

An aerial photograph of a city at sunset. In the foreground, a large, modern white building with many windows is visible. The city extends into the background under a warm, orange-hued sky. A tall industrial chimney is visible on the right side of the horizon.

FH

University of
Applied Sciences

TECHNIKUM

WIEN

Projektbeschreibungen

Outlook

- This slides describe 4 possible projects. **Students need to choose 2.** For each chosen topic, you can get 25 points at most.
- This means in the project you can earn 50 points.
- The other 50 points are quizzes and small exercises in moodle.
- The options are
 - Ray Casting (using pixel shaders)
 - Hardware Ray Tracing
 - Point Cloud Rendering
 - Neural Rendering



TECHNIKUM

WIEN

Topic 1, Ray Casting

Ray Casting in Shader Toy

- In dieser Aufgabe geht es um die Erarbeitung eines komplexeren Shaders, welcher Ray Casting einer einfachen Szene als PixelShader implementiert.
- Es wird empfohlen diese Aufgabe in ShaderToy zu realisieren.
- In sowohl Präsenz als auch im Eigenstudium wurde ShaderToy bereits behandelt. Hier gibt es ein [Tutorial](#) dazu (insbesondere [hier](#))
- Anforderung an das Projekt
 - Eine Szene mit mindestens einer Bodenebene, einer Box und einer Kugel
 - Beleuchtung nach diffusem Beleuchtungsmodell
 - Berechnung von Schatten mittels Ray Cast zur Lichtquelle

Abgabe und Diskussion

- **Struktur des Projektes**
- **Unterschied Ray Casting / Ray Tracing**
- **Beleuchtungsmodell (wie funktioniert die Berechnung der Farbe)**
- **Lessons learned**



University of
Applied Sciences

TECHNIKUM

WIEN

Topic 2: Hardware Raytracing

Hardware Raytracing

- Schauen Sie sich das DX12 Raytracing [Tutorial](#) von Nvidia an.
- Implementieren Sie zunächst das Tutorial (D3D12HelloTriangle).
- Dieses beinhaltet Hit&Miss Shader, Aufbau der acceleration structure und traversal Methode
- Erweitern Sie die Szene so, dass ein Würfel auf einer Ebene (z.B. großes Quad) steht
- Der Hit Shader muss auf das Objekt angepasst werden, sodass die Ebene und das Objekt verschieden schattiert werden.
- Weiters muss der Hit Shader für die Ebene angepasst werden, um einen Schlagschatten darzustellen (Schattenfühler → Sekundärstrahl zur Lichtquelle). Hier ein [Beispiel Tutorial](#).

Abgabe und Diskussion

- Falls Sie lieber Vulkan oder eine andere API (welche Hardware Raytracing erlaubt) verwenden ist dies auch möglich.
- Inkludieren Sie folgende Diskussionspunkte in der Abgabe:
 - Verwendete Technologie, kurze Beschreibung des Aufbaus des Projekts
 - Warum Ray Tracing?
 - Warum Hardware Ray Tracing und Abgrenzung zu Ray Casting.
 - Was sind die Erfahrungen mit der Verwendeten Technologie
 - Lessons learned
 - README über die Benutzung der Anwendung.



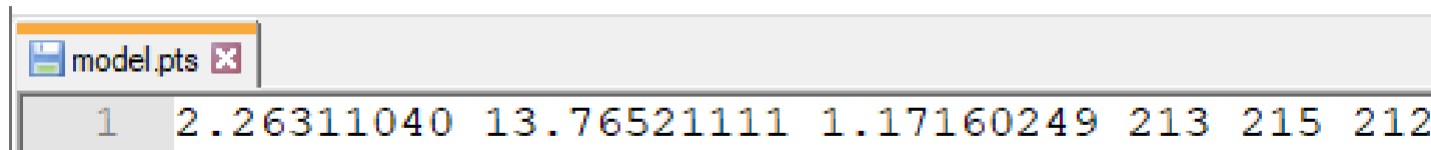
TECHNIKUM

WIEN

Topic 3: Point Cloud Rendering

Point Cloud Rendering, Part I Parsing

- For this task you can use your favorite environment/engine/codebase
- The task is to visualize the accompanying file „model.pts“. We also provide a verison as .ply file. You can also use for example CloudCompare to transform the format if needed (e.g. when you are using an engine which cannot load pts files).
- The pts file is very easy to read. Each line describes a 3-dimensional point and a RGB color, for example the first line:



The screenshot shows a text editor window with a single line of text. The text is: 1 2.26311040 13.76521111 1.17160249 213 215 212. The first column contains the number 1, followed by three columns of floating-point numbers, and finally three columns of integers representing RGB values.

- The first point thus is (2.26311040, 13.76521111, 1.17160249), the color is (213,215,212).
- As a hint when placing the camera, the axis aligned bounding box of the model is:
 - Min = (-4.563, 10.620, 0.501), Max =(6.436, 20.246, 4.140)

Point Cloud Rendering, Part II Parsing

- In this task it suffices to use simple sprites for splatting the points to the screen
- It is not necessary to generate circles etc., rectangular splats suffice but of course you can improve on the visualization as you wish.
- „Point size“ should be changeable in the application (e.g. by command line arguments, buttons or keyboard shortcuts)
 - Choose an implementation technique which fits your GraphicsAPI/Engine
 - e.g. point-sprites**, geometry shader, or instancing
 - **) not available in d3d11 and newer
- Use the color values as color for the points

Point Cloud Rendering, Part III Discussion & Report

- **Include a discussion of the following topics in the submission**
 - The discussion is part of the grading.
 - Also include an image of the visualization
 - Include a README file on how to use the project.
- **Explain the structure and the goals of the project.**
- **Explore the data-set. What is the weakness of the visualization?**
- **What challenges do you face when looking at this data?**
- **How could the visualization be improved?**



University of
Applied Sciences

TECHNIKUM

WIEN

Topic 4: Neural Rendering

View synthesis with Neural Radiance Fields

- The goal of this task is to experiment with view synthesis approaches, run them with your data, gain understanding and document your findings in the submission.
- Depending on your GPU and the available time budget, run one of the following **Neural Rendering techniques**:
 - Instant NGP (fast, NVIDIA)
 - Clone, build or just download the exe here: <https://github.com/NVlabs/instant-ngp>
 - Run, by following the instructions: https://github.com/NVlabs/instant-ngp/blob/master/docs/nerf_dataset_tips.md
 - Apples tiny-nerf implementation for tensorflow (fast, Apple)
 - Follow these instructions: https://developer.apple.com/documentation/metal/metal_sample_code_library/customizing_a_tensorflow_operation
 - Similar to Instant NGP
 - Be warned, we didn't test this!
 - The Original NeRF (slow, NVIDIA)
 - Tensorflow: <https://github.com/bmild/nerf>
 - Pytorch: <https://github.com/yenchenlin/nerf-pytorch>
 - MultiNeRF (slow, NVIDIA)
 - 3 Flavors of nerf in one repository
 - MipNeRF 360: NeRFs without aliasing
 - RawNeRF: raw-sensor data + NeRF = HDR reconstructions
 - RefNeRF: Better reflections
 - Repo: <https://github.com/google-research/multinerf>
 - NeRD (slow, NVIDIA)
 - Attempt to estimate BRDF surface parameters and lighting conditions
 - Repo: <https://github.com/cgtuebingen/NeRD-Neural-Reflectance-Decomposition>
 - Adop (slow, NVIDIA)
 - Not a NeRF at all, but a neural network filling the gaps in a fancy pointcloud.
 - Repo: <https://github.com/darglein/ADOP>
 - Whatever you find!
 - There is so much out there. If you can demonstrate and discuss it, you pass the assignment.

Submission / Discussion

- **Focus on the strength and weaknesses of the individual algorithms**
- What does work?
 - What is the advantage of this method over others?
- What does not?
 - Can you reliably break it? Without simply feeding it poor data/poses!
- You can collaborate by sharing images/sequences and comparing results of different methods.
 - It is OK to show results of others to provide a reference.
 - Take care to properly tag images that are not your own.
 - This still is not a group assignment.