Graded Assignment 02: Mesh

Submission Period: 07.05.2021 00:00 - 14.05.2021 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 **5 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: Theory (10 Points)

Consider a pyramid with a square base. The center of the base lies at the origin and extends 3 into each direction x, $-x$, z, $-z$. The tip of the pyramid lies 5 into the y direction. The pyramid is displayed on the left side in Figure 1.

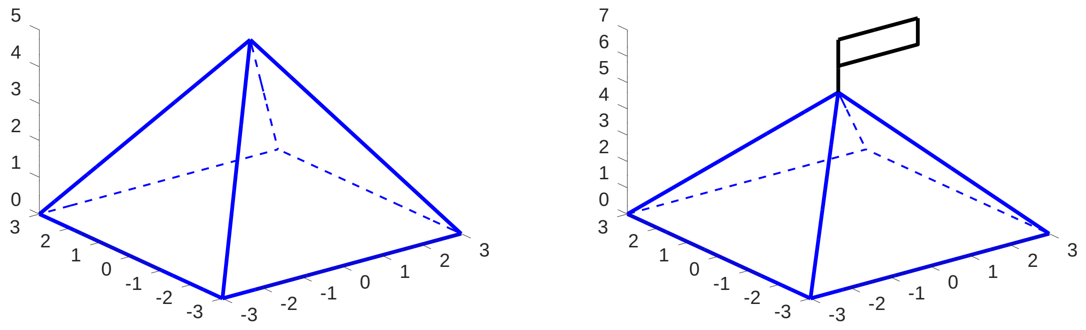

Figure 1: The pyramid as described in Task 1. The left side shows only the pyramid, the right side additionally displays the flag from Task 1.3.

1.1 Vertices and Faces (1.5 Points, Easy)

Write down a vertex as well as a corresponding face list, i.e., write an OBJ file to represent this pyramid (you can omit texture coordinates and normals).

1.2 Types of Meshes (5 Points, Easy)

Can we represent the pyramid without changing its shape using the following meshes? You may add vertices, edges and faces. If we can use that type of mesh, please specify the minimal number of vertices and faces needed and briefly explain in a sentence where you added structure. If it is not possible, briefly explain why not.

a) (1.5p) triangle mesh

b) (1.5p) quad-dominant mesh

c) (1.5p) quad mesh

Which type of mesh did you use in Task 1.1?

1.3 Topology (3.5 Points, Medium)

We now want to add a flag on top of the pyramid by adding a 1 long line as a flagpole and a 1 high and 2 wide rectangle in x direction as a flag as shown on the right side of Figure 1. Briefly answer the following questions.

a) (1p) How can we add the flag to the OBJ file from Task 1.1, i.e., which type of objects to we have to add (e.g., `v`, `vt`,...)?

b) (1p) Why is it not possible (without changing the shape of the pyramid) to use any of the meshes named in Task 1.2 for representation?

c) (1.5p) Can you think of a way to circumvent this issue while keeping the pyramid visually similar?

2 Task 2: Practice (10 Points, Medium)

In this task, you are going to implement a Wavefront object file loader that loads a `.obj` file into a *polygon soup*, i.e. a collection of polygons.

Specifically, implement the `LoadOBJ` function in the `src/geometry/mesh.ts` file. Note that you must implement the function without introducing any additional dependencies. Additionally, while doing the task, please ensure that the ESLint auto fix is activated. It helps you to find problems in the code and can format the code automatically (as discussed in tutorial 1, page 29-31).

In the `src/geometry/mesh.ts` file:

- A `Vertex` conveys geometric information, such as position, UV, and normal. A vertex position is of the type `Vector` using a four-dimensional homogeneous representation; a UV coordinate is also a position and also of type `Vector`. A normal is again of type `Vector`, however, it represents a vector and not a point.
- A `Face` is an interface representing a polygon, which has an arbitrary number of vertices. The class `Triangle` implements `Face`, and the number of vertices is equal to 3.
- A `Mesh` is an interface representing a polygon soup, and the `TriangleMesh` implements the `Mesh` interface representing a triangle soup.

To simplify the object file loader, you can assume that the input `data` string complies with the following constraints:

- The first character of a line can only be either: `v`, `vt`, `vn`, or `f`;
- The coordinates are separated by a single space;
- The indices of a vertex are connected by a slash (`/`);
- The vertices for a face are separated by a single space;

Your implementation is graded based on the following properties of the returned `TriangleMesh`:

- (1.5p) The returned triangle mesh contains the correct number of positions, UVs, normals, and faces for a given input.
- (1.5p) All vertex positions from the given input are stored in the `positions` property of the `TriangleMesh`.
- (1.5p) All UV coordinates are stored in the `uvs` property of the `TriangleMesh` for a given input.
- (1.5p) All normals are stored in the `normals` property of the `TriangleMesh` for the given input.

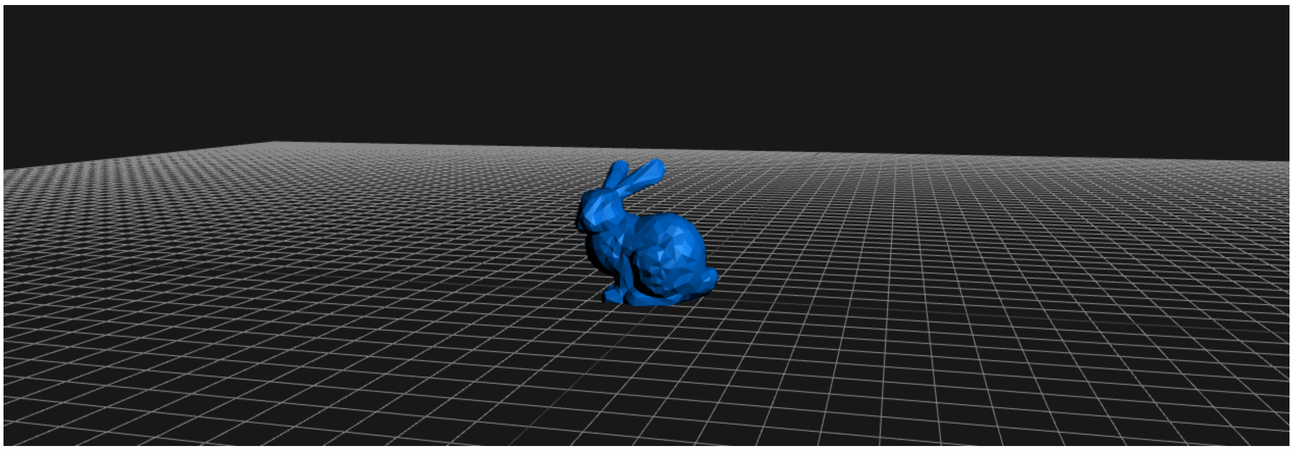

Figure 2: The final rendered scene if `LoadOBJ` loads the given bunny mesh successfully.

- (2p) All faces are stored in the `faces` of the `TriangleMesh` for the given input.
- (2p) The given bunny is rendered properly without misoriented faces for the given input.

Here are some hints:

- 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) The initial scene contains a blue tetrahedron and a grid plane with a dark background.
- You can use or change the code in `src/main.ts` for any testing purpose. If the `LoadOBJ` function implementation is correct, the scene will render an additional bunny as showed in Figure 2.
- In JavaScript/TypeScript, an array offers the method `push()` to append elements to the calling array and a string has the method `split()` that splits a string into substrings using the specified separator and returns them as an array. To turn a string into a number, one can use `parseFloat` for floating numbers, or `parseInt` for integers.

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-assignment2-<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-assignment2-12345678.zip`.
- For example, your folder structure should be exactly like this (except the matriculation number):

```
cg1-assignment2-12345678/
    ├── ANSWER.md           <- your text answers
    ├── assets
    │   └── bunny.obj
    ├── .eslintignore
    ├── .eslintrc.json
    ├── jest.config.js
    ├── package.json
    ├── package-lock.json
    ├── .prettierrc.js
    ├── README.md
    ├── README.pdf          <- your signature
    ├── src
    │   ├── geometry
    │   │   └── mesh.ts     <- your code
    │   ├── linalg
    │   │   ├── mat.ts
    │   │   ├── utils.ts
    │   │   └── vec.ts
    │   └── main.ts
    ├── tsconfig.json
    ├── .vscode
    │   └── settings.json
    └── webpack.config.js
```