# Assignment 1 - Poor-Man's Google Streetview
# 25 (+20) points

Deadline: 2022-03-23 23:59

Figure 1: Google Streetview Panorama

## 1 Introduction

In this exercise you should build a basic scene that visualizes a spherical image with correct UV mapping. In Google StreetView, a set of such spheres is placed one after the other along a path, which enables the user to rotate and view the environment more closely.

Such a sphere can also be used as a plain background in a VR setup. This replicates something that would be rendered in the distance to generate the immersion of, e.g., being in a city environment (city skyline, horizon). For this, a huge sphere around the camera at a very far distance is rendered[1].

## 2 Tasks

- Create a Unity Project, download image package from TeachCenter (or download your own panoramas from Google StreetView using a Down-

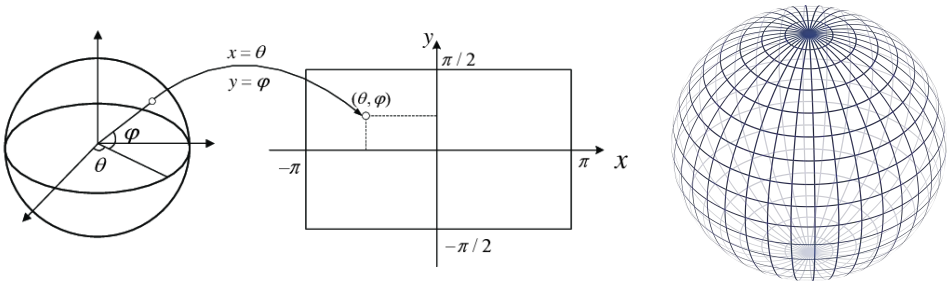
[1] There is a link to Assignment 3 which will be given later on.

loader[2]

- Import contained spherical panoramas as images/textures
- Create spheres at equal distances along a linear path
- Warp individual panoramas with a shader to the individual spheres
- Implement basic navigation to rotate within panorama with the mouse and to switch between panoramas with keystrokes
- Include minimal aid to help the user in navigation (e.g. W and S moves forward/backward between panoramas)
- Optional: implement smooth motion between panoramas (similar to Streetview)
- Optional: implement blending between panoramas upon switching (similar to Streetview)
- Optional: implement camera controls to

# 3 Mapping/Warping (20 points)

In order to warp the image correctly, implement a simple shader, based on the Unity shader example[3], which mirrors the X-Coordinate of the image texture and hence match the inside out view of the camera. Since the equirectangular projected texture's UV coordinates have to be properly mapped to the sphere to match the higher pixel density on the poles of the omnidirectional video footage, you should implement a shader which calculates the proper UV coordinates per fragment. The shader is attached to the sphere via a Unity material which holds the shader.




(a) illustation for UV mapping, source for image paper by [1] (b) pixel distribution sphere

Figure 2: UV mapping

[2] https://svd360.istreetview.com/

[3] https://docs.unity3d.com/Manual/SL-VertexFragmentShaderExamples.html

Figure 2a illustrates the process of mapping a pixel of the sphere into the UV coordinate space of the texture. The calculation of the UV coordinates is represented by equation 1.

$$u = 0.5 + \frac{arctan2(dx,dz)}{2\pi}$$
$$v = 0.5 + \frac{\arcsin(dy)}{\pi} \tag{1}$$

To implement Equation 1 the direction vectors for each vertex to the camera have been calculated and passed to the fragment shader which then calculates the proper UV coordinates for each fragment. Since the pixel density increases near the poles the effect of the proper UV mapping is not as obvious during normal usage compared to the shader which only mirrors the X-Coordinates. Yet, the effect gets more pronounced when looking at the poles where the incorrect mapped shader struggles to find the proper texture mapping. Taking care of this

(a) incorrect UV mapping (b) proper UV mapping

Figure 3: Shader Comparison

## 3.1 Basic Navigation + Help (5 points)

In order to enable a Streetview-like behaviour, implement mouse-based rotation within a single panorama (respective hints to be found in the Unity manual or on Youtube). Similarly, use keystrokes to switch between the individual panoramas.

## 3.2 Bonus Points (20 points)

**Smooth Motion (5)** Implement a smooth motion pattern between panoramas, rather than having a discrete jump.

**Blending (10)** Think about a way to blend between panoramas during motion, rather than having discrete jumps upon switching between panoramas (fade-effects?) and implement.

**Camera Controls (5)** Think about a way to change your camera view of the panoramic environment (between 4:3 and 16:9, focal distance of camera, etc.) and implement.

### 3.3 Implementation Thoughts for Those that like nice code

- If you have the basic concept around a single panorama ready, can you create a path generator based on $N$ images in a folder and use Prefabs?
- If you use your own panoramas and you know where they are, how about shaping the path according to the GPS position (around a curve, loading the data from EXIV or a txt file)?
- Think about other ways to extend your setup (with a textured ground plane from Google maps?).

## 4 Submission

You should submit a **30s video of you navigating within your Unity Scene between the individual panoramas** (.mp4). The video needs to contain all implemented tasks and bonus tasks. You need to upload two items to the TeachCenter: your Unity package and the video, compressed together as a .zip file. Include your name in the filename.

## References

[1] Deyang Liu, Ping An, Ran Ma, Wenfa Zhan, and Liefu Ai. Scalable omnidirectional video coding for real-time virtual reality applications. *IEEE Access*, PP:1–1, 10 2018.