Graded Assignment 04: Rasterization

Submission Period: 28.05.2021 00:00 - 06.06.2021 23:59 (Anywhere on Earth, AoE)

General Information

- This is one of the graded assignments. You need to collect above 50 points to pass with 4.0, and a score of 90 points or greater for a 1.0. In this graded assignment, you can collect a maximum of 20 points.
- Please **sign "Erklärung über die eigenständige Bearbeitung" in this document. A submission without electronic signature will not be graded, and a re-submission is *NOT* possible.** (Hint: You can use the macOS's built-in application Preview or Adobe Acrobat Reader DC on Windows. Handwritten signatures in a re-scanned document will not be processed.)
- It is prohibited to exchange solutions for the graded assignments with other students during the examination period. You must work on the graded assignments alone and independently and submit your own solution. If we discover any fraud or plagiarism in the submission, you will be asked to join an additional oral exam. In the worst case, both parties will be excluded from the course.
- We designed the assignment for and recommend you to invest *not* more than **8 hours** to work on this assignment. You can work on the assignments in German or English. Mixing both languages is allowed as well.
- If you have any questions regarding technical issues, please contact the assistants of the course immediately. The fastest way to do so, is the discussion forum in Moodle.

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.

Ort, Datum Unterschrift

Ich habe insgesamt etwa _____ Stunden Arbeitszeit für diese Abgabe aufgewendet. (Diese Information ist freiwillig, rein informativ und hat keinen Einfluss auf die Benotung.)

1 Task 1: Practice (20 Points, Medium)

In this task, you are going to implement a basic rasterizer for the bunny that you have transformed last week, rasterizing it from a 3D world into a simulated 2D screen.

Note that you can use any built-in APIs from JavaScript, but it is not allowed to introduce any new dependencies or to use APIs from `three.js`. Additionally, while doing the task, please ensure that the ESLint auto fix option is activated. It helps you to find problems in your code and can format it automatically (discussed in tutorial 1, page 30-32).

As both lecture and tutorial elaborated, a rasterization process traverses all existing triangles in a given scene, then samples and 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 utilize the point-in-triangle assertion based drawing method. The tasks are located in the `src/renderer/raster.ts` and `src/geometry/aabb.ts` and marked with `// TODO:` comments. A submission will be graded on completion of the following tasks:

- (2p) Implement the AABB constructor (`AABB.constructor`) = axis aligned bounding box
- (3p) Implement the intersection assertion for two given AABBs (`AABB.intersect`)
- (2p) Create and initialize a frame buffer (`Rasterizer.initFrameBuffer`)
- (3p) Implement a vertex shader that transforms a given vertex from model space to screen space (`Rasterizer.vertexShader`)
- (2p) Implement back face culling (`Rasterizer.isBackFace`)
- (3p) Implement view frustum culling to test if a triangle should be drawn in the screen space (`Rasterizer.isInViewport`)
- (3p) Implement point-in-triangle assertion (`Rasterizer.isInsideTriangle`)
- (2p) Update the buffer to draw a pixel on the frame buffer (`Rasterizer.updateBuffer`)

Note that each graded function is evaluated as a unit. This means that intermediate results are not considered in the grading.

As always, here are some hints for you:

- You can run the project by 1) installing all dependencies using `npm i` and 2) starting and executing the project using `npm start`. One can ignore any severity vulnerabilities warning. Your browser will open a tab automatically, and the page renders a screen consisting of a grid.
- If you uncheck the "screenSpace" option from the menu, you will see a bunny in projection space, as shown in Figure 1.
- You can use or change the code in `src/main.ts`, `src/gl.ts`, `src/view.ts` for any testing purpose, but it is not necessary for the completion of the task. If all tasks are implemented correctly, the rasterization will work as shown in Figure 2 or this live demo.

- Notably, the provided live demo is one of the completion indicators for the submission. A submission that behaves exactly like the live demo, however, does not necessarily represent a submission that can achieve all the points, as there are design details hidden in the task that may not be represented by the visualization.
- We recommend implement `Rasterizer.initFrameBuffer` first, then think about the order in which the tasks should be implemented. Making sure earlier steps work properly is the key to proceed to the next stage.
- Also, be careful about writing for loops. If there is an infinite loop, the browser will pause and stop working. In the worst case, this could crash the entire computer.

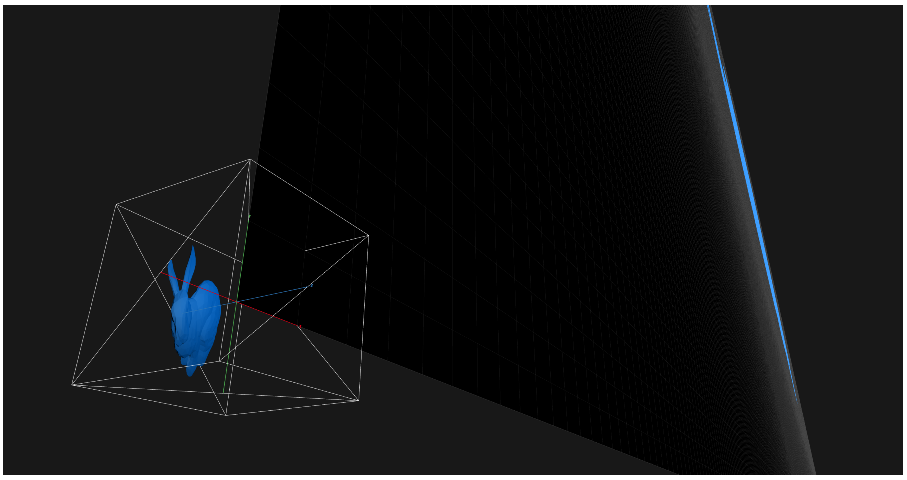

Figure 1: The bunny mesh in the projection space which should be rendered on the "screen".

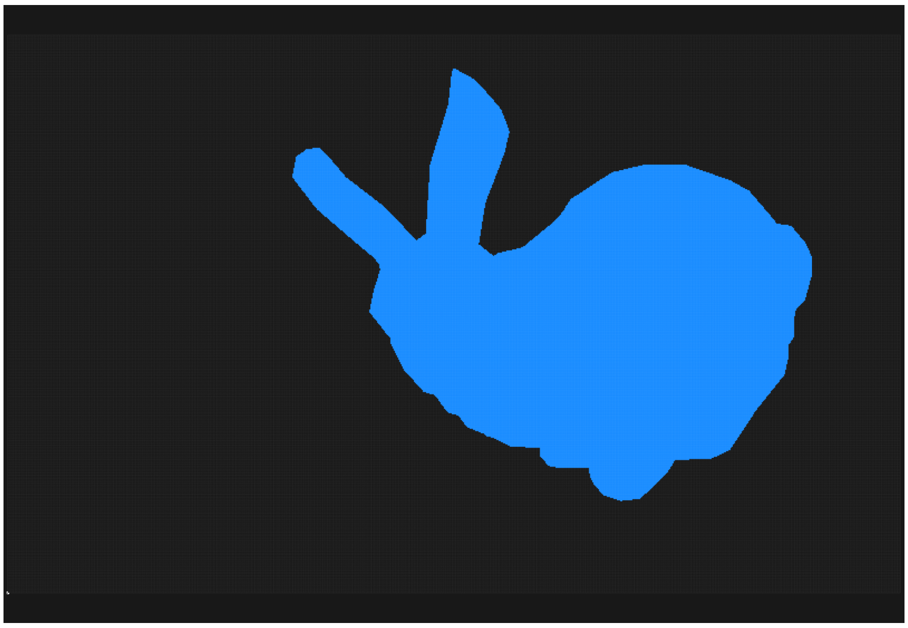

Figure 2: The rasterized bunny shape when the rasterization process is fully implemented [live demo].

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`.
- For example, your folder structure should be exactly like this (except the matriculation number):

```
cg1-assignment4-12345678/
    ├── assets
    │   └── bunny.obj
    ├── .eslintignore
    ├── .eslintrc.json
    ├── jest.config.js
    ├── package.json
    ├── package-lock.json
    ├── .prettierrc.js
    ├── README.md
    ├── README.pdf          <- your signature
    ├── src
    │   ├── camera
    │   │   └── camera.ts
    │   ├── geometry
    │   │   ├── aabb.ts     <- your code
    │   │   └── mesh.ts
    │   ├── linalg
    │   │   ├── mat.ts
    │   │   ├── quaternion.ts
    │   │   ├── utils.ts
    │   │   └── vec.ts
    │   ├── renderer
    │   │   ├── raster.ts   <- your code
    │   │   └── scene.ts
    │   ├── gl.ts
    │   ├── main.ts
    │   └── view.ts
    ├── .vscode
    │   └── settings.json
    ├── tsconfig.json
    └── webpack.config.js
```