TDT4230

Final project:

Introduction

This project was about investigating a more advanced or complex visualization method in detail. Some of the methods focused on for this project are: Fresnel refraction, cubemap texturing and dynamic texturing, however other minor features have been added as well. All of these methods were used to visualize an icicle scene, inspired by this image [2]. The refraction was used on the icicle to display the background and the light into the camera. The cubemap gives the scene a better looking environment, and the dynamic texturing was used to give more life to the scene by having the icicles be melting with water running down the side. Any other additions were made to add further details to the scene.

Requirements

The final report should document the process that went into implementing your project. It should give an idea of what you did to get to your final product, and what the greatest challenges were on the way. In at most 5 pages it should describe the following:

- Briefly describe your project topic.
- How does your implementation achieve its goal?
- What are some notable problems you encountered on the way? How did you solve them?
- What did you find out about the method in terms of its advantages, its limitations, and how to use it effectively?
- Briefly mention what resources did you used to learn about the technique. No need to include every link to everything you read, but I should get a general idea of how you figured it out, even if the answer ends up being pure experimentation!
- If working as a team: disclose who did what.

Date: 16.04.2023

Design and Implementation

To implement the cubemap this guide was followed [1]. To start the 6 faces(/images) of the cube are added to a vector and bound to a singular texture ID. The image is then loaded and bound to a texture map. The parameters for the cubemap define the wrapping of the images at the seem. A process of trial and error was necessary to find the correct orientation for the images in the cube. Given the already present camera movement, it is possible to change the perspective of the cubemap, however the perspective is limited to a front facing angle. The cubemap is then also given to the icicle shader to calculate the texture for the refraction which we'll come back to. An advantage of the cubemap over for instance a spheremap is it is easier and thus cheaper to implement. In general the cubemap requires more logic to implement, but it is in return better for creating the seems between the source images.

To implement the dynamic texture it was necessary to get the normals of the icicle. The icicle in itself had been created in blender, so a separate load object function was necessary to properly create the mesh for the object. For this the «tiny object loader» function was used [4]. Besides this the rest of the object loader function verifies the values and assigns the data of the vertices, indices and coordinates to the mesh. Next a loop in the initial game function reassigns the normals to a cylinder, to properly wrap the texture around the mesh. A loop was added for convenience to easily adjust the number of icicles by only making it necessary to add new coordinates to the new icicle. Then a time variable was added to the icicle shader, along with the necessary inputs to alter the diffuse texture of the water drop. An advantage with the dynamic texturing is that rather than a particle effect, it is a lot cheaper to make. The particle effect could be used to create realistic water drops, however they would need their own geometric details.

Lastly to implement the Fresnel refraction the cubemap had to be put as an input to the icicle shader. The shader then found the viewpoint vector by using the camera and a global vector for the VAO. Using this in the refraction function as the incident vector, together with the surface normal and ratio of indices of refraction [2], it was possible to calculate the refraction vector, which was then merged together with the cubemap texture and some additional alpha calculations to find the color output for the refraction effect. Transparency wouldn't achieve the same effect as refraction, especially as the refraction ratio in the code was made using an example from this site [1] for the ice refraction, which fit perfectly.

Another effect worth mentioning is the lens flare effect. The sun was added to the scene by creating a new mesh for a square by assigning vectors, indices and coordinates. Then a loop goes through the different lensflare images and the sun. In the loop the uniform position values are set and the textures and VAO are bound to each unique node in the scene. The flare shader which receives the inputs puts the images to the texture and scales the size given the scale and aspect ratio also set as inputs. Any additional information about the light sources were found from this source [3]

Discussion

The method for adding refraction uses the logic that the scene is rendered in 2D, so when bending the light-rays through the object they're in reality supposed to refract both on the way in, and on the way out. In the scene since the extra logic wouldn't be visible, it was deemed not necesary to add. An alternative to the way refraction was implemented is Raytracing, however this wasn't chosen for the project as the method is hard and expenssive to implement.

The sun image was put on a ball node with a hard coded coordinate in the cubemap. The ball then got a light node attached and a sprite to cover it. It was necessary with a depth mask to render the cubemap, but not for the sun, since the position in the render was behind all adjustable objects. However the lens flare images were added by calculating the path between the sun and the origin of the scene. So the same depth perception doesn't occur here. The lens flares are therefore visible in front of the icicles which is a visual flaw which could be fixed. A possible fix could be to create a dynamic cubemap texture, resulting in the sun and flare effects both being usable in the refraction.

The water droplets put as the texture for the visual texture in the icicles created a lot of confusion. Adding a realistic water drop didn't render properly, and adding any alpha color in between the drops had the effect of changing the size of the drops to somehow accomodate larger drops. By having the background to the drops be black it didn't change anything sizewise, and the resulting color on the icicle itself was a grey hue, which was an improvement giving it a more realistic look.

As mentioned when describing the implementation of the cubemap, the order of the faces of the cube had to be found through brute force. Some other elements this applied to were the creation of the mesh for the sun square and the back and forth issues when using multiple shaders. Additionally a lot of the time spent on problems in the project was to find errors, as open-gl is a hard framework to debug.

A limitation for the implementation of the scene was that a dripping effect from the icicle was never implemented. This would've been achieved by having a veil object beneath the icicle to continue displaying the dynamic effect.

Conclusion

When comparing the two images in the appendix the results were similar enough to be concluded as a great result [5][6]. The goal of the project was to investigate and get a better understanding of advanced visualization methods. As the three methods mentioned in the introduction was thoroughly implemented in the scene, the understanding of the methods, as the goal of the project, was achieved through the experience of using it in practice.

References

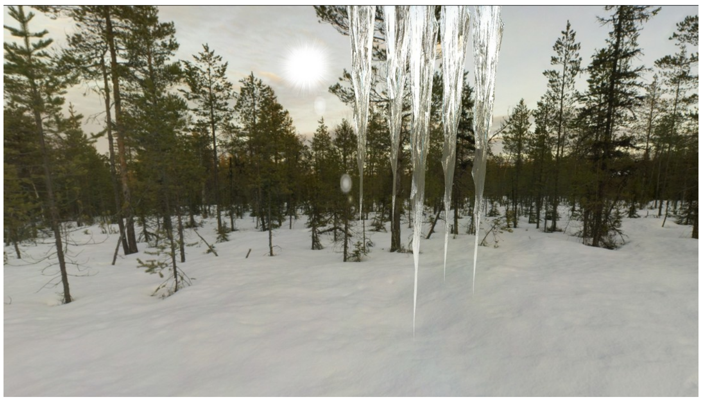
[1] The main source for creating the cubemap and how to add refraction in the shader:
https://learnopengl.com/Advanced-OpenGL/Cubemaps

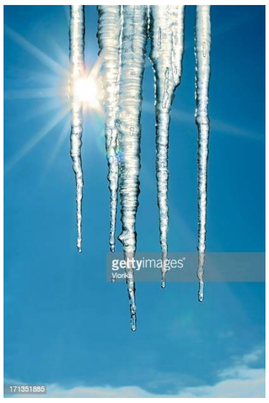
[2] Description of the refraction function:
https://registry.khronos.org/OpenGL-Refpages/gl4/html/refract.xhtml

[3] Explaining of light concepts:
https://learnopengl.com/Advanced-Lighting/Advanced-Lighting

[4] Library and how to use the tiny object loader function:
https://github.com/tinyobjloader/tinyobjloader

Appendix


[5] The resulting scene. Figur 1



[6] The inspiration. Figure 2

Date: 16.04.2023