Assignment 4: Rasterization

Submission Period: 21.06.2022 12:00:00 - 05.07.2022 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.

Task: Towards Screen-space (20 Points, Advanced)

In this task, we are going to implement a basic rasterizer for the bunny that we have transformed from the last assignment.

As we discussed in the class, a rasterization process traverses all existing triangles in a given scene, then draws each triangle one by one. To draw a triangle, we can either use a scan line algorithm, or use a point-in-triangle assertion based algorithm. In this assignment, we will implement the point-in-triangle assertion based drawing method.

Look for `// TODO:` comments in the `src/renderer/raster.ts` and `src/geometry/aabb.ts` files. Note that it is not allowed to introduce any new dependencies or to use APIs from `three.js`. The implementation is graded based on the following requirements:

- Implement the constructor of AABB structure (2p)
- Implement the intersection assertion for two given AABBs (3p)
- Create and initialize a frame buffer (2p) in function `initFrameBuffer`
- Implement function `vertexShader` that transforms a given vertex from model space to screen space (3p)
- Implement back face culling (2p) in function `isBackFace`
- Implement view frustum culling (3p) in function `isInViewport` to test whether a triangle is in the screen space.
- Implement point-in-triangle assertion (3p) in the `isInsideTriangle`.
- Update the buffer to draw a pixel on the frame buffer (2p) in the `updateBuffer`

As always, here are some hints for you:

- 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`.
- When you uncheck the "screenSpace" option from the top right menu, you will see a bunny in projection space, as shown in Figure 1.
- We recommend implement `initFrameBuffer` first, then use the provided unit tests to guarantee a sound implementation of AABB data structure. Making sure earlier steps work as expected is better suited for proceeding to the next successful stage.
- An interactive demo can be found here: https://www.medien.ifi.lmu.de/lehre/ss22/cg1/demos/raster/

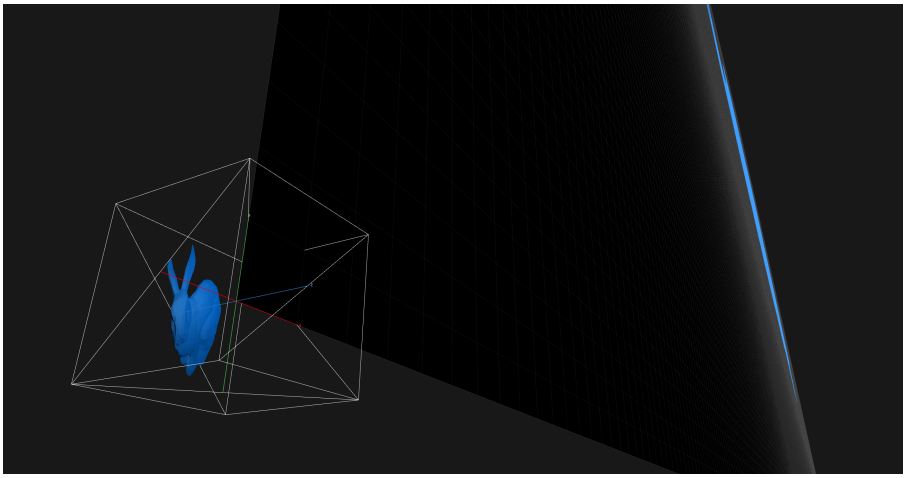

Figure 1: The bunny mesh in the projection space which should be rendered on the “screen”.

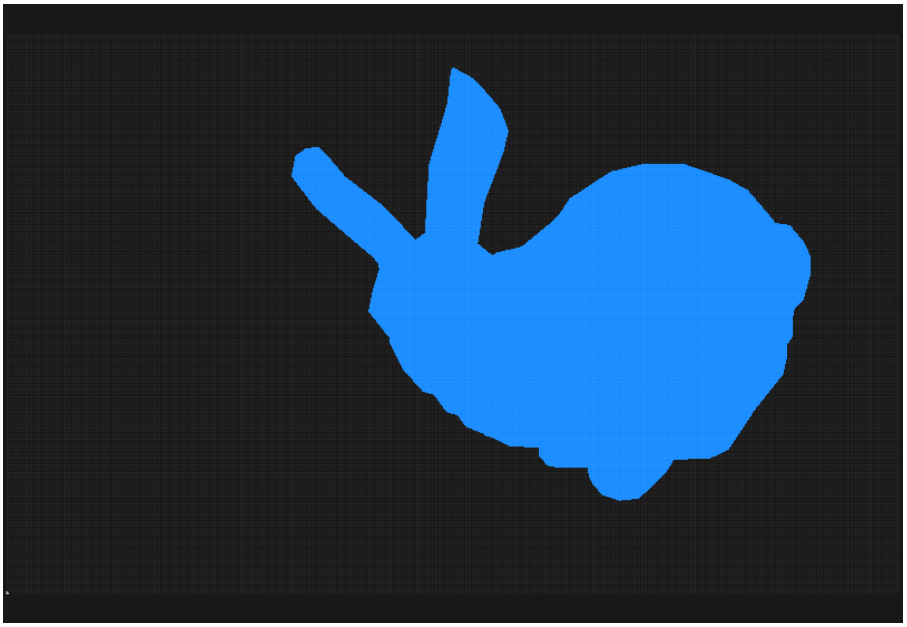

Figure 2: The rasterized bunny shape when the rasterization process is fully implemented [live demo].

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-assignment4-<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-assignment4-12345678.zip`.

  ✔ `cg1-assignment4-12345678.zip`
  ✘ `cg1-assignment4-<12345678>.zip`

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

```
cg1-assignment4-12345678/
├── .eslintignore
├── .eslintrc.json
├── .prettierrc.js
├── .vscode
│   └── settings.json
├── README.pdf
├── assets
│   └── bunny.obj
├── jest.config.js
├── package.json
├── package-lock.json
├── src
│   ├── camera
│   │   └── camera.ts
│   ├── geometry
│   │   ├── aabb.ts
│   │   ├── mesh.ts
│   │   └── object.ts
│   ├── gl.ts
│   ├── main.ts
│   ├── math
│   │   ├── mat4.ts
│   │   ├── quaternion.ts
│   │   ├── utils.ts
│   │   └── vec4.ts
│   ├── renderer
│   │   ├── raster.ts
│   │   └── scene.ts
│   └── view.ts
├── test
│   ├── raster.test.ts
│   └── utils.ts
├── tsconfig.json
└── webpack.config.js
```