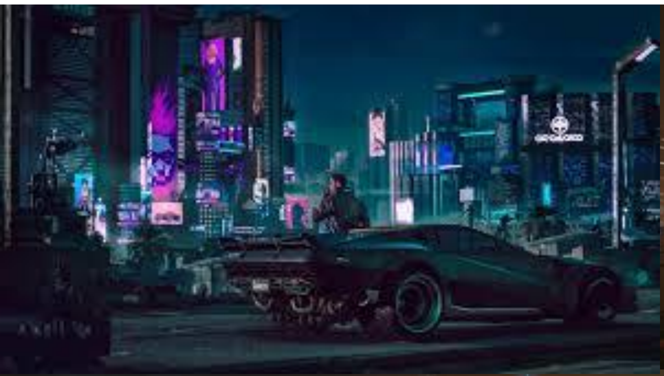
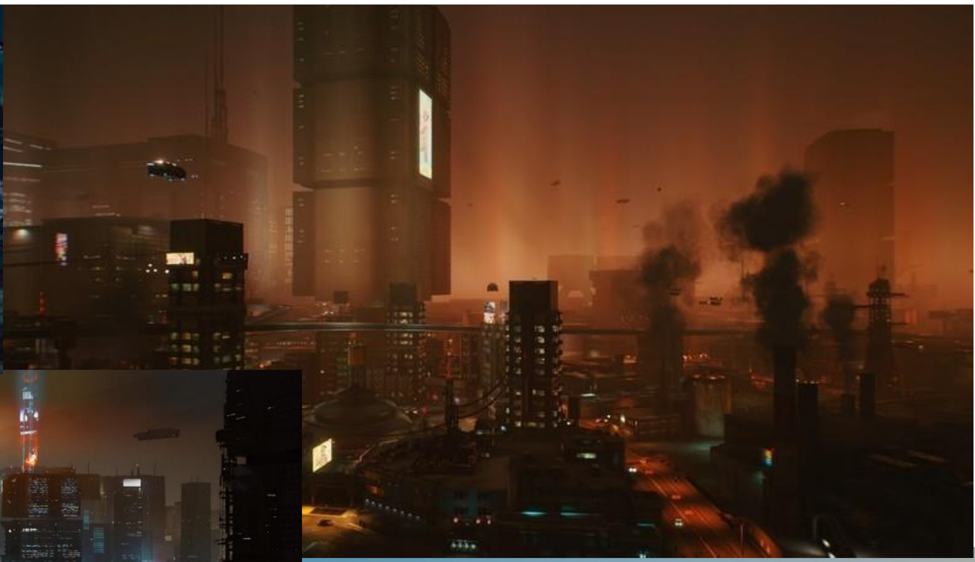
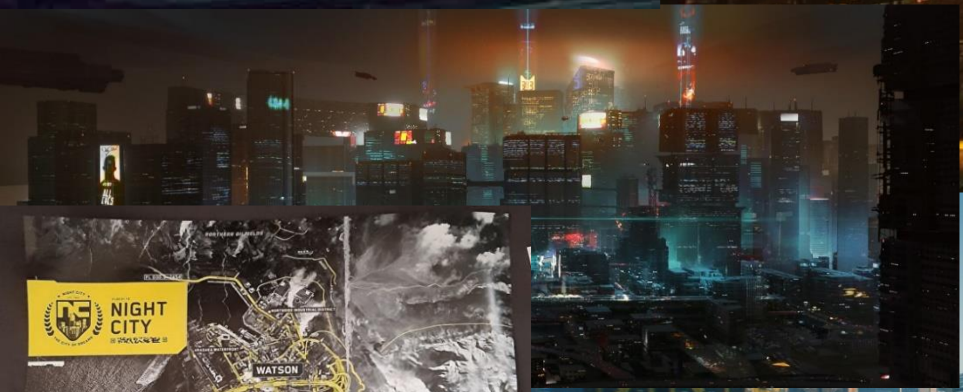
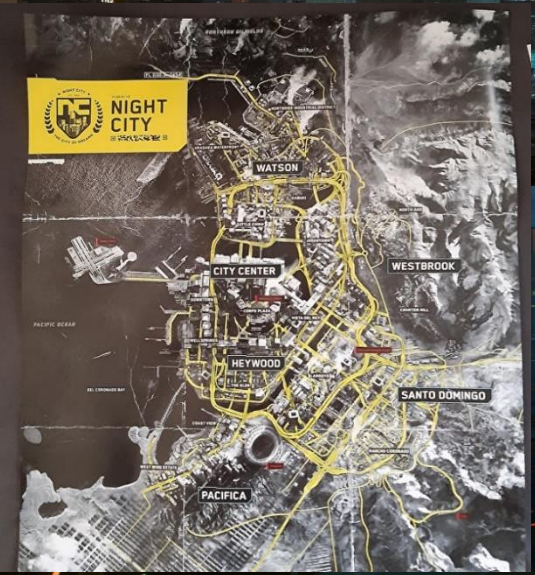
Research Procedural Art

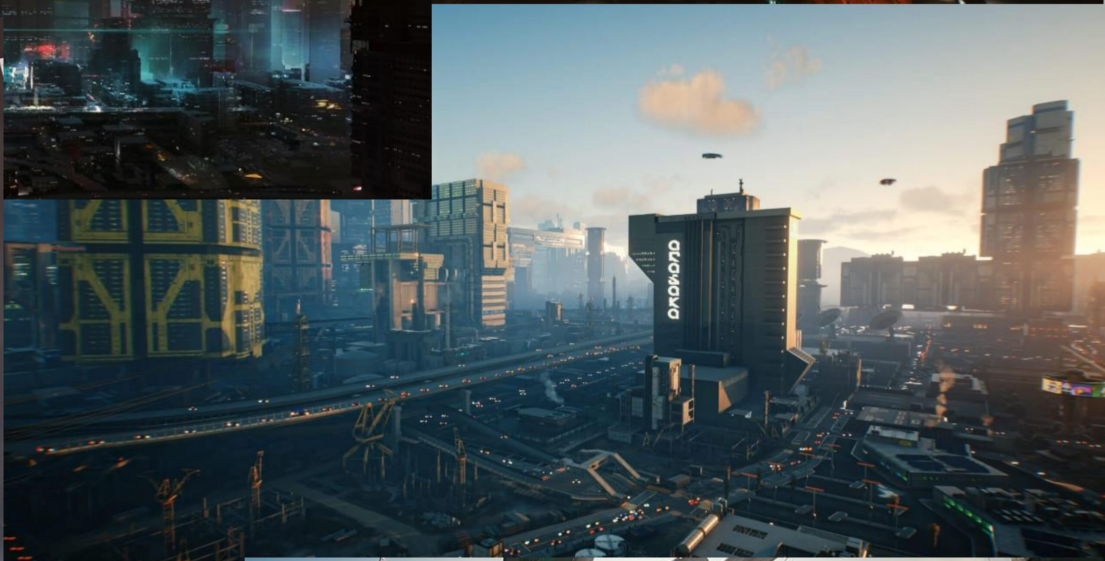
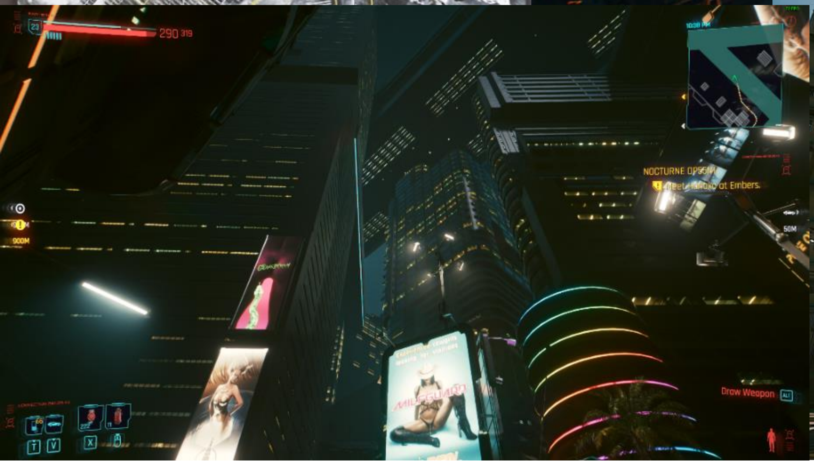
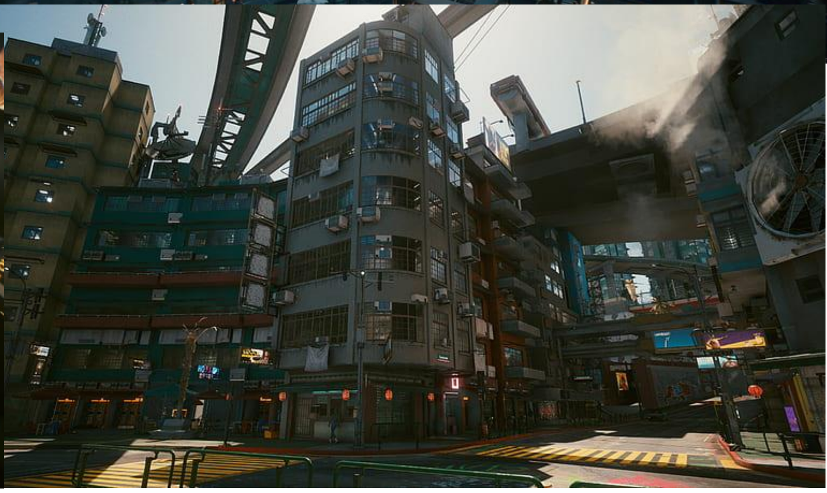
Nils Meijer 466301

Chosen theme: Downtown Night City

Time of day: preferably dynamic, otherwise deep night or sunset.

Visual references:

OPEN
DOTONBORI

Reference analyzation:

During the night, when the city is less active, the insane amount of light emitting from the buildings replaces the sun. Colour palettes that are rather common in a cyberpunk city *during night* are combinations of blue, purple, pink, white, green. There is a certain "glow" present slightly above the city, also known as "light pollution". It allows for a very artificial and active mood, characteristic for the city that never sleeps.

Architectural elements

The skyscrapers consist of countless tiny windows and are relatively straight with some exceptions. Sometimes, they are connected by the use of huge bridges/corridors high in the sky. Stacks often look like they have been duplicated multiple times, with minor adjustments each iteration.

City shape

Street-wise/city layout-wise, the organization is very structured, with streets often being parallel to each other, especially in the city center, sometimes causing quite sizable "grids" of streets.

Materials

The buildings often have a metallic material, reflecting the light cast upon them. Depending on the location of the building for which the material is used, the metal/glass is either squeaky-clean (corporate buildings, part of the elite/upper class of the city), or dirty and covered in dust/smudges (poorer areas of the city). A concrete material is also rather common, often being used for walls, next to metal.

Procedural techniques/tools

In the field of AI, there is a certain formula called the “Utility Function”. It can be used as a way for an NPC in a game to determine what their target should be, based on a set of parameters.

$$U\ (d,t) = \left(1 - \left(\frac{d}{S}\right)\right) * Dt + Wt$$

d = distance from this NPC to the target
t = type (what’s the type of this NPC?)
S = Sight (how far can the NPC see?)
Dt = Distance factor for Type *t*
Wt = Weight factor for Type *t*

Tools required for a cyberpunk style city/what am I going to work on:

Node placement system

A node placement system, to determine where roads are located. Based on this pre-defined roadmap, the city blocks are generated, a rough layout of the city is created. You can place/remove, move, connect/disconnect all nodes at will, giving a very large playing field, limited only by the user’s hardware specifications (can it handle that many buildings etc?)

These nodes represent intersections (except that the traffic lights are non-existent, very unsafe). At the same time, these connections between nodes represent the streets themselves. When all desired nodes have been placed, the city layout will be generated (where are all the city blocks located?). In figure 1, you can see a visualization. The black dots (as said in the image, though possibly not readable due to my handwriting) represent the nodes (simple GameObject with a MeshRenderer, collider and Node script), and the red strokes represent the streets which get created because of the nodes being connected to any given node. Based on the resulting map, the city blocks are generated (which is what the numbers 1/8 represent, although 8 city blocks is an entirely arbitrarily number of course).

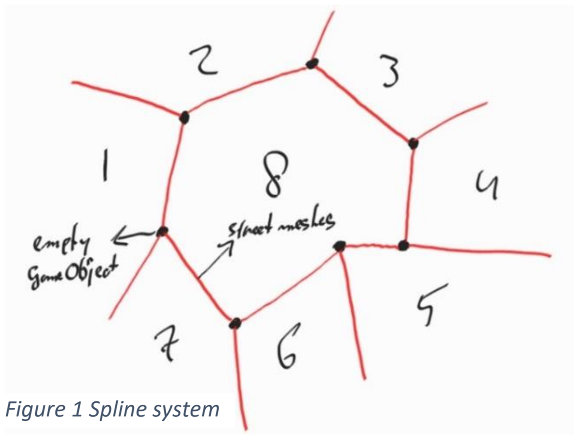


Figure 1 Spline system

To correctly determine where road meshes should be instantiated, we’ll have to take the connected nodes into account. Every node will receive an “intersection mesh”, which is a simple plane. Of both the intersections, I find the vertices of those planes that are closest to each other and based on those vertices, generate a procedural road mesh.

Building procedural generation

To generate the procedural buildings themselves, the CityBlockGenerator can choose from 2 types of buildings, Houses and Skyscrapers. I can create any number of prefabs for those buildings, and a random building will be chosen based on the building type.

 I can create a new Skyscraper prefab, add a set of stacks/building blocks to that prefab, and the script can only choose from those blocks. This should prevent the “bowl of oatmeal”, because it’s impossible for *all* stacks to be mixed (except if the user chooses to only

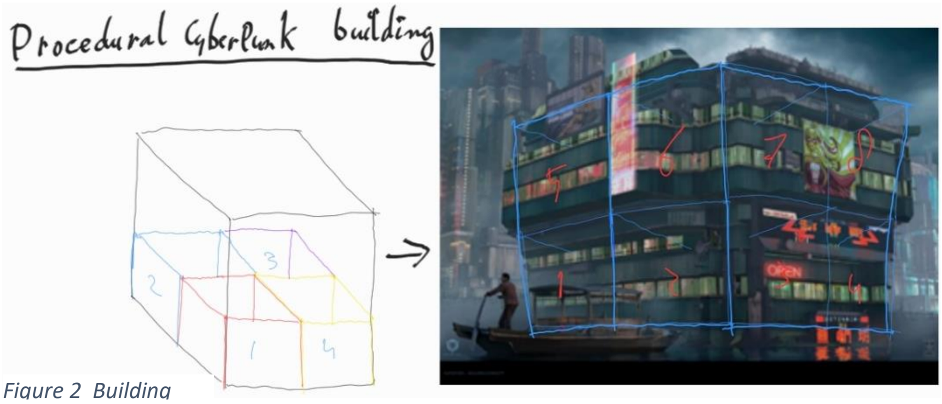


Figure 2 Building

have 1 building prefab for each type for example.) A building consists of 3 stack types: a floor, middle stacks, and a roof (although in my end-result only middle stacks and roofs are used due to time constraints for the artist I'm working together with). *After generation*, the user is able to select any city block > any building in that city block > any stack/building block in that building and swap that stack with other stacks to see how they fit within the building. This allows for complete control & flexibility.

A stack can also receive a certain amount of "billboards" assigned to them, which is fitting for the cyberpunk style. By default, these billboards are only enabled for skyscrapers but if the user wishes, they can of course enable them for houses as well.

City procedural generation

As mentioned before (Procedural Techniques/Tools), I am planning to adjust the "Utility function" (often used for AI characters in games) to determine what kind of building should be generated (tall skyscraper in the center of the city, or a civilian property with only a few floors and a different appearance than a corporate building). See Figure 2 to see a visualization of this. During the 8-week project in term 4 last year, I implemented this formula for a "skunk NPC".

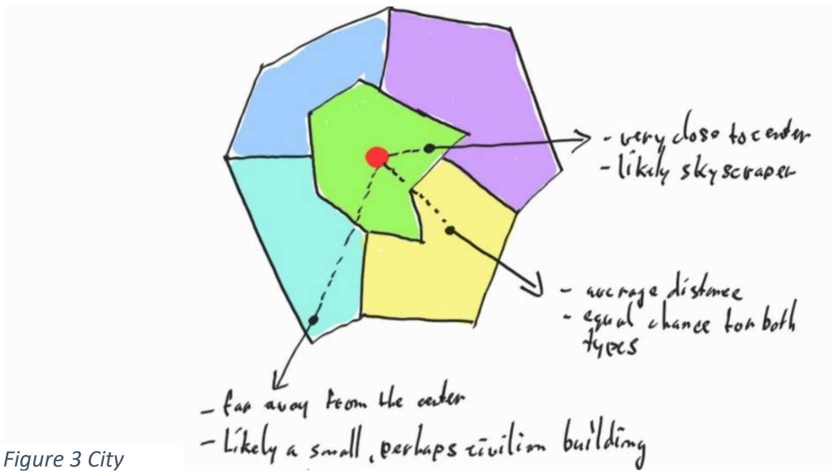

- very close to center
- likely skyscraper
- average distance
- equal chance for both types
- far away from the center
- Likely a small, perhaps civilian building

Figure 3 City

<u>Utility function</u>

Formula given during the Advanced Tools course, with changes done by me to adjust to my idea:

$$Utility\ value\ (distance, currentBuildingType) = \left(1 - \left(\frac{distance\ to\ center}{maxDistanceToCenter}\right)\right) * \text{distanceFactor} + \text{typeFactor} + \text{randomValue}$$

distance = distance to the center (red dot)
currentBuildingType = type (e.g. 70% is corporate/skyscraper, 30% is civilian/housing)
maxDistanceToCenter = Sight (distance from the center of the city to the most outer node)
distanceFactor = Distance factor for Type *t*
typeFactor = Weight factor for Type *t*
randomValue = a random integer number between a given set of numbers, so that there shouldn't be a clear difference in building types.

For each new city block, the user can input new values for this formula. However, after doing some tests, I found it takes quite some practice/effort to fully understand the impact of every variable in the formula so it can be challenging to find the skyline you want. However, with practice this should work out fine. With the custom editor, the user can destroy and recreate the city block quite quickly as well, so they don't have to restart their entire process because they weren't happy with one city block. To make sure there is still diversity in that block, every building has a randomized value that's included in the utility function, so it's still *possible* a skyscraper is slightly further away from the center and a house is slightly closer, but only as an exception. For each building in a city block, the formula should be executed twice for each type to determine what type has priority. And by using this formula and changing the values for every city block (for example, setting very high values for skyscrapers so you only get those, or setting equal values so you get a 50/50 ratio), you can create unique and distinguishable neighbourhoods.

Example:

- Type weight = desired ratio: skyscraper weight: 200 | house weight: 600
- Distance factor = desired ratio: skyscraper weight: 16 (more chance if it's close to the center) | house weight: 4 (less chance if it's close to the center)
- I changed the division "distanceToCenter / maxDistanceToCenter" to a subtraction because it gave better results.
- Highest value has preference.
- maxDistanceToCenter will be the distance from the center of the city to the top left corner (arbitrary choice, distance to all corners in a rectangle is of course the same).

Example:

Blue = distance
Red = non-preferred building type for distance
Green = preferred building type for distance

$$\text{value}\ (10, \text{house}) = \big(1 - (10 - 75)\big) * 4 + 600 = 864$$

$$\text{value}\ (10, \text{skyscraper}) = \big(1 - (10 - 75)\big) * 16 + 200 = 1256$$

$$\text{value}\ (50, \text{house}) = \big(1 - (50 - 75)\big) * 4 + 600 = 704$$

$$\text{value}\ (50, \text{skyscraper}) = \big(1 - (50 - 75)\big) * 16 + 200 = 616$$

$$\text{value}\ (65, \text{house}) = \big(1 - (65 - 75)\big) * 4 + 600 = 644$$

$$\text{value}\ (65, \text{skyscraper}) = \big(1 - (65 - 75)\big) * 16 + 200 = 376$$

As you can see in the example calculations, the higher the distance to the center, the more preference there is for the House type. The other way around, the closer to the center, preference for skyscrapers grows.

Contributions & collaborations

For the programming part I made the entire tool myself (which is mandatory anyway, as we're not allowed to use anything from fellow engineers). As for assets, I'm using the models, materials and post-processing of Stephanie Temmink.

Sidenote: because I am not able to show the full functionality of my tool with the game-ready assets, I will provide another scene with my "prototyping" assets (a set of cubes/stacks with different materials) that are able to represent my tool better because there is more choice in building blocks (after-generation building editing).

I downloaded the skybox from the following link: https://polyhaven.com/a/dikhololo_night