Assignment 5: Material

Submission Period: 05.07.2022 12:00:00 - 19.07.2022 12:00:00 (Central European Summer Time)

General Information

- This is one of the graded assignments. In this assignment, one can collect a maximum of 20 points. **A submission that cannot be compiled will not be graded.**
- There are two different aids for that may be helpful for accomplishing the assignment. 1) a certain visual effect that is described in detail in the assignment, and can be verified visually using `npm start`; or 2) provided tests that can be executed using command `npm test`.
- Note that **passing all provided tests, or achieving a similar visual effect does not represent one can collect all points**. The evaluation of submissions will also run additional tests that are not provided to check the general robustness and soundness of a submission, such as possible edge cases. We recommend carefully considering an implementation if desire more points.
- **Dependent tasks assume previous results to be correct**. This means a subsequent implementation that depends on a previous incorrect implementation is also considered incorrect.
- It is prohibited to exchange solutions for the assignments with other students during the examination period. You must work on the assignments alone and independently and submit your own solution. If we discover any fraud or plagiarism in the submission, **both parties will be excluded** from the exam.
- If you found the task description ambiguity or potential mistakes in the provided code skeleton, please contact cg1ss22@medien.ifi.lmu.de or ask in the tutorial class for further clarification.

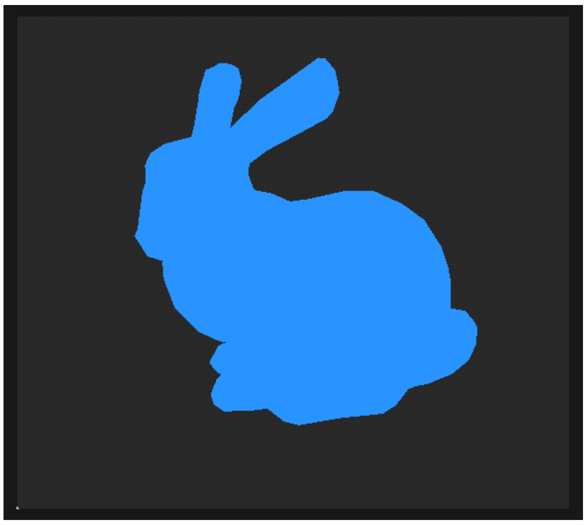
Erklärung über die eigenständige Bearbeitung

Ich erkläre hiermit, dass ich die vorliegende Arbeit vollständig selbstständig angefertigt habe. Quellen und Hilfsmittel über den Rahmen der Vorlesungen/Übungen hinaus sind als solche markiert und angegeben. Ich bin mir darüber im Klaren, dass Verstöße durch Plagiate oder Zusammenarbeit mit Dritten zum Ausschluss von der Veranstaltung führen.

In this assignment, we will continue the journey of the bunny we have been working over the semester. This time, we are going start from what we achieved in Figure 1a then use a texture to put color on the bunny so that it looks more lovely, as illustrated in Figure 1b.

As we discussed in the class, texture mapping is a process that queries the colors of a texture image using vertex UV coordinates. For the pixels within a triangle, we need to interpolate the coordinates for any position inside the triangle using barycentric interpolation. To deal with magnification and minification, we can create a MIP map hierarchy of the corresponding texture image. Then we can query the color using bilinear interpolation on a discrete MIP map level, or via trilinear interpolation between two discrete levels in the MIP map hierarchy. The tasks are distributed in three different files and marked with `// TODO:` comments:

- `src/renderer/raster.ts`
- `src/material/texture.ts`
- `src/linalg/interpolate.ts`


(a) A rasterized bunny without mapped texture.

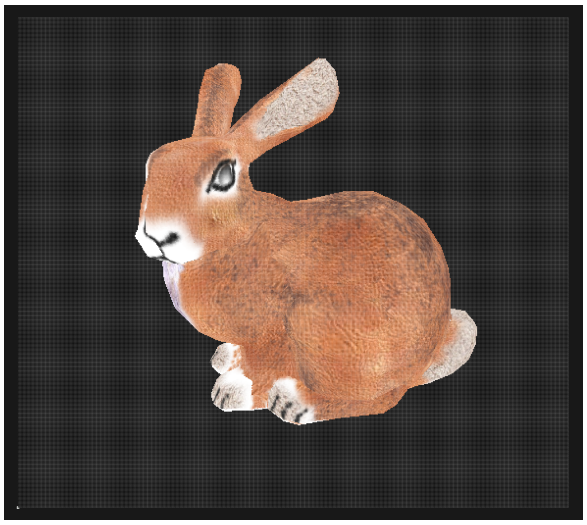

(b) A texture mapped bunny (with MIP map).

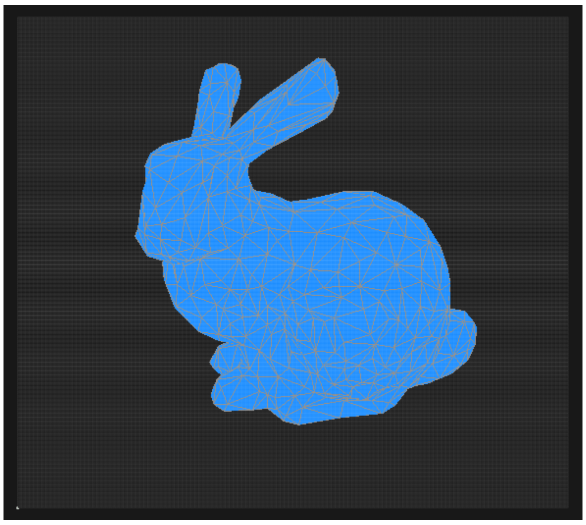
Figure 1: The final goal of the assignment.

Task 1: Fill Colors (10 Points, Easy)

In this task, we will apply naive texture mapping to the bunny and also draw it as a wireframe. To achieve this task, the following steps need to be implemented:

- (1p) Create and initialize a depth buffer (`initDepthBuffer`)
- (2p) Implement a depth test (`passDepthTest`)
- (3p) Compute the barycentric coordinates of a given triangle (`computeBarycentric`)
- (2p) Implement barycentric interpolation (`barycentricInterpolation`)
- (2p) Draw the mesh as a wireframe (`drawWireframe`)

The implemented effects are illustrated in Figure 2a and 2b.


(a) A rasterized bunny with wireframe.

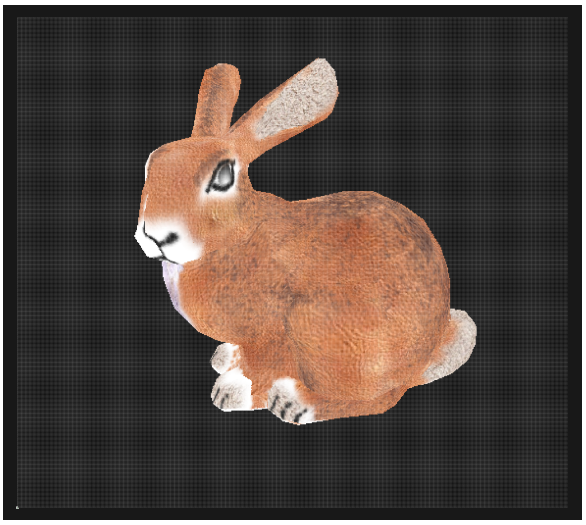

(b) A bunny without applied MIP map.

Figure 2: Intermediate results of the assignment.

Task 2: MIP Mapping (10 Points, Advanced)

Finally, we will complete the texture mapping of the bunny by adding a MIP map hierarchy. The result is illustrated in Figure 1b. To achieve this task, the following steps need to be implemented:

- (2p) Create a MIP map hierarchy (`createMipmap`)
- (1p) Implement linear interpolation between two values (`Lerp`)
- (1p) Implement linear interpolation between two vectors (`LerpV`)
- (2p) Implement bilinear interpolation (`queryBilinear`)
- (2p) Implement trilinear interpolation (`queryTrilinear`)
- (2p) Compute the estimated MIP map level (`computeMipmapLevel`)

The difference of the bunny rendered with/without applied MIP map can be found on the top of the bunny's head, on the side of the ear, and on the border of the body. The MIP mapped version is little bit blurry (antialiased), as shown in 1b (with MIP map) and 2b (without MIP map).

You can run the project by 1) installing all dependencies using `npm i` then 2) start and execute the project using `npm start`. Your browser will open a tab automatically. 3) A few additional unit tests are provided and can be used partially for testing your implementation using `npm test`.

An interactive demo can be found here:
`https://www.medien.ifi.lmu.de/lehre/ss22/cg1/demos/texture/`.

Submission Instructions

Please use the provided submission template and follow the submission instruction below to submit your solution to Uni2Work.

- Delete the two folders: **`node_modules`**, and **`build`**.
- Rename your folder to `cg1-assignment5-<your matriculation number>`, and compress everything as a single `.zip` file. For example, if your matriculation number is 12345678, then the zip-file's filename should be `cg1-assignment5-12345678.zip`.

  ✓ `cg1-assignment5-12345678.zip`
  ✗ `cg1-assignment5-<12345678>.zip`

  Your folder structure should be exactly like this (except the matriculation number):

```
cg1-assignment5-12345678/
   ├── .eslintignore
   ├── .eslintrc.json
   ├── .prettierrc.js
   ├── .vscode
   │   └── settings.json
   ├── README.pdf
   ├── assets
   │   ├── bunny.obj
   │   └── bunny.png
   ├── jest.config.js
   ├── package.json
   ├── package-lock.json
   ├── src
   │   ├── camera
   │   │   └── camera.ts
   │   ├── geometry
   │   │   ├── aabb.ts
   │   │   ├── mesh.ts
   │   │   └── object.ts
   │   ├── gl.ts
   │   ├── loader.ts
   │   ├── main.ts
   │   ├── material
   │   │   ├── material.ts
   │   │   └── texture.ts
   │   ├── math
   │   │   ├── interpolate.ts
   │   │   ├── mat4.ts
   │   │   ├── quaternion.ts
   │   │   ├── utils.ts
   │   │   └── vec4.ts
   │   ├── renderer
   │   │   ├── raster.ts
   │   │   └── scene.ts
   │   └── view.ts
   │
```

```
├── test
│   ├── texture.test.ts
│   └── utils.ts
├── tsconfig.json
└── webpack.config.js
```