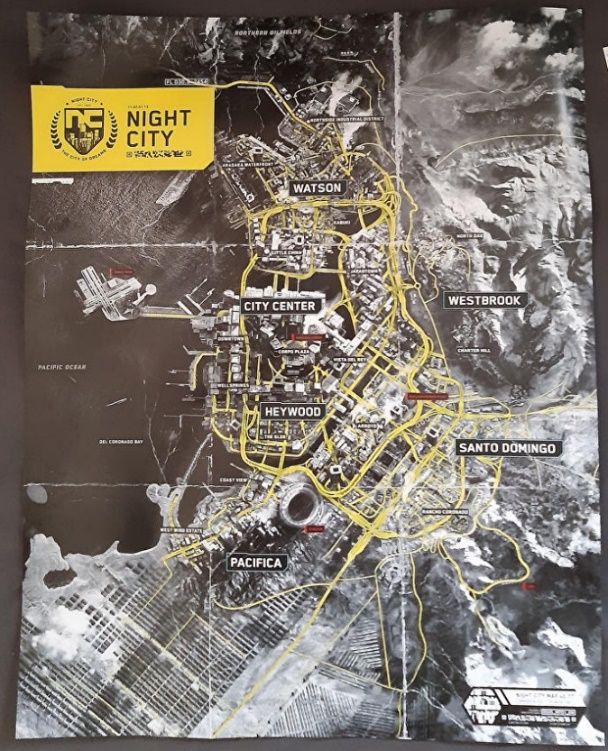
Research Procedural Art Nils Meijer 466301

**Chosen theme:** Downtown Night City

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

A picture containing text, indoor

Description automatically generated**Visual references:**

**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. Floors 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.

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:**

**Spline placement system**

A spline placement system, to determine where roads are located. Based on this pre-defined roadmap, the city blocks are generated. Depending on the shape of the available space, the proper building should be generated for each place in the city block.

Using the location of the city block, the height of buildings should be modified. The closer a block is to the city center, the higher a building has a chance to be. This should give a nice skyline, with the tall buildings mainly focused in the middle of the city.

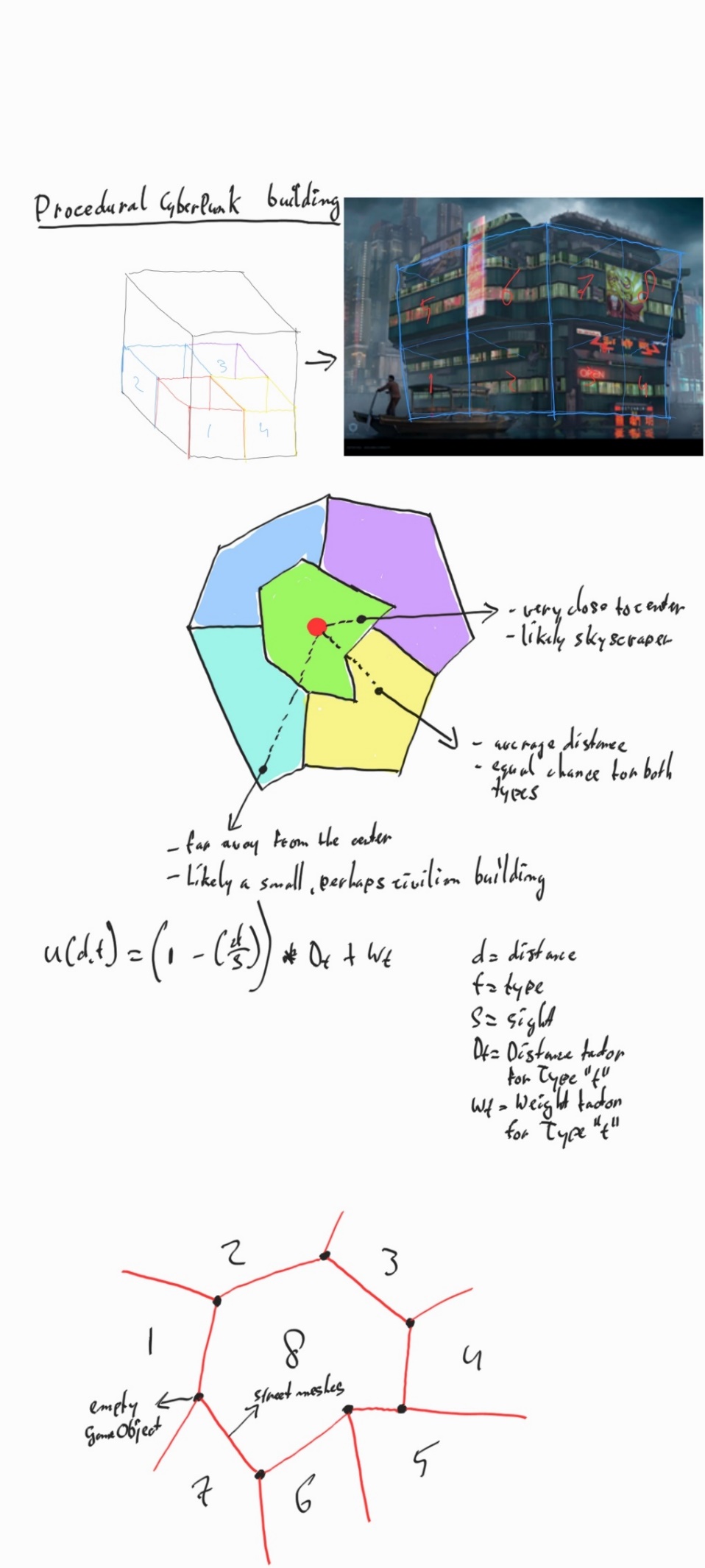


Figure 1 Spline system

This system is used to place so-called “nodes”, which connect streets together. 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 (empty GameObject) that can be placed by me, and the red strokes represent the streets which get created because of the nodes being connected to any given node (determined by the connections made by me). Based on the resulting map, the city blocks are generated (which is what the numbers 1/8 represent, although 8 city blocks is entirely arbitrarily.)

To correctly determine what road meshes should be instantiated, we’ll have to take the connected nodes into account. Based on the relative angle of the connected node, we have to select a specific road mesh. Having a strange angle is very likely, and I do not have enough time to figure out how to make procedurally generated road connections intersections as well. So instead, after the user has placed a rough outline of what the city map should look like, the algorithm will pick the first node in the list and spawn in the correct intersection mesh (for example, with 2 or 3 connections to other nodes). Then, we have to slightly adjust the position of the connected node so that a perfect angle is achieved, resulting in seamless road-intersection connections.

**Building procedural generation**

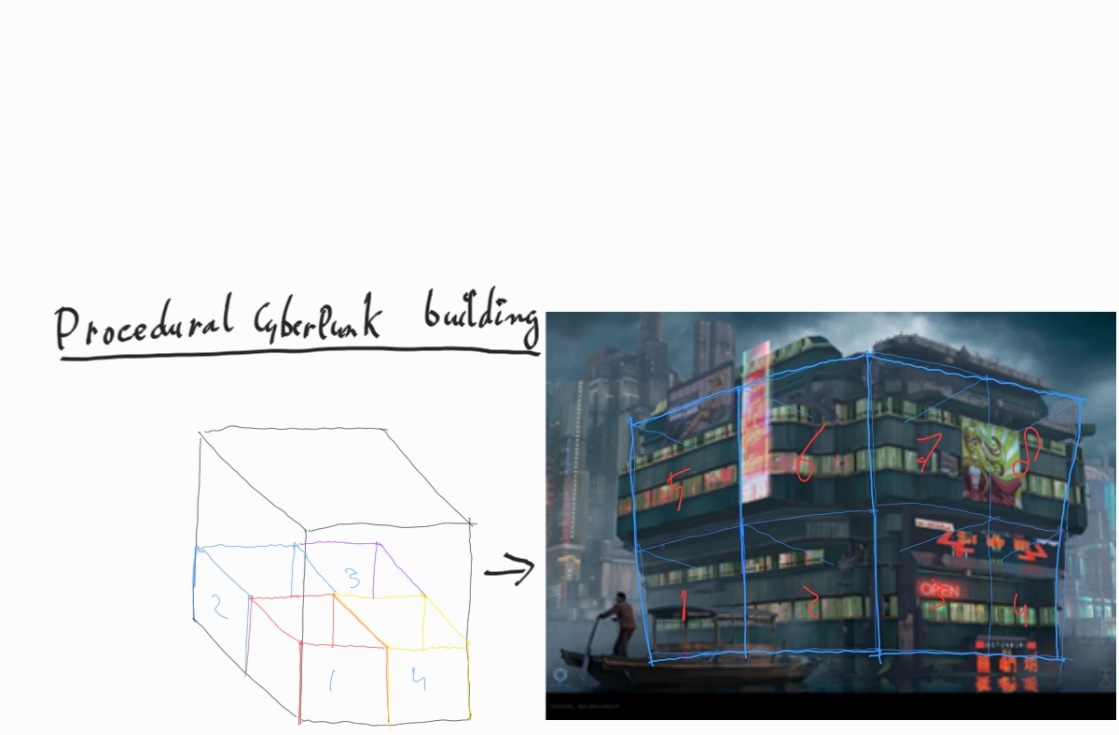


Figure 2 Building layout

To generate the procedural buildings themselves (because not only do I want the city blocks to be procedural, the buildings should be unique as well instead of choosing from a list of preselected complete buildings), they will consist of multiple different components (as seen in Figure 1). Depending on the parameters given by me, fitting prefabs will be chosen. After the full city has been generated, I should be able to modify individual buildings by selecting specific “stocks” of the building, and either choose a new one from a list, and/or regenerate the building itself entirely.

**City procedural generation**

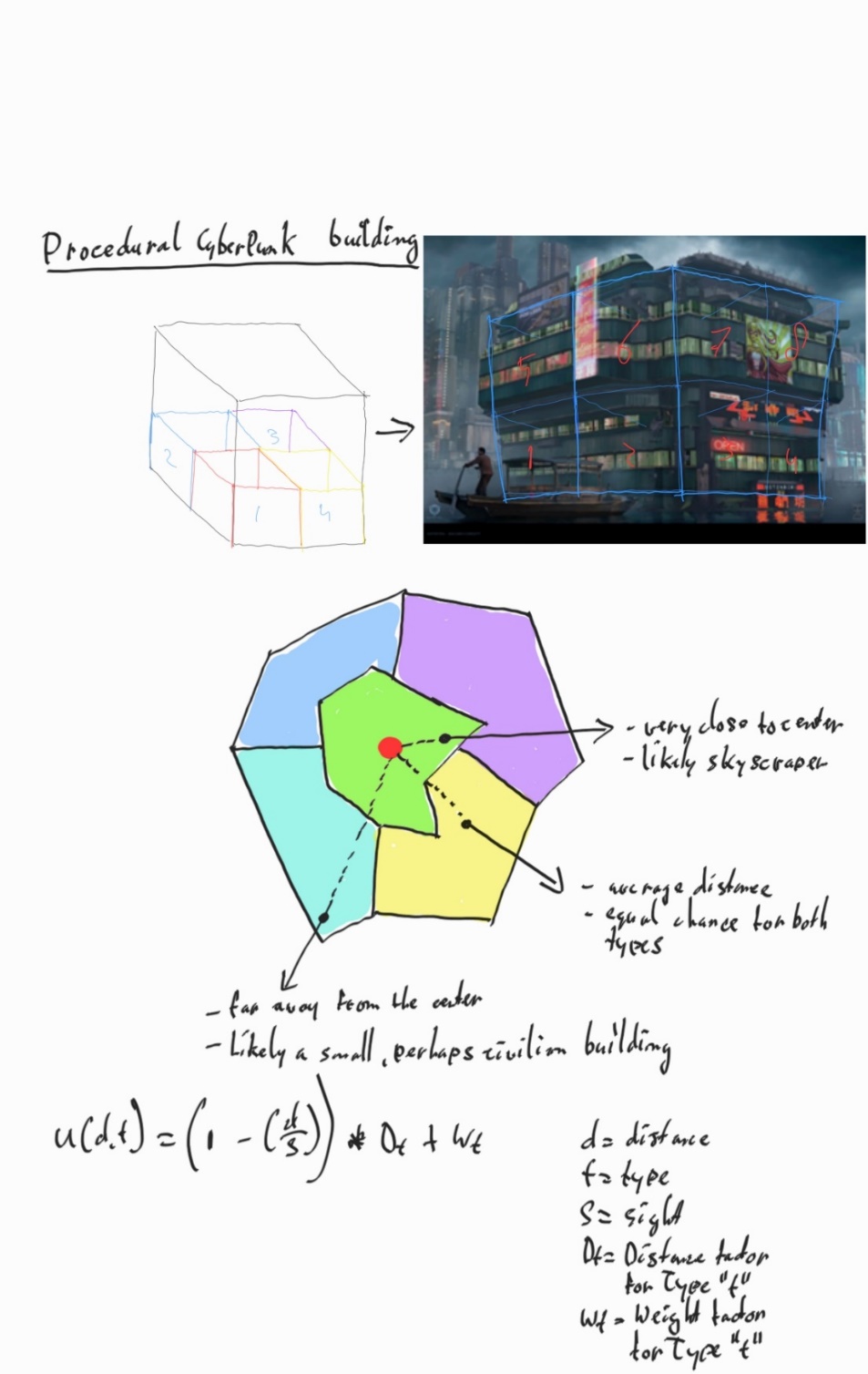


Figure 3 City layout

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”.

Utility function

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

distance = distance to the center (red dot)

currentBuildingType = type (e.g. 70% is corporate/skyscraper, 30% is civilian/housing)

maxDistanceToCenter = Sight (should this building even be considered for a certain type of building? Too close to the center to be a house, or too far away from the center to be a skyscraper?)

distanceFactor = Distance factor for Type *t*

typeFactor = Weight factor for Type *t*

randomValue = a random integer number between -5 & 5, so that there shouldn’t be a clear difference

Depending on what city block the building is in, I can assign certain values to the yet-to-be made buildings in that specific 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. I will be using numbers ranging from 1 to 100, to make sure the values can easily be worked with instead of incredibly small or incredibly large numbers.

**Example:**

* City block 3 has a 70%/30% skyscraper/house *desired* ratio, with an arbitrary value of 20 buildings total | 14 skyscrapers, 6 houses.
* The current ratio (how many buildings have already been generated for each type?) is 50%/20% | 10 skyscrapers, 4 houses.
* Type weight = desired ratio: skyscraper weight: 30 | house weight: 20
* Distance factor, dependent on type so link it to the type weight:
  + skyscraper: 0.2 \* 30 = 6
  + house: 0.5 \* 20 = 10
* The distance from the current building to the city center is an arbitrary 50 units (50m, let’s say a building has an average width/depth of 3m/3m).
* The max distance to the center is an arbitrary 75 units
* If distance > maxDistance, the function won’t execute and the other type will automatically be given priority, even if the ratio for that type has already been exceeded.
* I changed the division “distanceToCenter / maxDistanceToCenter” to a subtraction because it gave better results.
* Highest value has preference.

**Example:**

Blue = distance

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