risks and Contingency Plans

Developing a project like this by a person who has never done something similar, with limited time and little documentation on the Internet can lead to encountering some issues that can slow down the development of the project or even stop it.

I will use this table of *Risks and Contingency Plans* in order to be one step ahead to prevent any possible risk.

**Table 2**  
Risk and Contingency Plans

Risk	Solution
There is not enough time to finish all the planned assets	Reduce the number of assets that will be in the asset pack
There is no chance to upload the asset pack to UE Marketplace or Unity Store	Create a website announcing the asset pack and let people download it from that website
The customization of the whole environment is too complex for users	Add some guidelines (video/document...) to improve product usability
The asset pack is not as modular as I expected	No worries as far as I am able to optimize it and facilitate flexible level design.
There is not enough time to create something (material, model, texture, blueprint...)	Look for it on the Internet and use or modify that free asset for my project



### 3.5. Initial Cost Analysis

The project is not aimed to make any profit. Bear in mind that there are similar projects done by professional teams which are sold in *UE* and *Unity* stores. So, I will not be getting any economic benefit but a study will be done in order to determine what would be the price in the market.

All the work will be developed during the course of four months.

I have divided the costs in different groups:

- Hardware
- Software
- Salary and Living Costs

**Table 3**

Initial Cost Analysis - Total Costs

Total Costs	Price (€)
Hardware	1450€
Software	0€
Salary and Living Costs	3600€
<b>TOTAL - 4 MONTHS</b>	<b>5050€</b>

**Table 3.1**

Initial Cost Analysis - Hardware

Hardware	Price (€)
Desktop Computer (5 years old)	990€
Wacom Pen Tablet (4 years old)	250€
Peripherals (2 years old)	210€
<b>TOTAL</b>	<b>1450€</b>

**Table 3.2**  
Initial Cost Analysis - Software

Software	Price (€)
Autodesk Maya 2022	0€
ZBrush 2021	0€
Substance Painter	0€
Unreal Engine 4	0€
Photoshop 2022	0€
Trello	0€
TeamGantt	0€
<b>TOTAL</b>	<b>0€</b>

*\*Many of the software will be using the student's free license*

**Table 3.3**  
Initial Cost Analysis - Salary and Living Costs

Salary and Living Costs	€/month
Salary	600€
Household Expenses	50€
Feeding	250€
<b>TOTAL</b>	<b>900€</b>
<b>TOTAL - 4 MONTHS</b>	<b>3600€</b>

*\*Taking as a reference the salary I have in my Socialpoint internship (8€/h)*

## 4. Methodology

### 4.1 Phases of Development

The development of this project will be divided in 3 main phases, with their corresponding sub-phases:

#### 1. Pre-Production

- **Preparation Phase:** Set-up of the project.
- **Investigation Phase:** Exploration of the theory surrounding the topic.

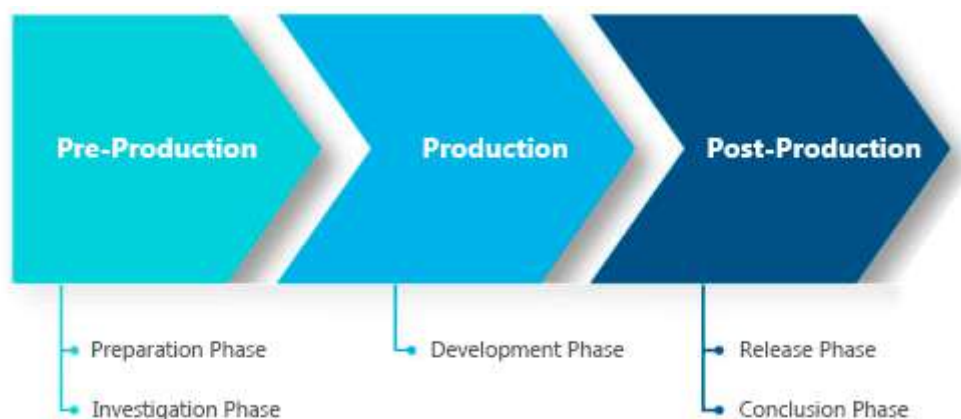
#### 2. Production

- **Development Phase:** Creation of the modular environment and documentation.

#### 3. Post-Production

- **Release Phase:** Preparation and upload of the pack.
- **Conclusion Phase:** Finalization of the memory.

**Figure 18**  
Phases of Development



#### 4.1.1 Pre-Production

- **Preparation Phase:** During this phase, I want to analyze and decompose the project to have a general overview of how I will manage this. First, I do a brief research about the topic proposed in order to identify which objectives are available to reach according to time and what are the needs to achieve them. In this phase I also create the document and lay out the main structure of it.

- **Investigation Phase:** This phase is dedicated to investigating the concept of modularity in game environments deeply and how it is being applied nowadays. Meanwhile, I will also be developing all the previous sections of the memory before entering into the development part of the project.

#### 4.1.2 Production

- **Development Phase:** The most important part of the project, the demonstration of the thesis. Basically, in this phase I will be creating all the 3D assets needed for the environment creation taking into account that they must be modular. In addition, I will also be exposing in the memory how to carry out this process.

#### 4.1.3 Post-Production

- **Release Phase:** This phase starts once the Production is totally finished. Here I will do the release of the asset pack in the UE Marketplace, including a full description of the pack as well as the creation of a demo environment in order to showcase the whole project.
- **Conclusion Phase:** The last phase consists in finishing the memory in its entirety and making sure it is ready to be delivered.

## 5. Development of the Project

### 5.1 Modular Environment Workflow in Game Industry

Usually, the most common way to create a modular environment is modeling for coordinate-based snapping and texturing with tiling textures and trim sheets.

#### 5.1.2 Modeling for Coordinate-Based Snapping

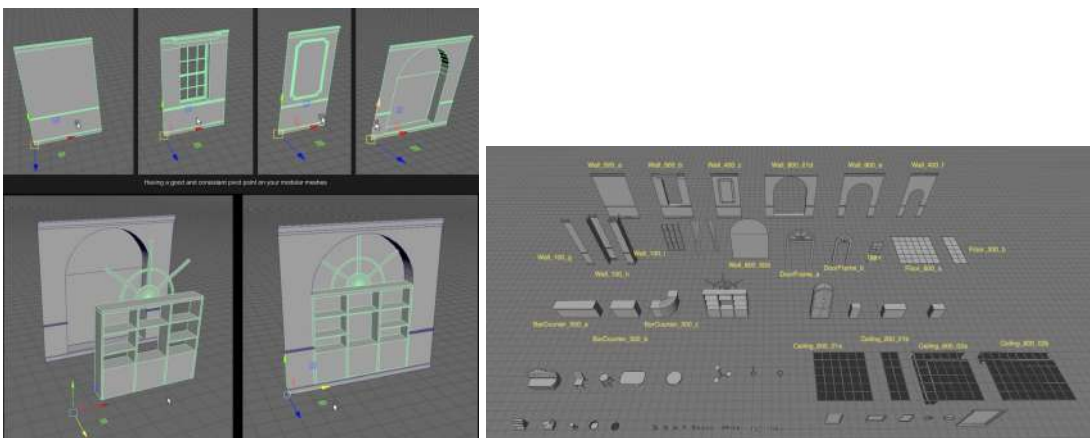
Since the invention of the Cartesian coordinate system in the 17th century, we have been able to represent a point in theoretical space with the use of three coordinates. Naturally, as the internal language of computers is based on mathematics, it follows that the earliest games and game engines used the same coordinate system to determine where an object was located in virtual space. This way, we can fit individual assets within standard grid units, which is the basis of creating modularity in games.

Coordinate-based modularity primarily concerns the process of modeling assets and yields great benefit in the later assembly of assets into environments. Successful use of the grid snapping technique usually involves more preparation than changes to modeling or texturing workflows. However, artists must maintain a balance between the standardized rules of the modular system and their aesthetic values as an artist.

According to *Joel Burges* (Level Designer) and *Nate Purkeypille* (Environment Artist) at *Bethesda Softworks*, designing a modular system within a coordinate grid is a cumulative process and each decision will have ramifications that will affect, sometimes negatively, decisions made later in the project.

**Figure 19**

Set of Modular Elements prepared for Coordinate-Based Snapping



### 5.1.2.1 Preparation

To begin planning for a coordinate-based modular environment, designers must first determine the basic units by which their coordinate plan will be divided. Typically, a number is chosen because it is easily divisible into smaller units, facilitating the creation of assets of various sizes larger and smaller.

In many cases, concept art is used as a basis to understand which pieces need to be created and how their scale can be divided into units. Designers, then, consider how to achieve an environment that closely resembles the concept with grid-bound pieces, formulating a basic set of modules that must be created. By doing a blockout, an artist can quickly create a set of pieces that more or less resembles each intended piece, allowing experimentation within a game engine to determine if changes are needed.

**Figure 20**  
Example of Blockout for a Modular Environment



Nowadays, audiences expect realism and immersion into the game's environment. So, a well-planned modular system balances variation with standardization, allowing level designers to create distinct spaces with interesting features, with minimal deviation from the intended relationship between modules.

### 5.1.2.2 Workflow

Creating assets for a coordinate-based snapping environment kit involves certain considerations for 3D artists but doesn't deviate much from the traditional process of creating 3D game assets. Texturing takes place after models have been created, and so they fall outside of the scope of this technique, which affects only the modeling stage.

Artists must maintain an awareness of how their model fits within the grid space for which it is designed and how it must relate to other kit assets. They must ensure their assets fit entirely within their assigned grid space, without occupying the place of any neighbor pieces because, if that happens, there will appear a visual anomaly called *z-fighting* where the hardware will try to render both objects at the same time in the same world position.

Nowadays, 3D modeling software operates on a Cartesian coordinate system, easily facilitating the creation of grid-bound assets. An artist can manually enter the desired position of certain points or use the features of most 3D software to place edges and vertices on a precise point of the virtual space.

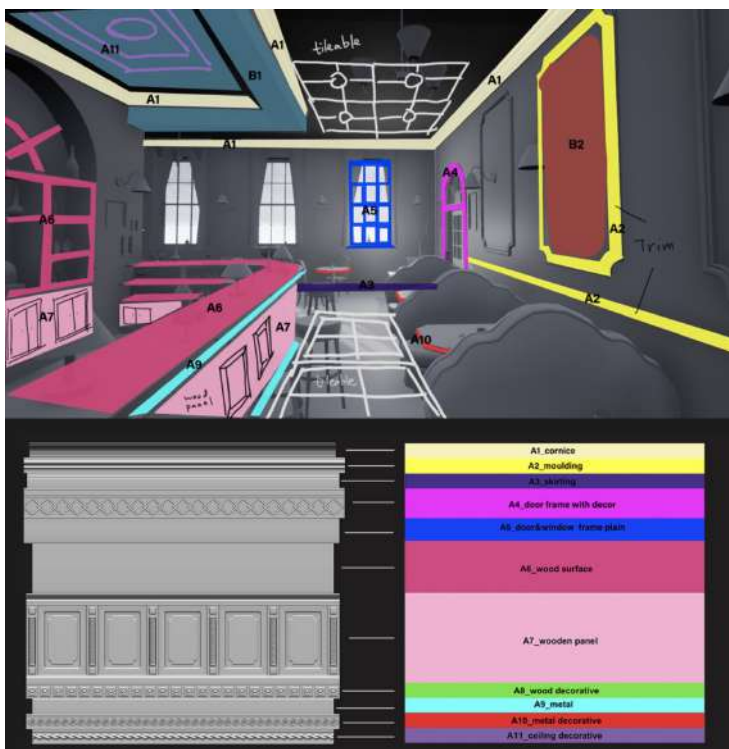
A common mistake when creating modular kits is misplacing the origin point of a model. The origin point is a single point in virtual space by which software knows the model's location. It is manipulated within a game engine and snapped to points on a grid, and a kit will not assemble correctly if the origin point is misplaced. Origin points can move frequently during the modeling process, as individual pieces are combined or used as a point of reference, and the artist must take care that it is reset to the correct location before the model is finalized. It is important to bear in mind that combining many models into a single asset can make the origin point being moved to the center of the objects. This is not ideal for modular assets because the placement of origin points can negatively affect the usefulness of asset kits.

### 5.1.3 Texturing with Tiling Textures and Trim Sheets

When designing textures for modular environment models, artists commonly break down each object into two simple categories: parts which should use a tiling texture, and parts which should not. Some objects, such as a brick wall, involve flat planes of repetitive texture, and are best represented by creating a tiling texture image. Other objects are more detailed and might include various areas that are distinct from one another and require more specific texturing, these ones need a trim sheet texture. However, modern game engines allow artists to combine both tiling and other materials in one asset by assigning different materials to different faces of the model.

Trim sheets are an evolution of the texture atlas, a technique in which artists combine many textures onto a single image that gives flexibility and efficiency to the artist. In the traditional workflow, textures are created for the model which they will cover after it is completed. Otherwise, when working with a texture atlas, artists usually map the UVs of a model to an already prepared texture, which allows them to reuse textures across multiple assets that use the same material and give them a wide range of performance benefits.

**Figure 21**  
Example of Trim Sheet Texture





Other areas of the model are mapped to the tiling texture, mimicking real-world environments which often consist of details set against flat areas of texture, for example the floor of a bar. When done well, this technique also allows an artist to texture unplanned assets with relative ease, allowing a single texture sheet to cover both current and future assets that will consist of the same materials.

### 5.1.3.1 Preparation

Planning for the use of trim sheets is dependent upon a complete understanding of the shapes and details a set of models will include. Not only must all needed shapes be anticipated, but texture space must be efficiently divided, room left for variations if desired, and the scale of each area must allow for enough resolution on surfaces without distortion.

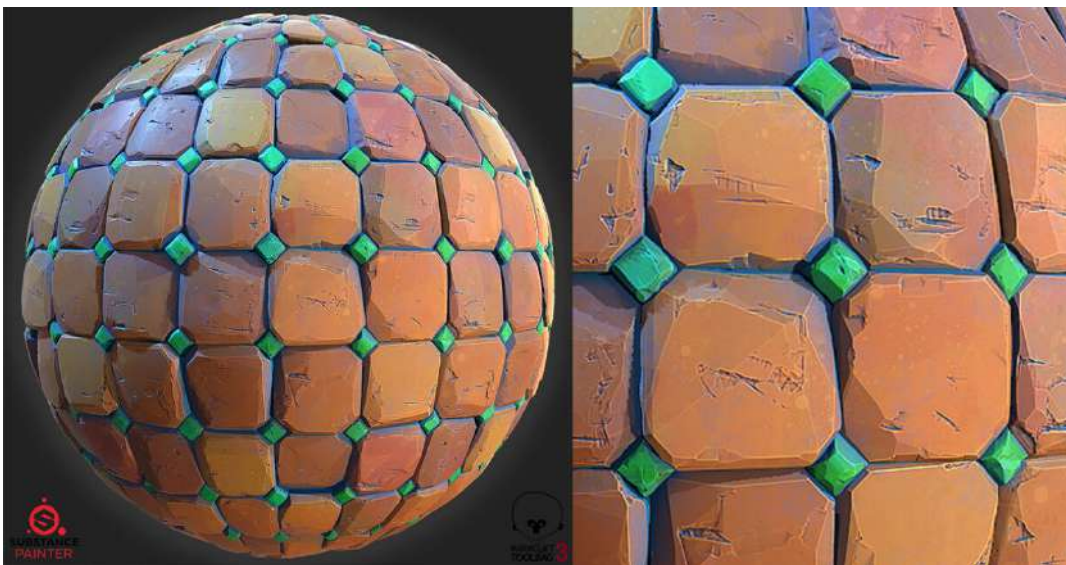
### 5.1.3.2 Workflow

Unlike the traditional non-modular workflow, artists using a modular workflow can create their textures before their final models. As with the texture atlas technique, the UVs of each model will be projected on top of texture images already prepared, and the ability to immediately see textures projected onto the model during this process is one of the main benefits of using this method according to *Thiago Klafke*, Environment Artist at *Blizzard Entertainment*. Using graphics software, the artist precisely lays out the texture, either by creating each within its given space, or creating it separately and combining the textures into one image afterwards.

Additional texture maps are created with the same layout, often using a copy of the original diffuse texture as a base and working on top of it. This process can be quite complicated. Artists may be required to individually cut and alter different parts of the image as appropriate for the texture type they are creating and its intended look in a game engine.

Either before or after the trim sheet, the artist must also create one or more tiling textures. The process usually involves modeling or sculpting with the use of tools or modifiers in graphics software that ensures that the texture repeats seamlessly along its edges. After creating other texture maps as required, the result is an image that can be infinitely repeated across a surface with the full visual fidelity offered by modern texturing techniques.

**Figure 22**  
Example of Tiling Texture



Working with tiling textures and trim sheets has very interesting advantages for the overall asset creation workflow. Primarily, the rigid structure of the traditional non-modular process is reorganized, as modeling and texturing can be done in any order the artist prefers, and both of these as well as UV mapping can be adjusted with minimal consequences. Adjusting the model may require some adjustment of the UVs in the future, but textures will not be affected. Editing the textures will provide instant visual feedback when applied to the model and has no effect on the model or UVs as long as the layout of the trim sheet is not changed. These differences fundamentally change the whole workflow of creating the assets to one that allows iteration at all stages if needed.

## 5.2 Solutions for Modular Environment Art Fatigue

The term 'Art Fatigue' is used to describe the conditions of a player becoming complacent and losing interest in an environment due to excessive reuse of assets, repetitions, or an overall sense of sameness. It is difficult to create an environment asset pack that avoids this feeling. Modular environments are completely dependent on the reuse of assets, so modular workflows typically include steps taken to avoid art fatigue. Neglecting this part of the process can lead to broken player immersion, or in worse cases, the perception that the game is of lower quality than similar products without this issue. Such was the case with some of the criticism leveled at dungeon environments in *The Elder Scrolls IV: Oblivion*.

### Figure 23

Room of *The Elder Scrolls IV: Oblivion* by Bethesda Softworks



This room configuration was placed in many dungeons without any variation in the game. The perception of repetition is noticeable, so the most common way to avoid this is by creating as many different asset variations as possible. Unfortunately, doing so may negatively impact device performance, and most obviously increases the workload involved in creating the asset pack. There is a point at which creating additional assets becomes less economical than using other techniques to minimize art fatigue, and studios must strike a balance between the two.

In order to avoid the art fatigue in my modular environment kit, I have proposed some solutions to make the scenario look more diverse.

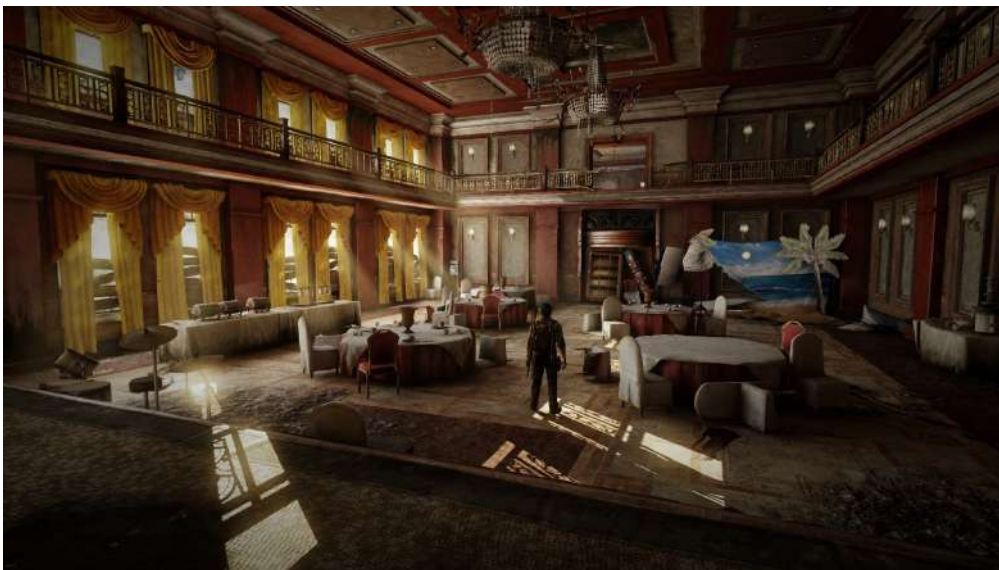
### 5.2.1 Lighting Solution

The way we perceive light in an environment has a strong effect on our interpretation of the space and form of the surroundings. Along with developing techniques for creating assets of higher visual quality, many 3D artists develop their understanding of light, and some studios even employ lighting specialists. It is possible to manipulate mood and perception by placing, tuning, and moving light within a scene.

The color of light and contrast of shadows directly influence our perception of an object's color and texture. Increasingly, game engines are able to realistically represent the behavior of light in space, and modern hardware can handle the calculations required to do so. Nowadays, modern game engines make it possible to calculate high quality lighting and bake it into another form of texture image, a light map, that is applied to all assets within a scene.

#### Figure 24

Scenario full of Modular Elements in *The Last of Us: Part II* by Naughty Dog



### 5.2.2 Hero Asset Solution

Are unique assets created to add context to a specific space. Their main goal is to fit the rest of the assets used in the area but are not intended to be used as a repeatable module.

Assembling an environment from an asset kit sometimes yields a complete and well-optimized space for gameplay, but can also seem to be empty and lacking in detail that differentiates it from similar spaces. For reasons of both aesthetics and gameplay, it can be beneficial to add hero assets. Combining hero assets with a modular environment distinguishes one space from others making the scenario unique and memorable for the player while maintaining the efficiency and flexibility.

#### Figure 25

Modular Environment with a Hero Asset by *Kobus Viljoen*



Hero assets are essential to create a visual interest and break up repetition in modular environments, but it is important to bear in mind that these assets are best placed in areas of low pressure or pacing, to allow players to appreciate the artwork.



### 5.2.3 Decals Solution

A decal is an asset that consists of an extremely simple model, such as a flat plane, that is placed close against the surface of another asset such that the player perceives the decal as an attached detail.

#### Figure 26

Example of Decals (Graffiti)



At its most basic level, the use of decals enables artists to place images arbitrarily on other surfaces, similar to hanging a poster on a wall. Placing decals is an effective technique for covering repetitive details or creating a focal point on an object where one hasn't been modeled.

Decals express the capability of texture images to store visual information in a very direct way. Used well, they lend artists the ability to change the appearance of an asset without creating a full variant of the asset, usually requiring far less work. As a simple individual asset, a decal can be moved or toggled in and out of existence easily by game engines. Whether the decal is used to break up repetition, they can also be a powerful tool for minimizing art fatigue.

### 5.2.4 Extra Props Solution

Apart from the modular elements of the environment it is also important to create props that fit the context of the scenario. In my case, I am doing a cyberpunk city, so paper bins, trash, streetlights, neon signs, etc. are some props that have to be created in order to avoid having a repetitive environment. Assembling the whole scene with some of these props will give a more powerful visual interest and diversity to the city.

**Figure 27**

Extra Props Moodboard of my Cyberpunk Environment



## 5.3 Breakdown of the Project

In this section is documented the whole process I have carried out to create *Neotropic*, my cyberpunk environment.

### 5.3.1 Game References

The cyberpunk environment came into my mind before the release of *Cyberpunk 2077* by *CD Projekt*. It was a theme that I really liked, so I decided to go for it!

Also, the environment art style of *Borderlands* is quite unique. The shader that makes everything look cartoonish is something that has always impressed me.

When you want to create something new it is fundamental to take a look to other games. In my case, I took *Borderlands 3* as my main reference.

#### Figure 28

Reference Game: *Borderlands 3*



#### Figure 29

Reference Game: *Borderlands 3*

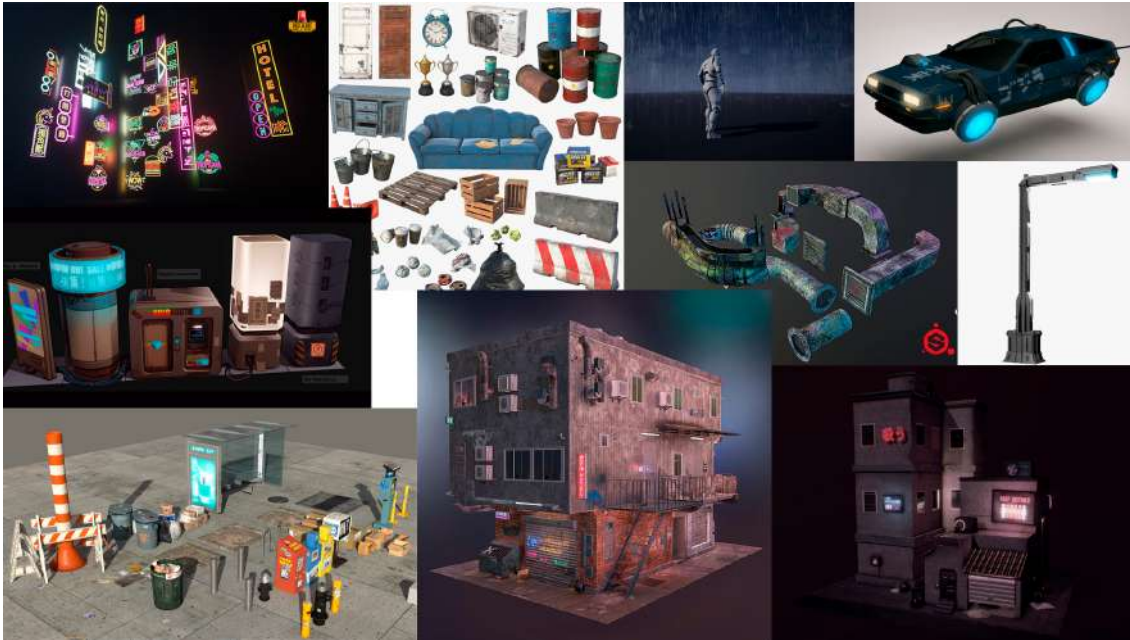




Once this was clear, I started to think about more concrete things. So, after looking through many photo references, I created a moodboard with the elements I wanted to have in my environment kit.







**Figure 30**





Moodboard of *Neotropic* Environment



### 5.3.2 Asset List

**Table 4**  
 General Assets Information

Name	Variants	Image
Building Module	4-6	
Building Accessory (Roof)	2	
Building Accessory (Window)	4	
Building Accessory (Pipes)	4	
Building Accessory (Door)	2	
Emissive Signs	8-10	

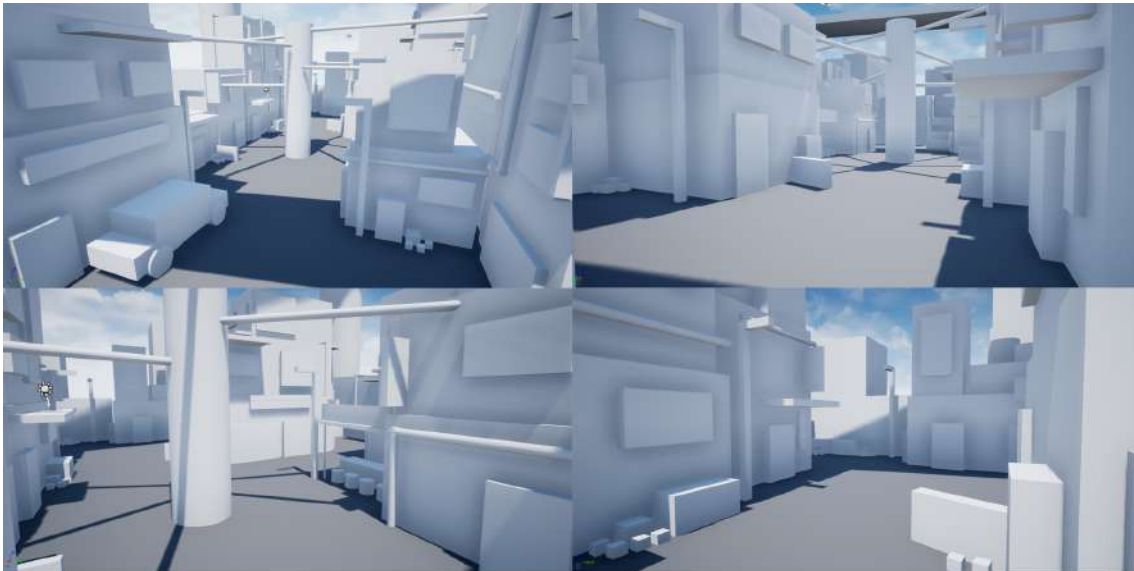
Decals	7-10	
Big Hero Asset	1	
Street Props	1	
Machines	3	

### 5.3.3 Blockout

The idea of the scenario is to make the ‘hero asset’ the focal point of the whole environment. I placed it in the center, so it is surrounded by many buildings. I have tried to add a lot of emissive elements so that these lights in the scene will have a lot of importance since the ambient light will be dark as it will be night time. Moreover, in order to achieve this cyberpunk sensation I have placed many trash elements all over the scenario to make the player feel that the city is not under control, that streets are dirty and it is not safe to walk there.

**Figure 31**

Blockout Renders of the Environment



### 5.3.4 Main Features

In this section I will talk about the most important features of my asset pack and how I carried them out.

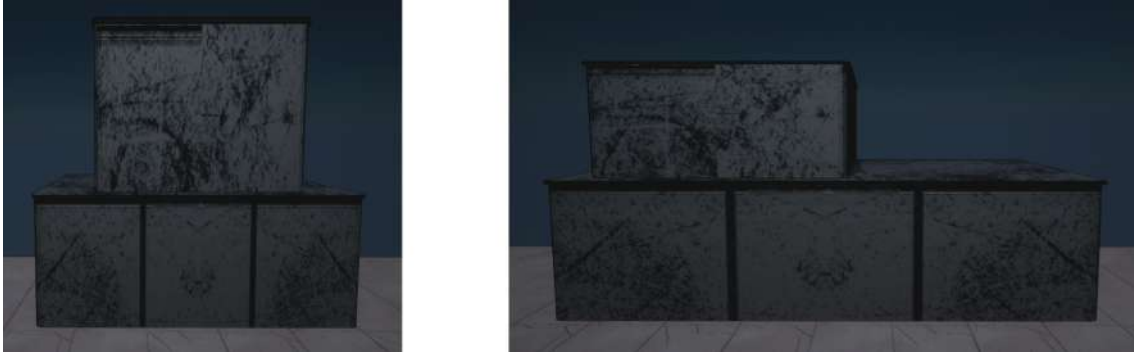
#### 5.3.4.1 Modular and Tileable Buildings, Walls and Railings

Modular and tileable, both concepts are closely related. On one hand, a modular building would not look good without a tileable texture because if we scale it the texture would deformate. And on the other hand, a tileable texture would not make sense without a modular building because if we are not going to scale modules of the building we do not need a tileable texture.

Let's see an example of how a building looks like when we scale it with a non-tileable texture and how it is with a tileable one:

**Figure 32**

Building resized with a Non-Tileable Texture

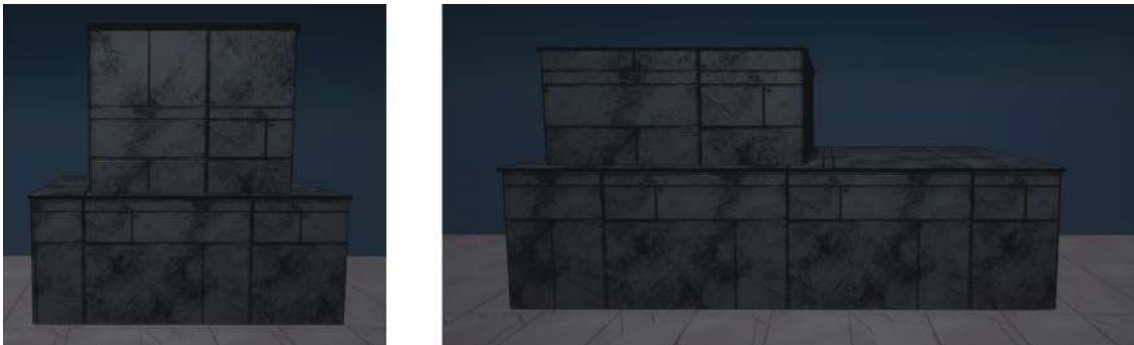


As you can see in the image, the texture gets resized according to the object. This makes the asset look very odd.

As we said, in order to avoid this situation we have to use a tileable texture which is what we did in the project. Let's see how it works in the same building as before.

**Figure 33**

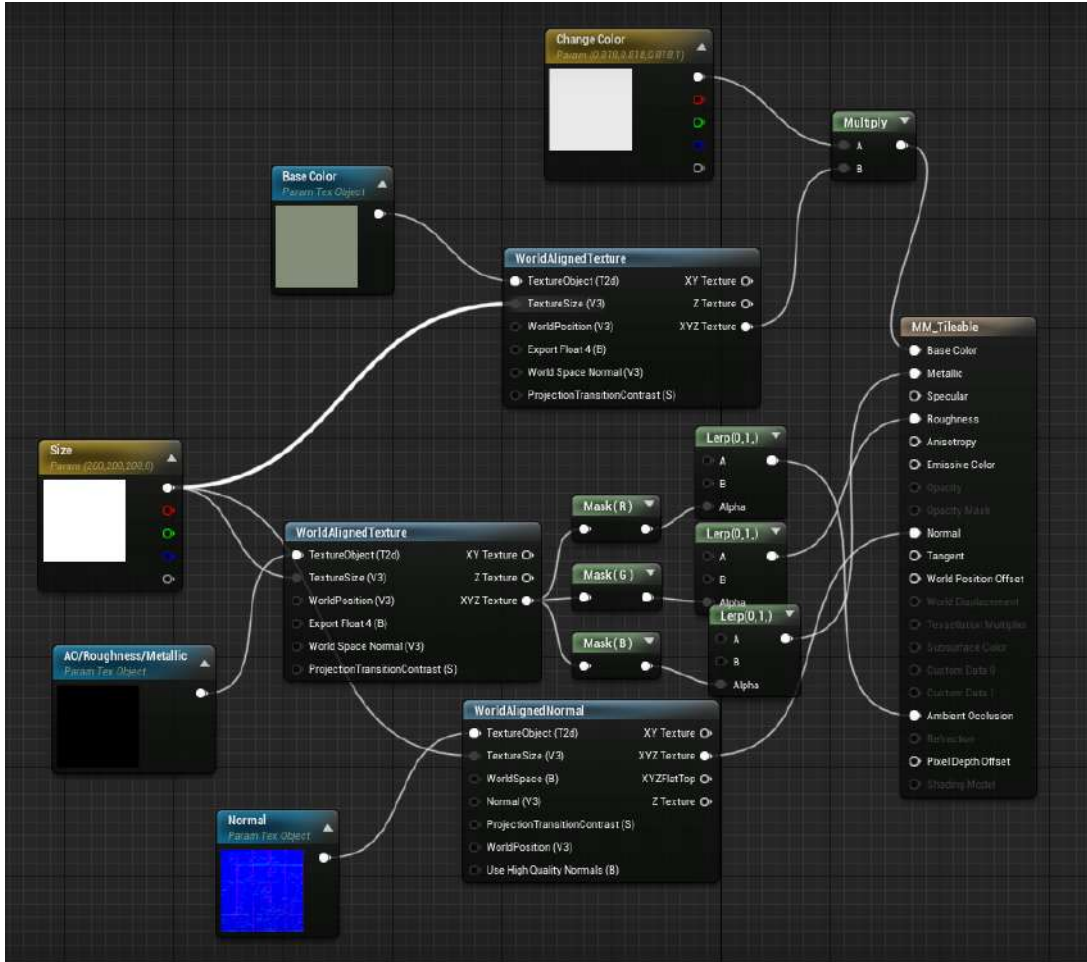
Building resized with a Tileable Texture



We can see that it doesn't matter how much we resize the building, it will always keep the texture with the same aspect. Moreover, with this type of texture we are able to change the size, so we can make all the building textures look different by changing a vector parameter of three numbers (one for each coordinate).

The blueprint we created in Unreal Engine to handle this feature on a master material is the following one:

**Figure 34**  
Blueprint of the Tileable Master Material



The UE function of ‘WorldAlignedTexture’ tiles a texture in world space, so what I did was create three Texture Samples and convert them to a Parameter Texture Object, then I linked them with their corresponding ‘WorldAlignedTexture’, create two ‘Constant3Vector’ as parameters to store the size and the desired color, and finally connect them to the Master Material (MM).

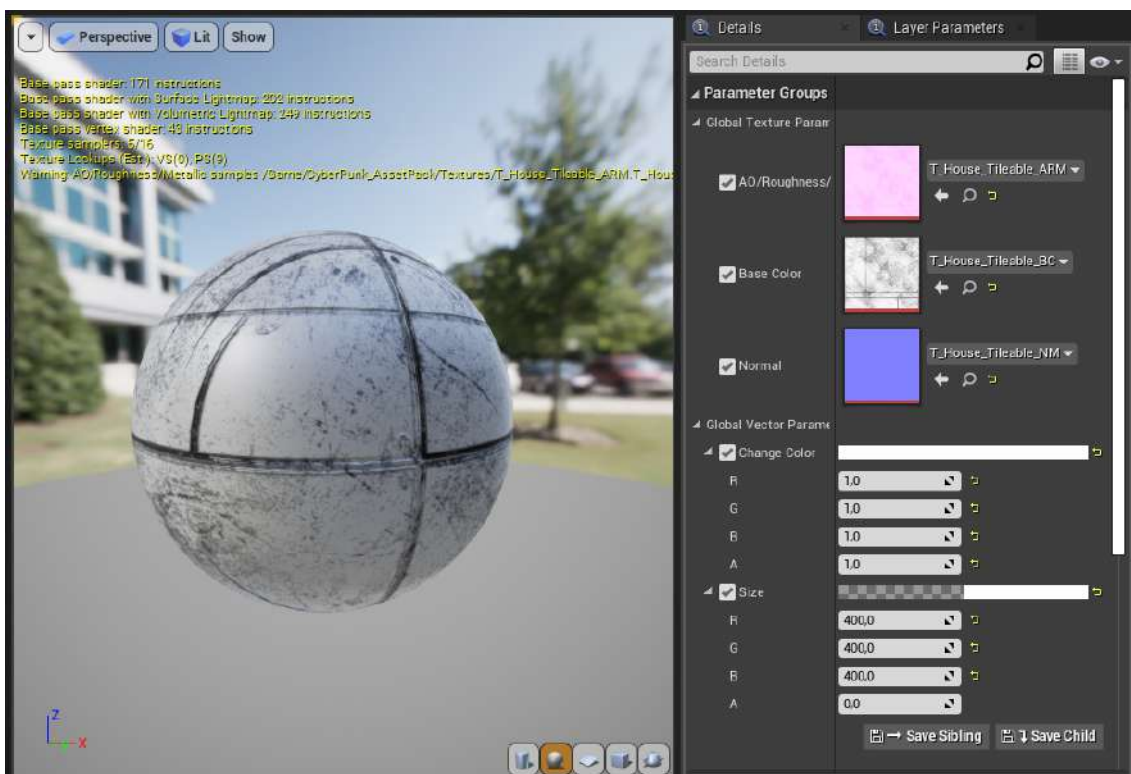
During the development of the project I exported textures from *Substance Painter* with the *Unreal Engine Packed* format. This way I only got three textures (Base Color, Normal Map and Ambient Occlusion/Roughness/Metallic) and it is easier and more optimal for me to handle three files which contain information of five maps.



So, in order to split the three combined textures (AO/Roughness/Metallic) in one file I had to use a component mask to get the corresponding R, G or B information, and then use the LinearInterpolation function to finally connect it with the Master Material (MM).

Once this is done, you can create Material Instances (MI) to customize the textures with the maps and size you want. Following these steps, any material with the Tileable Master Material as parent will have these adjustable parameters:

**Figure 35**  
Details of a Tileable Material Instance



### 5.3.4.2 Custom Decoration

Obviously, a city is made up of more than just some building modules. That is why it is important to create many props in order to provide as much variety as possible to the environment.

Once you have all kinds of common city props, it is important to create Actor Blueprints which will combine some of them to get new props. It is more or less like the modularity of the buildings but with the props. Here is an example:

**Figure 36**

Ceiling Blueprints



Another common technique to optimize time is reusing props but creating new textures. Let me show the vending machines of my project:

**Figure 37**

Vending Machines Blueprints





### 5.3.4.3 Interior Cubemaps

In order to give more depth and sense of realism to the environment I decided to create cubemaps simulating some possible building interiors.

The process I carried out to create them starts by getting some HDRI images of the interior you want to replicate. Then, we go to *3dsMax* and, summarizing, we create a cube where we apply the HDRI texture and then we create a sphere that will wrap around the previous cube. Now, we project the cube texture to the sphere using a reflection material map. And finally, we do some renders and go to *Photoshop*.

Looking to the Unreal Engine Cubemaps Documentation ([Link](#)) we will find this image template:

**Figure 38**

Cubemap Image Template

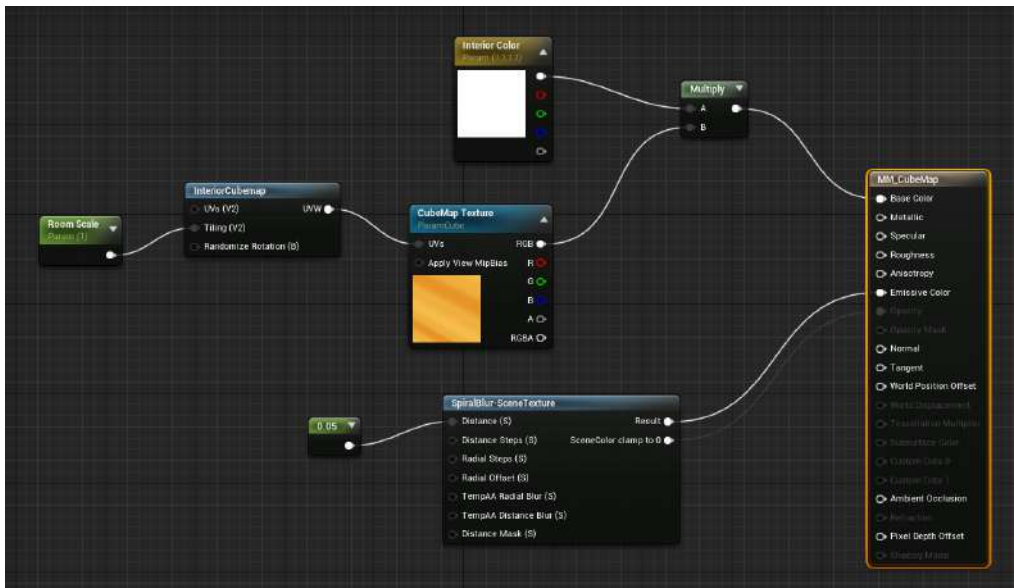


We have to fill the template with the renders exported from *3dsMax*. These renders will have a specification in the name file referring to the orientation (cubemap1\_LT, cubemap1\_RT, cubemap1\_FR, etc).

Knowing this I found out the order of the sequence. From left to right, or from 1 to 6, the order is: Left, Right, Back, Front, Up, Down. And obviously, don't forget to rotate each image according to the rotation of the corresponding number. And finally, we have to save the file with the *DDS - NVIDIA Texture Tool Exporter* format.

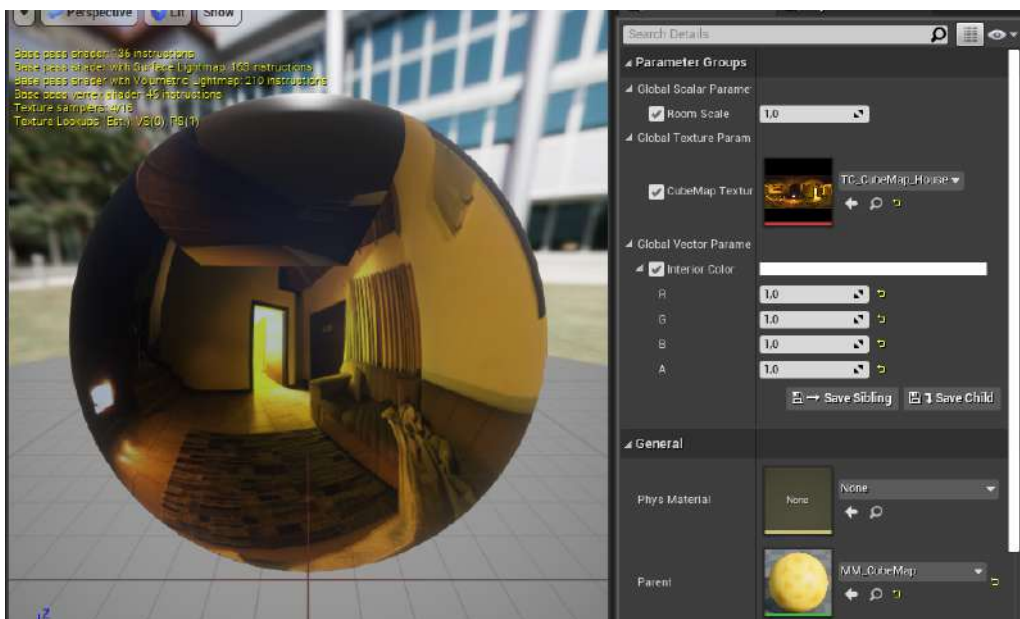
Now we go to our *Unreal Engine* project and we create this Master Material (MM):

**Figure 39**  
Blueprint of the Cubemap Master Material



We are using the UE function called 'InteriorCubemap' which converts the DDS file we exported from *Photoshop* to a standard texture that we can use in any geometry shape. Moreover, I also created a 'Constant3Vector' parameter that allows me to change the color of the interior and a scalar variable as a parameter to adjust the room size.

**Figure 40**  
Details of a Cubemap Material Instance

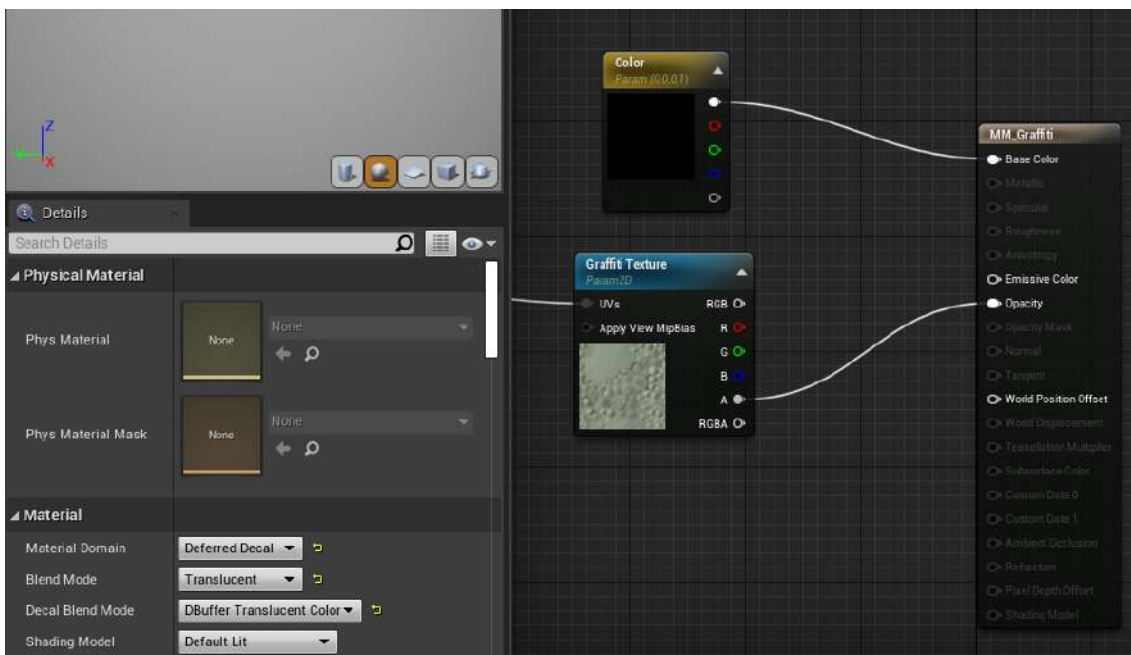


### 5.3.4.4 Colorizable Decals

It is a simple Master Material (MM), easy to create and very useful for the Cyberpunk theme. Basically, it consists in creating some decals in Photoshop with a transparent background. Then, import the textures to your *Unreal Engine* project and create a material that gets a texture from a parameter, a 'Constant3Vector' that handles the color of the material. The most important thing is the blend configuration of the material which have to be like this:

**Figure 41**

Blend Configuration of the Decal Master Material



### 5.3.4.5 Dynamic and Emissive Signs

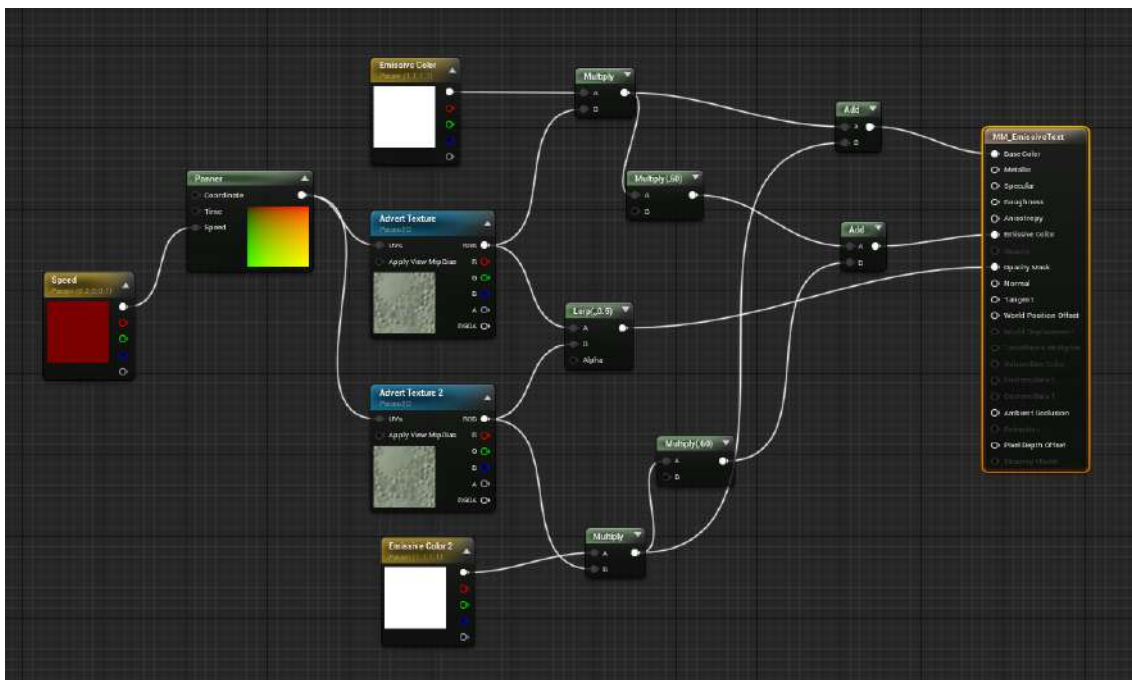
To carry out this feature I used alpha textures. So, first we need to change the Blend Mode of our Master Material (MM) to 'Masked'. This action will enable the Opacity Mask input node and we will be able to use the alpha textures that we are going to create.

The Emissive Master Material (MM) is able to support two different emissive colors, one for each Texture Sample Parameter which will be the alphas used for each sign.

This Texture Parameter will be directly connected to the Opacity Mask of the material but also to a Multiply function that will handle the color material. In order to make the material dynamic we will use the 'Panner' function connected to the UVs node of the Texture Sample and a 'Constant3Vector' parameter which will determine the moving speed of the texture.

**Figure 42**

Blueprint of the Emissive Master Material



### 5.3.4.5 Toon Shader

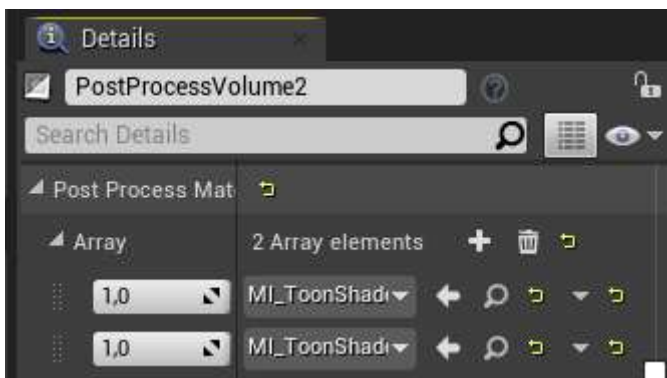
To make the environment look like *Borderlands 3* we need a toon shader. We will be able to adjust the shader values to make it look like we desire. We will need two Master Materials, the first one that will affect everything, and the second one that will remove the shader to the sky.

I followed a Youtube Tutorial which you will find in the References section. But, summarizing, we will mainly use the UE function called 'SceneTexture' which will need inputs and outputs that will control the depth difference between neighbours pixels.

In order to support this post processing effect we will need a 'PostProcessVolume' element in the scene that stores both materials in his array of elements. It is important to check the option called 'Infinite Extent' because it will apply the effect to the whole environment instead of only inside the box of the element.

**Figure 43**

Array storing the Toon Shader Materials





Just to compare how it looks with the shader and how would look without it I have done a test and here is the result:

**Figure 44**

Comparison: Without Toon Shader vs With Toon Shader



As you can see in the image, the shader absorbs most of the illumination and creates a little stroke around every mesh. It is important to say that the shader is modifiable, we can adjust the distance from the camera we apply the effect, change the color of the strokes and set the size of the stroke as we wish.

### 5.3.4 Demo Scene

After finishing all the assets it is time to create a demo scene. It is important to do it because during the process of creation you realize that there are small details to be improved, such as some pivot points position or any other error in the texture due to the UV's of the mesh. So, in my case, during this step I also took the opportunity to improve some details and change things that I did not like at all.

**Figure 45**

Render of the Demo Scene



**Figure 46**

Render of the Demo Scene





**Figure 47**

Render of the Demo Scene



**Figure 48**

Render of the Demo Scene



### 5.3.5 Submit Content in UE Marketplace

To submit a package to the *Unreal Engine Marketplace* we need to follow their guidelines for file nomenclature and folder organization.

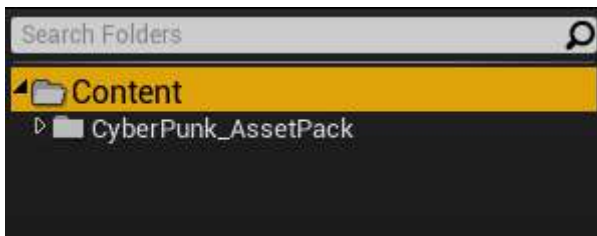
In my case, I didn't know about the existence of these guidelines, so I had to change every file name of the project. It was quite tedious, so I encourage you to do it from the beginning.

#### 5.3.5.1 Folder Organization

First of all, we create our folder inside 'Content'. This folder will be the root of our project, so it has to be the only existing folder in 'Content'.

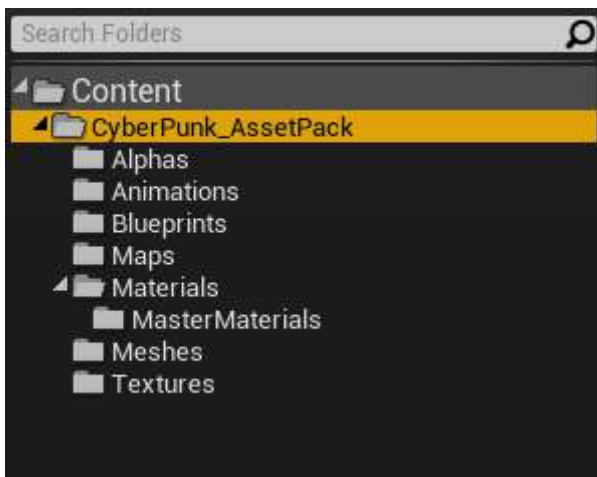
**Figure 49**

'Content' folder of the Project



**Figure 50**

'CyberPunk\_AssetPack' folder of the Project



As you can see, the 'CyberPunk\_AssetPack' folder contains all the assets that I have created and used to build up the final environment.

### 5.3.5.2 File Nomenclature

Obviously, each folder contains what the folder name itself indicates but I have established some rules in the nomenclature of the files:

**Prefix** and **suffix**:

\* *Italic* means that it is optional

**Alpha** → **A**\_Name\_Number\_*Part*

**Animations** → **AN**\_Name

**Blueprints** → **BP**\_Name\_*Number*

**Material Instances** → **MI**\_Property\_Name\_*Number*

\*where *Property* can be: Tileable, Neon, Decal, etc.

**Master Materials** → **MM**\_Name

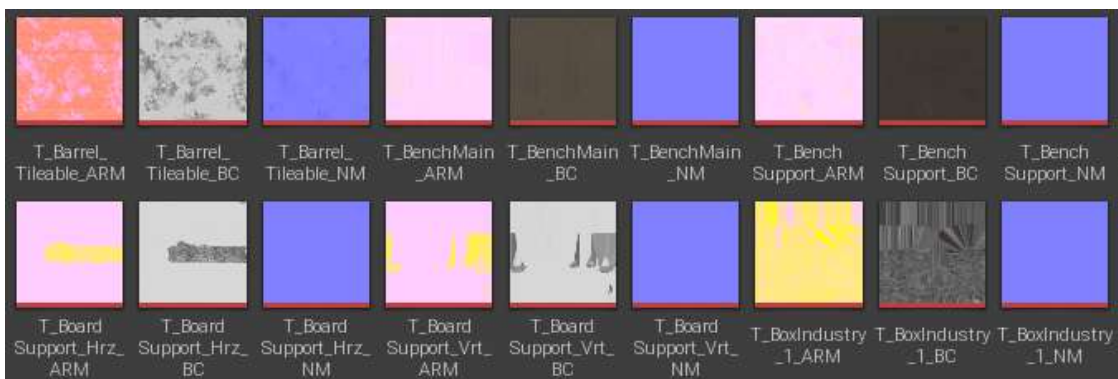
**Static Meshes** → **SM**\_Name\_*Number*

**Textures** → **T**\_Name\_*Number*\_Map

\*where Map can be: BC (Base Color), NM (Normal Map), ARM (Ambient Occlusion/Roughness/Metallic)

**Figure 51**

Nomenclature Example



## 6. Conclusions

First, I would like to do a little retrospective. I think it is important to do it when you finish a project because you get conclusions and knowledge that will be important for future projects. In my case, I think that I wasted some time redoing things that now I already know how to handle, so for the next project I will be able to avoid these errors and optimize my time. Now I am also aware of what it takes to carry out such a project, I did not know that organizing and checking the tasks done and to be done takes a big part of your time but it is totally worth it otherwise the project would become a chaos.

When we proposed the goals to achieve before starting the project I thought that it was not very difficult, that it was just a long-term project and I just needed to be constant. But on the contrary, I have encountered tricky problems that I did not expect but I finally could overcome them. In the end, these problems are what make you learn and improve.

Taking a look at the goals I can say that I have been able to accomplish them. I have provided a modular environment to UE Marketplace, although it is still in the approval process. In my opinion, I have also made a good document of the project where I have explained everything learnt about the topic. About the asset pack, my goals were to do a good-looking and optimized environment, and I firmly believe that I have achieved it.

Finally, on a personal level, the project has meant a lot to me. I am very happy with the result, I know that it is difficult to compete in the UE Marketplace as there are asset packs made by professional teams that, obviously, are better than mine. But apart from this I am very happy with the result because I wanted to make an environment similar to Borderlands and we can see in the image above that this objective has been fulfilled.

### Figure 52

Comparison: *Borderlands* vs *Neotropic*



For future versions of the package, I will create more props and adjust some little details to ensure that users that got my asset pack are satisfied. As I said before, I could not submit it to UE Marketplace because the process is a bit long but it will be there very soon.

## 7. References

Nataska Statham, Joao Jacob, Mikael Fridendalk. “Entertainment Computing”.

ScienceDirect. March 2022

<https://www.sciencedirect.com/science/article/pii/S1875952121000732>

Mike Bernstein. “Modularity in Creating 3D Game Environments”. Theseus. May 2017.

[https://www.theseus.fi/bitstream/handle/10024/146758/Bernstein\\_Mike.pdf?sequence=1&isAllowed=y](https://www.theseus.fi/bitstream/handle/10024/146758/Bernstein_Mike.pdf?sequence=1&isAllowed=y)

Scott Jones. “Investigation into modular design within computer games”. Polycount.

May 2011.

[http://wiki.polycount.com/w/images/7/70/Investigation\\_into\\_modular\\_design\\_within\\_computer\\_games\\_v1.0.pdf](http://wiki.polycount.com/w/images/7/70/Investigation_into_modular_design_within_computer_games_v1.0.pdf)

Paul Mader. “Creating Modular Game Art For Fast Level Design”. GameDeveloper.

December 2005.

<https://www.gamedeveloper.com/production/creating-modular-game-art-for-fast-level-design>

Richard Vinci. “Victorian Street Blueprint and Modularity Demonstration”. Youtube

Video. June 2017.

<https://youtu.be/0sgOGB8ZrZQ>

Lea Kronenberg. “Balancing Modularity and Uniqueness in Environment Art”. Beyond Extent. September 2020.

<https://www.beyondextent.com/articles/balancing-modularity-and-uniqueness-in-environment-art>

Vitaly Zhdanov. “How to Create a Modular Environment Design: Abandoned Soviet Warehouse”. CG Master Academy. November 2020.

<https://blog.cgmasteracademy.com/how-to-create-a-modular-environment-design-abandoned-soviet-warehouse/>



Nathan Alderson. “Making a Stylized Cyberpunk Scene in Blender and UE4”. 80lv. September 2021.

<https://80.lv/articles/making-a-stylized-cyberpunk-scene-in-blender-and-ue4/>

Kim Aava. “Realistic vs Stylized: Technique Overview”. 80lv. December 2017.

<https://80.lv/articles/realistic-vs-stylized-technique-overview/>

HpdizzoArt. “Modular CyberPunk City Asset Pack”. UE Marketplace Content. November 2019.

<https://www.unrealengine.com/marketplace/en-US/product/modular-cyberpunk-city-asset-pack?sessionInvalidated=true>

Clara Tan. “The Workflow Behind an Abandoned Bar Modular Environment”. 80lv. November 2021.

<https://80.lv/articles/the-workflow-behind-an-abandoned-bar-modular-environment/>

Unreal Engine Documentation. “Using Texture Masks”.

<https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/Materials/HowTo/Masking/>

Unreal Engine Documentation. “Creating Cubemaps”.

<https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/Textures/Cubemaps/CreatingCubemaps/>

Aniyal-Environment Artist. “Foggy Night UE5 Beginner Tutorial”. Youtube Video. May 2022.

<https://youtu.be/WC50UEj9i5E>

Dean Ashford. “UE4 - Tutorial - Hologram Material”. Youtube Video. April 2017.

<https://youtu.be/wlz6iyIOkys>

GomVo Developer. “Celshading Effect”. Youtube Video. September 2019.

<https://youtu.be/XRUi60ldycM>