ORTHO GRAPHIC

The next generation of ground representation:

Procedural orthophoto optimization for flight simulators, or other applications

Table of Contents

1. Introduction

Providing realistic imagery has always been a challenge in flight simulators. While modern approaches on the market avail of modern physics and weather models, one challenge remained: realistic and yet believable representation of the ground.

Modern versions of Microsoft's Flight Simulator software offer a relatively new feature called photogrammetry, coupled with ground imagery directly streamed from their own Bing service. While this approach seems feasible in theory, it has often been subject to criticism.

The product of Laminar Research, X-Plane, focuses primarily on the flight model and how a plane behaves while flying it. The developers always made it clear that the focus is not on a realistic, but only a somewhat believable appearance of the ground. The community has developed solutions that bring satellite imagery into the flight simulator.

While satellite imagery has heavily improved the realism of the ground representation in X-Plane, the imagery is often inconsistent: discoloration or seams, perspective distortion or in some cases, parts of an image are blurred out.

I have become aware of approaches many users attempted to rectify this situation. Some have simply recommended to switch satellite providers. Others have attempted to manually color-correct imagery by altering the histogram – however this may only solve localized imagery issues, it does not solve the challenge itself.

These challenges translate into both flight simulation platforms. Not a small number of flight simulator users have voiced their criticism when it comes to this approach, but up until today, no viable alternative existed that is able to solve the challenges presented.

Orthographic is a generative approach to solve the challenges, and is capable of providing realistic, yet consistent ground imagery.

Purpose of this document

This whitepaper is describes the technical aspects of Orthographic and its inner workings, and is mainly intended for potential customers, people with a technical background, and potentially software developers. Further, this document aims to provide transparency on the generic development process of the software in order to provide a holistic picture of the functionality of the software.

Details about the following points are discussed in this document:
→ current challenges with satellite imagery in environments where consistency is required
→ research to solve the presented challenges
→ implementation of the Orthographic software in order to resolve the challenges
→ Digital Twin applications

2. Current challenges with satellite imagery

Since mankind has launched devices into Earth's orbit, it was inevitable that at some point, these devices will be used to take photos of the ground. While the idea is simple, its implementation has its own challenges. Most notably Earth's size, and the space to store all of these photos. Further, image resolution is a factor, among many other aspects we do not need to address in this document.

2.1 Stitching and discoloration

Earth is big from our point of view, and photographing every square meter of solid ground, let alone in different zoom levels, is a massive undertaking. Therefore, some approach needed to be agreed upon, to provide imagery that is usable in services like applications, or even web services.

When the material taken by one or more satellites, sophisticated software is used to merge these images into seamless images, “stitched” together at just exactly the right borders.

The reasons why a “stitching” needs to happen, can be many – but most prominently this technique finds its application when a part of the material is completely unusable. Either due to technical error, or the ground was partly or completely covered by clouds, or by extension, the ground was too dark due to the shadow being cast by clouds.

If this happens, oftentimes the only viable solution to solve this issue is by either acquiring material from another provider and purchase use rights, or have the same satellite fly over the same location, and take more pictures. These pictures will naturally have a different look and feel as there might be different atmospheric conditions, different time of day, or the satellite is at a different altitude, to name a few factors. If however, the images taken are usable and can be applied to replace the otherwise faulty imagery, “stitching” is applied.

Let us have a look at some examples of this “stitching” occurring.

Source: Bing Maps

Here we see one such occurrence. While we can observe a coherence in shapes and the objects visible fit perfectly in the image, a clear line is visible, dividing the image in two distinct and completely different color contrasts.

Source: Bing Maps

Not only can we see a straight, vertical divider line of a stitch in this image, we can also see a more irregular horizontal line stretching off to the left. We therefore see three different color contrasts.

If you are interested to learn more about this technique, the Wikipedia article has further information on it: https://en.wikipedia.org/wiki/Image_stitching

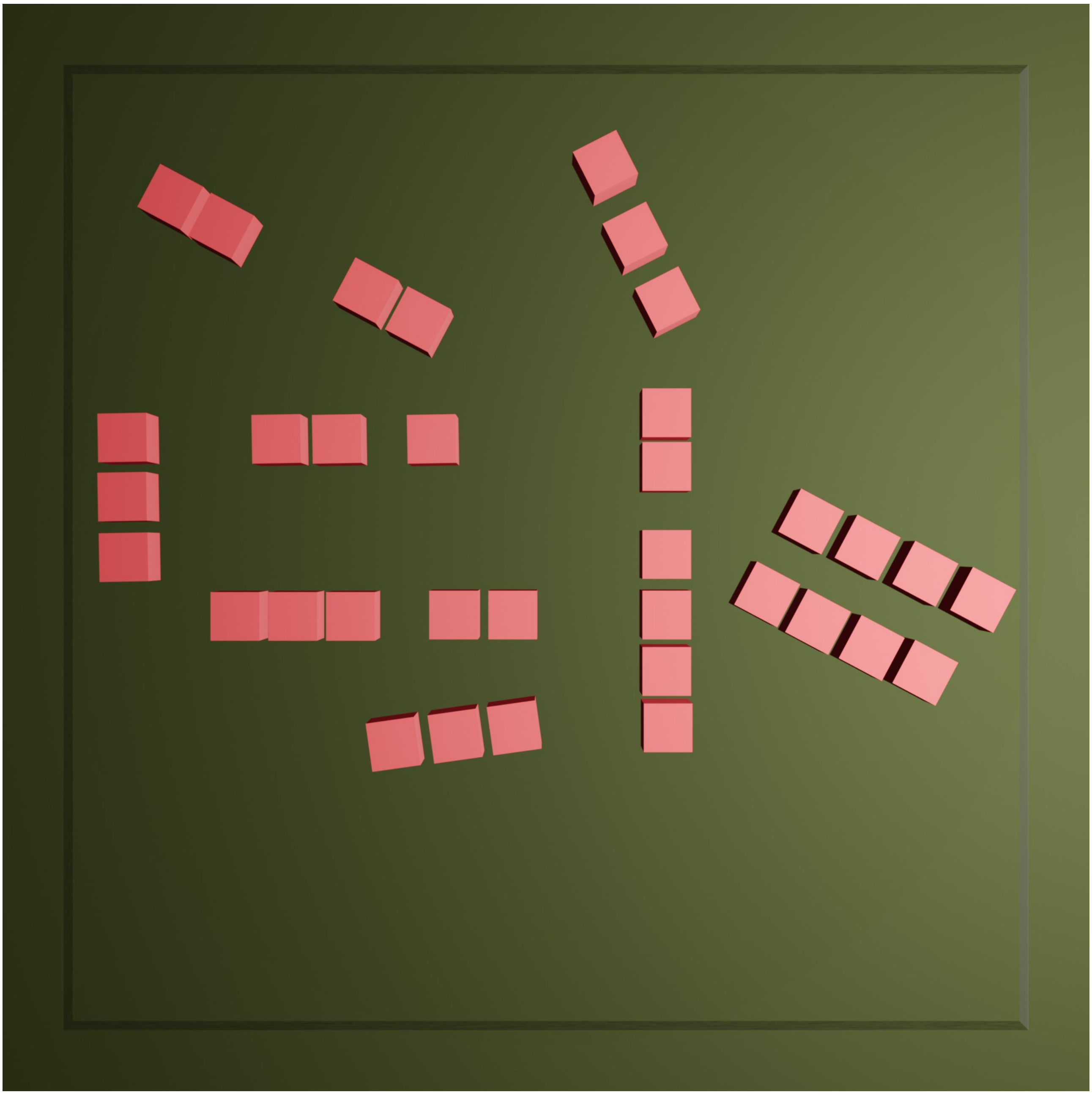
This phenomenon is often also described as discoloration – the effect that throws coherent appearance and coloring off the balance, due to different images being merged together into one.

2.2 Perspective distortion

When taking photos with any camera – no matter if being installed on a satellite, an optical camera or smartphone, it is taking an image of what the lens is picking up. That is a perspective, a view from a defined point in space. This is, naturally, subject to perspective effects. Meaning, the view "flees" toward a certain point – often the center.

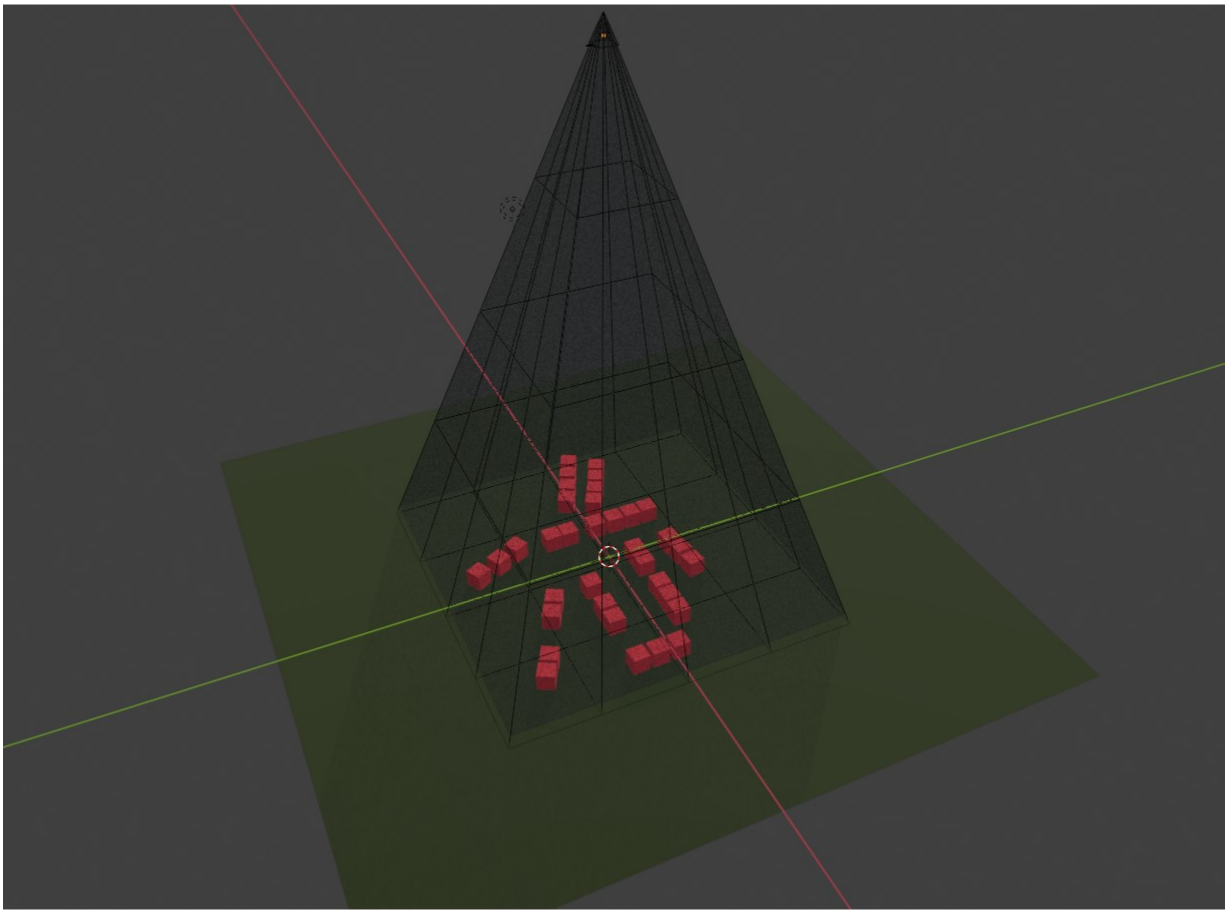
In other words, every camera has a frustum – a cone inside which it can take a picture of.

Observe a graphical example.



In this example, the red cubes represent houses on the ground. As showcased in this simple graphic, different sides of the houses are visible depending on the perspective frustum.

We can have a look at this from another point of view. Observe the next graphic.



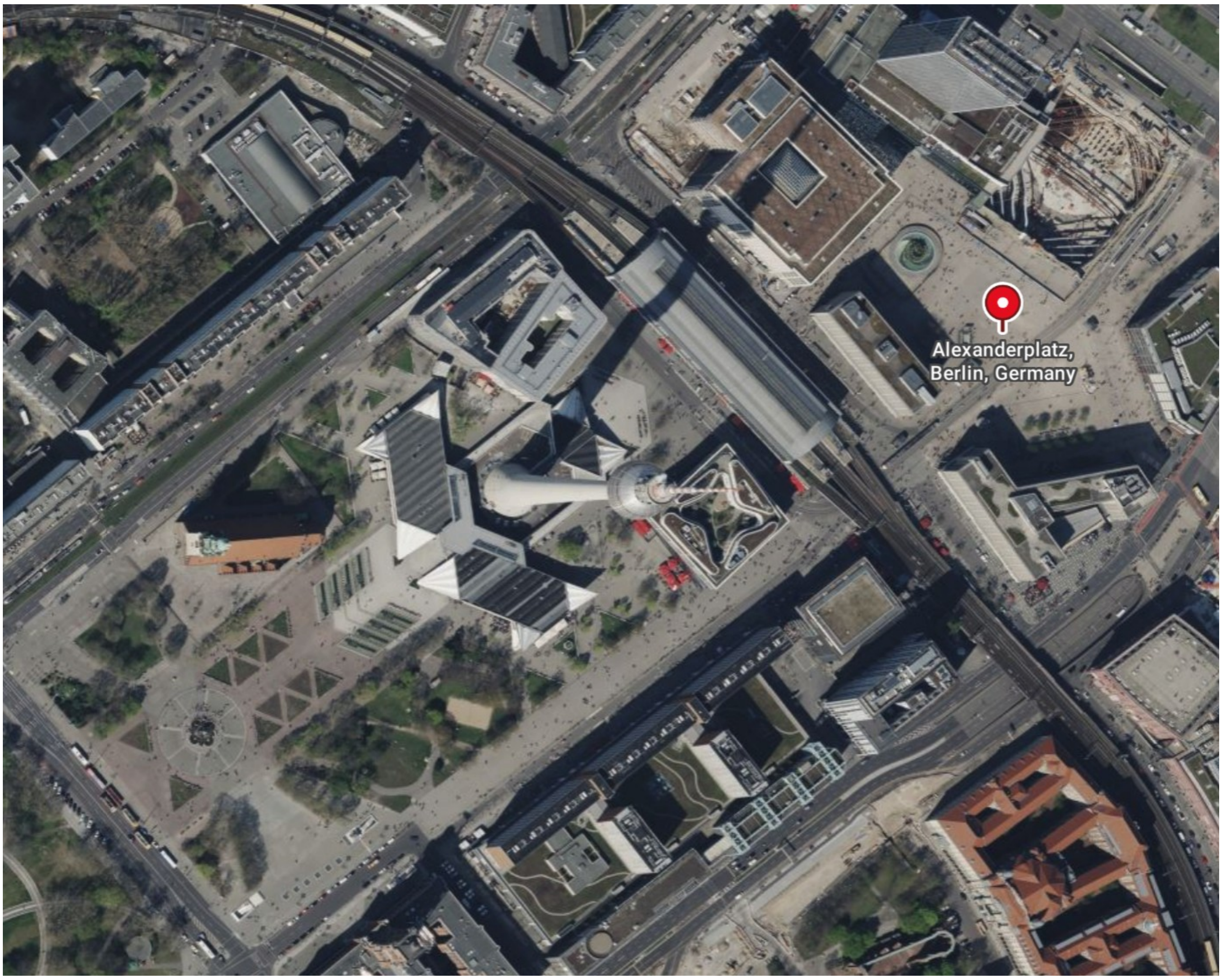
The black pyramid-like structure represents the cone – the frustum – of the camera, which would, if not obstructed by the ground, extend into infinity.

While techniques have been developed to counteract this effect in aerial imagery, it has not been eliminated entirely due to the limitations of optical cameras in the physical world.

Florian Tower Dortmund - Source: Bing Maps

Here we can observe on such example in an aerial photo.

Battling this distortion is proving to be particularly difficult, if not impossible, with larger objects such as the one visible in this photo. If you observe closely, an effect similar to the graphic shown before can be seen on some of the larger buildings.



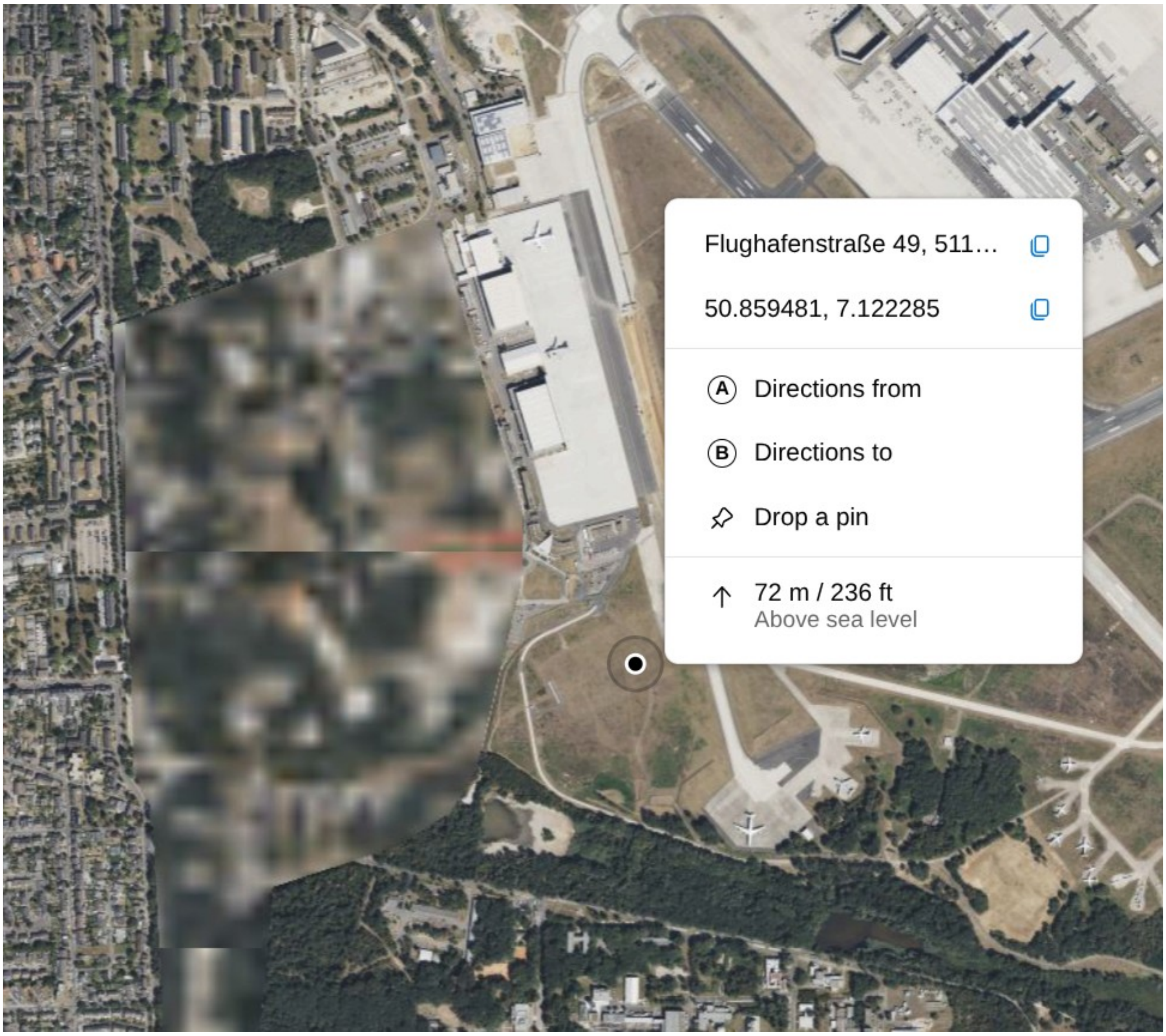
Source: Bing Maps

The same is happening with the iconic tower on the Alexanderplatz in Berlin, Germany.

2.3 Pixelation / Blur-out

The third challenge that arises with consistency in satellite imagery is, of course, perfectly understandable: privacy, and in some cases, secrecy. This is most often the case with military bases. Satellites may have snapped a picture of something that isn't meant for the public – and naturally, web services such as Bing Maps and Google Maps need to comply with the law and cooperate with the military – and that means that the areas where there are military bases will be made incomprehensible. In all cases I have encountered, these bases have a blur on them or are pixelated.

Let's have a look at a few examples as well.



Source: Bing Maps

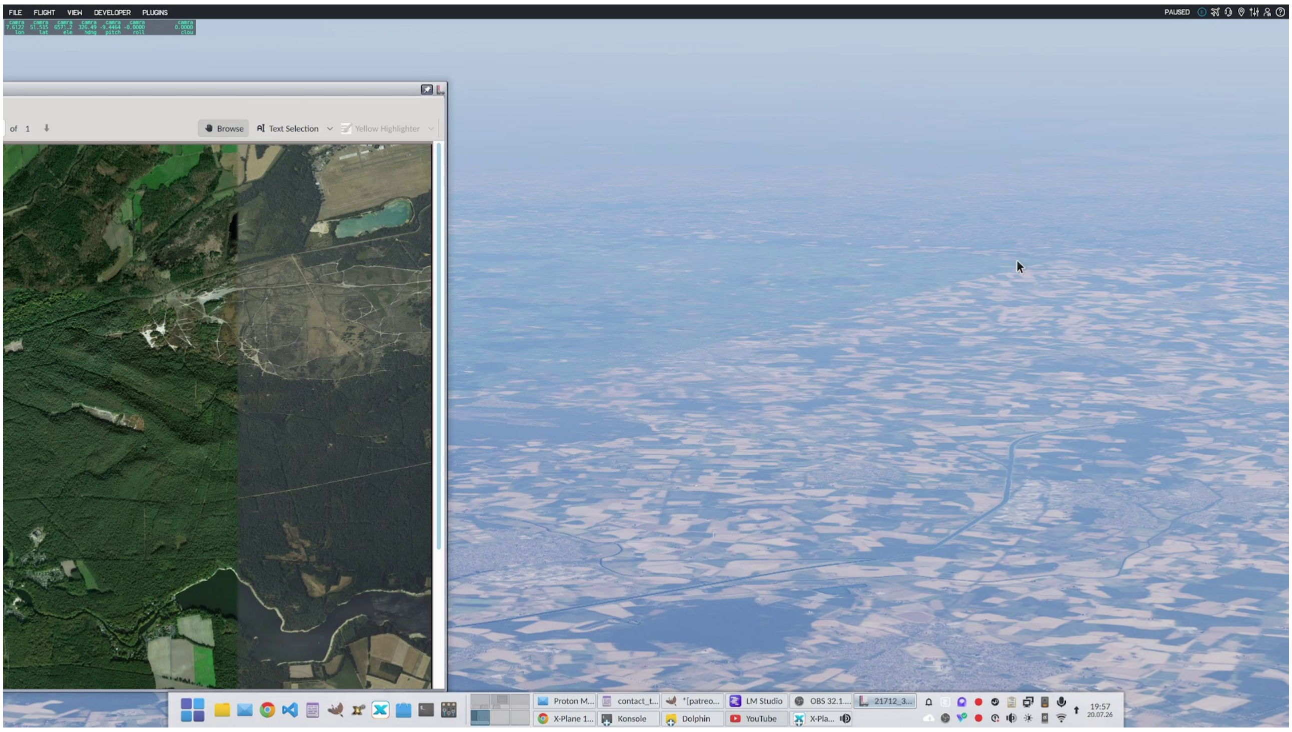
This is near the Cologne-Bonn airport, which has a military base nearby.

Source: Bing Maps

Another example of privacy on an aerial maps web service.

2.4 Challenges transport into flight simulation platforms

When generating scenery with one of the well-known software solutions, particularly for the X-Plane flight simulator, these challenges will transport automatically into the images being requested by the tool that downloads the orthophoto tiles. The tiles are being delivered with the “faulty” imagery, and is then visible in the scenery when flying over it.



Source: Orthographic promotional video – (c) MarStrMind

In the simulator, there is a clear line visible, dividing the area into a zone with a more greenish tint, while most of the remaining area is seemingly not affected by it. Here we can observe the aforementioned discoloration – the effect caused by stitching – which translated into the flight simulator.

This is not a truly realistic representation of the ground.

Let's have a look at a new, viable approach, and possibly a next-generation approach to eliminate all challenges presented with aerial photography when used in environments where consistency is required – such as flight simulators.

Although many aspects of the flight can be handled by the autopilot including the approach on ILS, pilots also often rely on visual cues in the area as well. Such imagery is not helpful – no matter how advanced the simulator may be.

3. A viable, new approach

It has come to my attention that I am not the only person who dislikes inconsistencies of this kind in the flight simulator imagery. While some have tried manual ways, for example to color-correct affected images by hand, none of the approaches has been able to yield a truly reliable approach to correct the imagery.

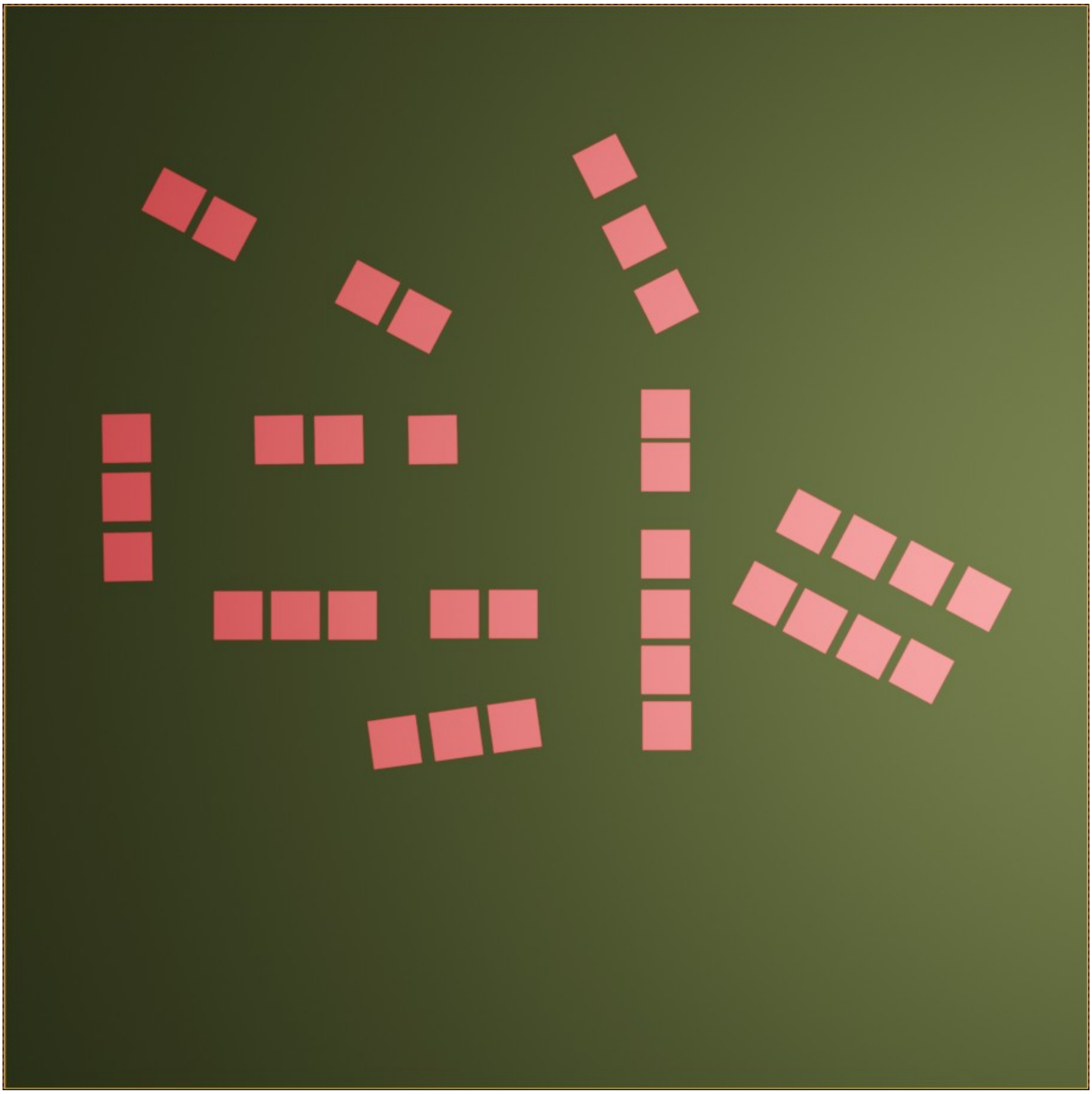
Therefore, a different approach had to be engineered. An approach that would – ideally – be able to provide truly consistent imagery, free of artifacts and distortions, while retaining the same optics as aerial photos taken by satellites or planes at high altitudes.

3.1 Perspective vs. orthographic projection

As discussed earlier, all photos taken with a lens in the real world have perspective projection. Of course, the camera is subject to three-dimensional perspective of the physical world.

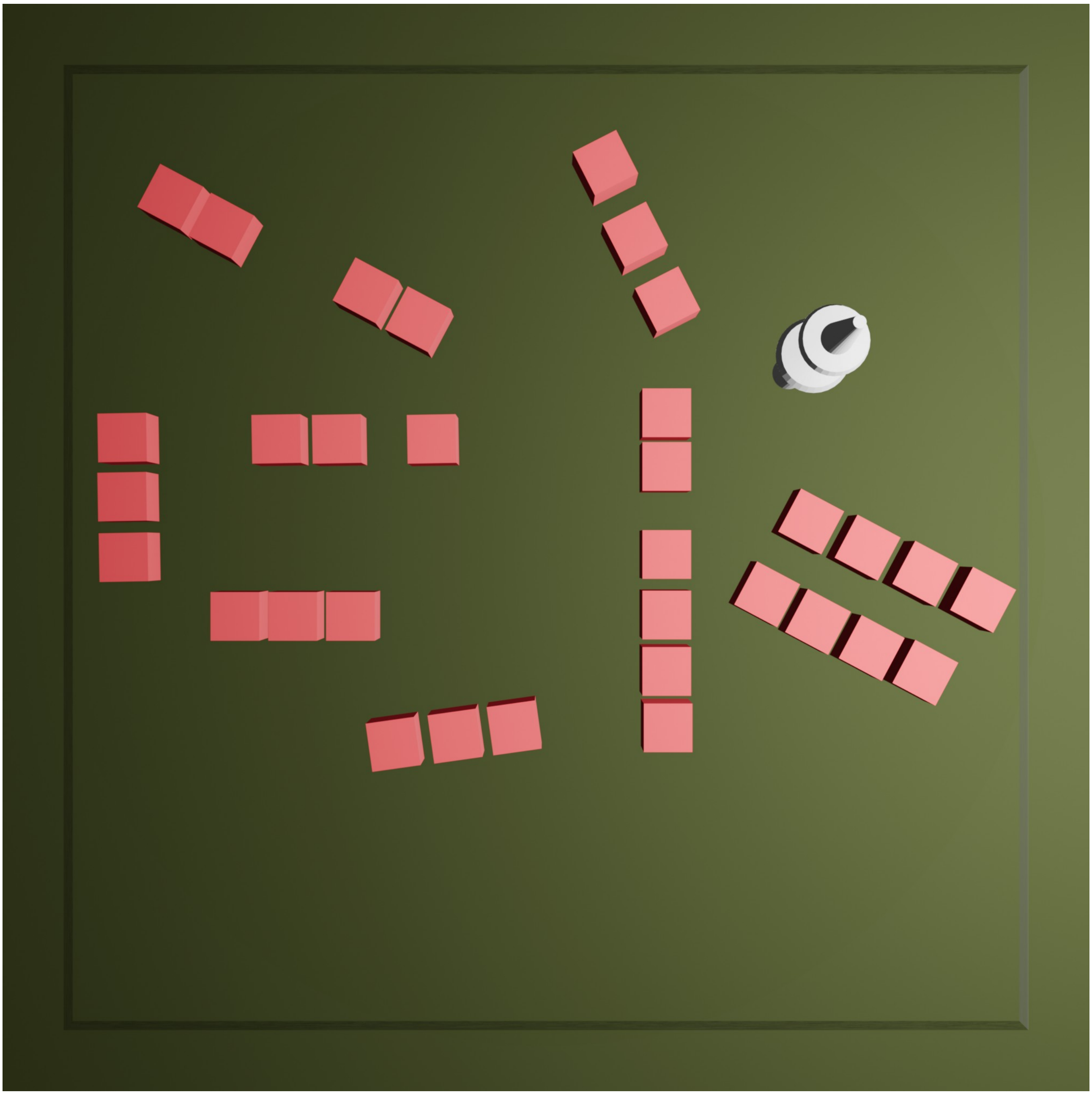
In technical applications, for example when drawing up blueprints of a building or structure, orthographic projection is being applied. It is a projection in which the “camera” does not have a pyramid-like cone extending from its location into infinity – it is rather a rectangular tube extending into infinity.

This dramatically changes the perspective, and effectively removes any perspective distortion that occurs with a normal frustum.

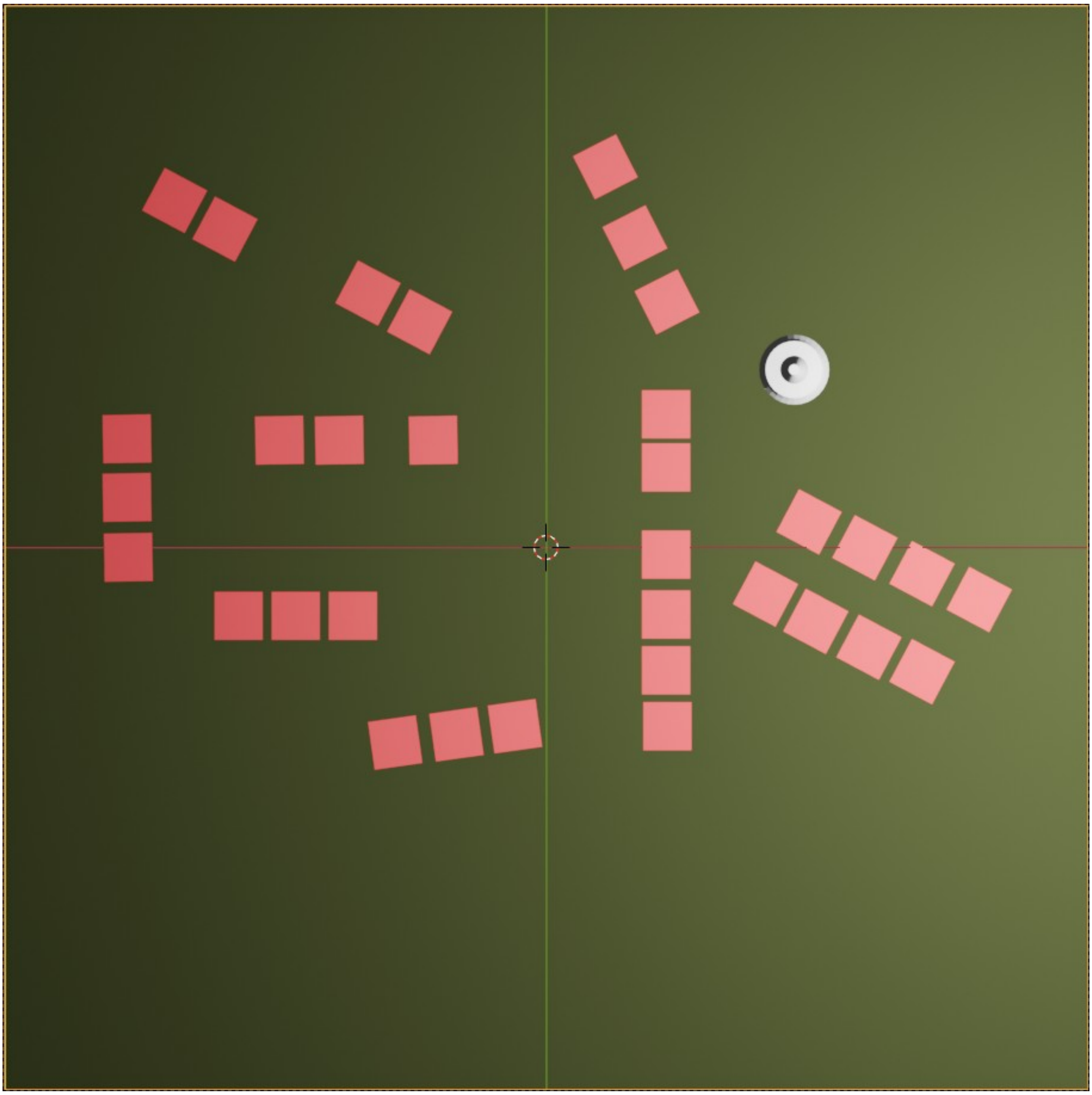


Similar to the image we have seen earlier, where the red squares still represent houses, this image was produced using the orthographic projection mode. As you can see, we can only see the symbolic roofs of the buildings and houses, we no longer see their sides – at any point in this graphic.

That is due to the rectangular frustum tube instead of the pyramid one.



Circling back to the example with the towers earlier, here is a simple graphic containing a tower in the residential area, with normal perspective. We can also observe a similar distortion as we have seen in the actual photograph. This would translate into a fixed imagery into the scenery on the ground.



If we, however, switch to orthographic projection, we achieve a much more desirable representation of objects on the ground. Here, the tower is also freed of distortion artifacts – much more suited for a representation on the ground.

3.2 Parts of the solution

Now that we know of the differences between these two options of projection, I believe it is clear that the core pillar upon which this approach rests is orthographic projection – hence the name of my solution.

Another, also very integral part of the solution, is to understand that in order to solve all challenges presented, something needs to be removed from the equation. In this case, the “something” that has been removed, is satellite imagery itself.

In order to work on a proper solution, it is most desirable to remove faulty factors from the approach. As we have seen, satellite imagery is prone to be undesirable. Therefore, I cannot work with it.

4. The next generation of ground representation

In this chapter, I will discuss the technical aspects of the solution, and what has been implemented in order to achieve the publicly available imagery and scenery. Development and challenges are being approached, and how they were solved.

The development on this approach has been going on for three years as of the time of this writing.

4.1 The aim

It is important to understand that there is a clear difference between real, and realistic. The aim of Orthographic is to offer and provide, realistic and believable ground imagery, or ground textures, for flight simulators – with X-Plane being targeted primarily. The result shall be believable, and realistic in appearance.

The aim is not to render images that look like photographs being taken by a satellite, but free of the challenges. In other words, Orthographic does not want to provide 100% real images.

4.2 Realistic representation without a satellite

To achieve a ground representation of any kind without relying on real-world imagery, there is only one other way: data. This data is readily available by a number of services, with the most popular and most accessible one being data from OpenStreetMap – or OSM for short.

Many different datasets exist, and there are services available that can provide data of areas you are interested in – from small towns to entire continents. Or, if you have the space, the entire world.

For the entirety of development, OSM data has always been the cornerstone to the results I have publicly presented in the promotional video and on my Patreon page.

4.3 Development bullet points

Initially, I have been performing research on how such a solution can be implemented. Among the candidates were the programming languages C++, C# and Python. Testing with different libraries, and portability has yielded a result in favor of C++ and Python – but ultimately I decided to develop this project in Python, as the image libraries and other components were simpler to access and to work with.

→ Developed in Python
→ Currently working with Python 3.14
→ Virtual environments used for many components
→ Pillow and numpy used in almost all components
→ Multi-Threading used for parallel rendering of tiles
→ Largest software project I have worked on

4.4 Data in use

The solution is implemented and realized using up-to-date datasets from OpenStreetMap. To be more precise, the PBF datasets have been acquired from GeoFabrik.

At this time I am concentrating on Europe and Germany – therefore, these datasets have been acquired initially.

The PBF files are being processed by a Python component I have developed, which then proceeds to convert the information found within into the format that is required by my implementation.

4.5 Material in use

Once the feasibility of this undertaking was determined to be possible, a pool of material to be used needed to be built up in order to link an OSM tag with a graphic image.

For this step, no generative AI was used. Experiments using generative AI were conducted, but it was found that the results are not usable for the project.

The photo-realistic material currently in use has been collected over weeks by me personally, and then stored in a format that will later result in textures for particular areas. In other words, I have collected and hand-crafted textures for forests, farmlands, buildings, meadows, etc. They have then be stored into different folders, and the code was adjusted to dynamically access the different resources as needed.

Currently, 15 different textures for a large number of tags have been compiled.

4.6 Reconstructing the ground

The process to render the images, which ultimately make up the scenery, can basically be broken down into three generic steps. Each step requires considerable amounts of time, even with a machine that would be considered high-end by today's standards.

At the time of this writing, rendering out scenery for a complete latitude and longitude tile, takes roughly six hours.

Generally, the steps are as follows:

Extract OSM data
During the development, a list of tags has been compiled, which covers 90-95% of all occurrences of tags worldwide. PBF files contain far more than that, therefore I am filtering for this particular data Orthographic wants, and then store it into a format, which allows me to do spatial queries very fast. Depending on the amount of data within a PBF file, this step can take some time.

At this stage, Ways and Relations are being processed properly, but also, indepedently.

Render subtiles
Once all the data is in place, the rendering process can commence. As multiple layers need to be rendered, this step uses Multi-Threading to render each layer independently. During this phase, safeguards are in place to prevent race conditions, which would otherwise end the process of rendering prematurely.

When the layers for a subtile are finished rendering, they are handed over to the next step in the pipeline. Its task is then to combine the single layers into a single image, in the correct order.

During the final step, the renderer makes sure that every pixel in the image is of a solid color. Some areas have no data, so I am deploying a mechanism which fills in the blanks with believable imagery.

Generate scenery
Once all subtiles are complete, the final step is to use these images to produce a mesh using an Digital Elevation Model (DEM), and apply those properly on the mesh. In addition, the mesh is equipped with the X-Plane 12 water technology, to complete the scenery package.

4.7 Notable challenges in development

Consistent imagery
Once I had established a process to render polygons from OSM data, I needed to figure out how to make polygons have the same resource – as they often extend to other subtiles.

Initially, I experimented with a path-mechanism, similar to what's being used in games. This only worked partially, and ended up being dismissed.

Once I learned that all Ways have a unique ID in OSM data, I was able to assign a resource to a Way ID – and thus solving the challenge. This mechanism is in place to this day.

No clear cut borders for certain tags
This is primarily for forests, grass areas, farmlands, meadows, and a few other similar tags. I experimented around for about six months to find a solution. Ultimately, a mechanism was applied that applies a random noise around the borders of the polygons, and then use the noise-adjusted mask for rendering.

Prior to that I have experimented with other things, such as ovals on the borders, which did not really work too well.

Relation members
While rendering out Ways and their polygons is a relatively simple task, Relations is a group of Ways – or polygons – with different roles. Initially, I read the data incorrectly, which either caused relations not to show up at all, or them being rendered with undesirable results. Only recently I have posted an example of this on my Patreon development blog.

This warranted a deeper research into the code, and it was determined that there was indeed a flaw in the mechanism that processes members of a relation. Fixing this required a few days, but the result is rewarding.

Relations render correctly now.

Render speed
While the initial implementation worked, the render speed even for a subtile left much to desire. During the course of the development I have implemented improvements at many points in the render pipeline, with one change being the most notable one. Initially, all way polygons were rendered separately, including their texture backgrounds. This, of course, took a lot of time. In the current implementation, all polygons required are collected into a single pool of polygons, before the mask is being rendered. This yielded the biggest speed boost yet.

Shadows for realism
After looking at satellite images again for reference, it became evident that some elements cast a shadow into their surroundings. Over the course of development, a number of approaches have been tried to achieve this effect. The current mechanism applies shadows only to some tags: mostly forests and buildings. The shadow casting emulates a sun shining from the upper left onto the area. Shadows are randomized or a number of tags, and forests emulate different lengths of shadows, to simulate different heights of trees. At this time, this is the best approach for this challenge so far.

Consistency across latitude and longitude tiles
No resource ever ends exactly at the border of a latitude and longitude. This ties in with the first point, but I needed to make sure that the same resource is being used across all tiles. This resulted in a re-adjustment of the database format, allowing for truly global consistency, without needing to worry about where a resource is. This adjustment also affected relations, meaning that those also render correctly across all latitude and longitude borders.

5. Digital Twin applications

While the current implementation produces scenery for the X-Plane flight simulator, the technology can easily be adapted to provide synthesized ground imagery for Digital Twin applications. Feasible might be an additional mode in mapping services, like Google or Bing.

Choices might then include: Map, Satellite, Orthographic – where the third option would render a map using the synthesized ground imagery using this technology.

Other options may include ground imagery of areas where changes are planned, for example how an area looks now and how it might look with proposed changes. Furthermore this technology might be applicable to city or landscape planning, and can aid in providing a realistic outlook on the area, without the requirement for a satellite to take photos of an area.

As water and road rendering is now implemented, the technology can also be used to render out the ground imagery, and use the results in topological surveys with ground mesh models in 3D applications like Blender. The resulting imagery can also be applied to projects in QGIS.

This is only a selection of possible use cases for this technology. If you are interested in developing an implementation for a Digital Twin application, you can contact me directly at <marcus@marstr.online> .