Real-Time Rendering 4 Seasons Docs

General

Group: 2

Members:

- Manuel Eiweck 01633012
- Peter Kain 12119833

Github Link: https://github.com/Emanum/rtr24-4seasons

Story/Scene description

We want to create a Rendering demo which showcases one scene (or very similar looking scenes) in all 4 seasons. The base theme of the scene should be in a Japanese/Chinese style. E.g temples, lanterns, Trees, nature.

The transition between seasons will be hidden by either the camera movement itself (blocking scene with one object) or just a fade in black animation.

For music a general background music is planned that fits the scene.

Effects

- Depth of Field
- Reflections
- SSAO

Tech stack

Programming Languages:

- C++
- Vulkan
- Auto-VK-Toolkit - https://github.com/cg-tuwien/Auto-Vk-Toolkit
- Audio Engine (SoLoud, ...)

Controls

- F …. Start/Stop following the saved camera path. It will loop endlessly.
- R … Start/Stop the recording of a new camera path. The old one will be overwritten
- When Quake Camera is enabled (via mouse click) Use WASD + mouse to move around

Tested on NVIDIA 3070

Project Status

Basic Rendering Engine

We setup the project with Auto-VK-Toolkit based on their scene loader template. This allows us to load a model and also provide us with a basic render loop and support for textures and a free flying camera.

Modelling of the Scene

Then we worked on getting to build a scene in Blender and find a way to export it in a way that allows us to import it via the Auto VK Toolkit. We found the following setup works for us:

- Export as GLTF this forces Blender to export the Textures as a separate file.
- Furthermore we export the scene as an OBJ with the Setting Path Mode 'copy'. This way the .mtl file has only the relative path to the texture file.
- In our application itself we load the obj file. Internally assimp also automatically loads the textures and prepares our GPU Buffers

Automated Camera Path

We implemented a mechanism to record and play a camera path. The Camera Path is stored in a plain textfile alongside the assets. We store the position of the camera as well as a direction vector.
The camera path is followed along then and interpolated via the saved control points via a bezier curve.

Next Steps

For the next steps we want to continue to build a more complex scene in Blender. As well as implementing our effects which we did not started yet.

Planned Effects

Depth of Field – Manuel Eiweck

Real Time Rendering 4th Edition - Chapter 12.4

https://www.youtube.com/watch?v=v9x_50czf-4

including published code pieces
https://github.com/GarrettGunnell/AcerolaFX/blob/main/Shaders/AcerolaFX_BokehBlur.fx

(Screen Space-) Ambient Occlusion, Reflections – Peter Kain

Real Time Rendering 4th Edition - Chapter 11.3

As a starting point to implement the effects we got a copy of the Real Time Rendering 4th Edition.

Tomas Akenine-Mller, Eric Haines, and Naty Hoffman. 2018. Real-Time Rendering, Fourth Edition (4th. ed.). A. K. Peters, Ltd., USA.