Assignment 2: Mesh

Submission Period: 24.05.2022 12:00:00 - 07.06.2022 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: Dealing with Polygon Meshes (20 Points, Advanced)

In this task, we are going to implement a simplified Wavefront object file loader that parses a `.obj` file as a *polygon mesh*.

Specifically, look for `// TODO:` comments in the `src/geometry/mesh.ts` file.

A `Vert` conveys geometric information, such as position, UV coordinates, and normal. A vertex position is of the type `Vec4`; a UV coordinate is also a position, and also of type `Vec4`. A normal is again of type `Vec4`. However, it represents a vector and not a point.

A `Face` represents a polygon which contains a list of vertices.

A `Mesh` represents a collection of faces.

To simplify the parsing complexity, we can assume that the input `data` string subjects to the following constraints:

- The first character of each line can only be either: `v`, `vt`, `vn`, or `f`;
- The coordinates are separated by a single space character " ";
- The vertex indices are connected by a slash character "/";
- The vertices for a face are separated by a single space character " ";
- `f` always comes after all other attributes.
- No missing data, all coordinates and indices are complete.

The implementation is graded based on the following requirements:

- The returned polygon mesh contains the expected number of positions (1p), normals (1p), UVs (1p), and faces (1p) for a given input
- All vertex positions, normals, UV coordinates from the given input are stored in the `positions` (1p), `normals` (1p), and `uvs` (1p) properties of the `Mesh` correspondingly. The elements order of these attributes (each 1p, total 3p) must match their appearing order of the given data
- All polygon faces are stored in the `faces` of the `Mesh` for the given input (2p), and the order of faces and corresponding vertices matches their appearing order of the provided data (2p).
- Primitive iterator "`Face.primitives`" iterates all triangle primitives of a face (2p). If its callback function "`iter`" returns false, then stop the iteration (1p).
- Primitive iterator "`Mesh.primitives`" iterates all triangle primitives of all faces (2p). If its callback function "`iter`" returns false, then stop the iteration (1p).

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) A few additional unit tests are provided and can be used partially for testing your implementation using `npm test`.

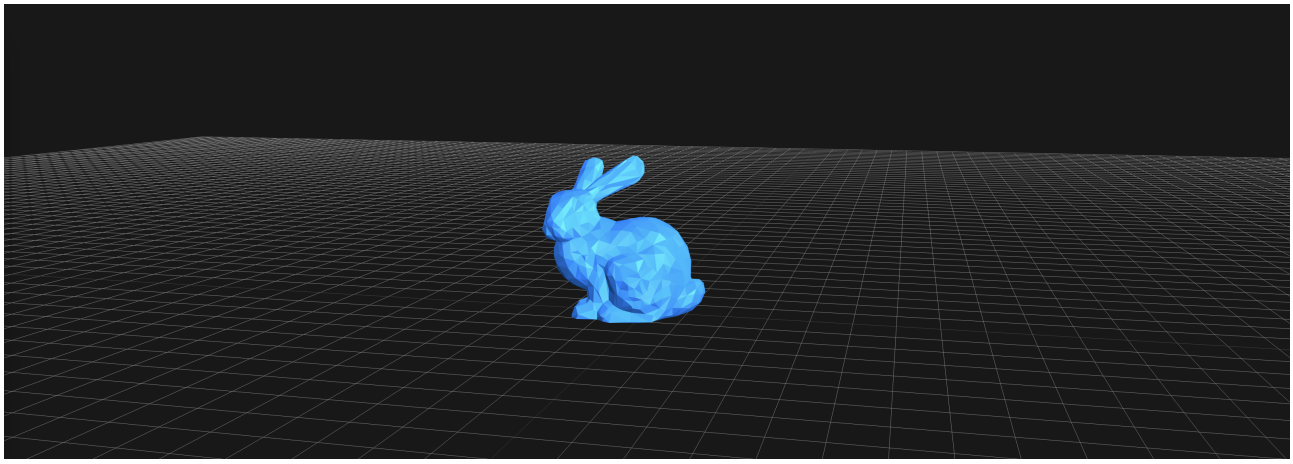

Figure 1: When the bunny is rendered.

- If the implementation meets the expectation, the scene will render an additional bunny, as shown in Figure 1, not vice versa.
- 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 (no need to import any dependencies).
- An interactive demo can be found here: `https://www.medien.ifi.lmu.de/lehre/ss22/cg1/demos/mesh/`

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`.

  ✔ `cg1-assignment2-12345678.zip`
  ✘ `cg1-assignment2-<12345678>.zip`

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

```
cg1-assignment2-12345678/
    ├── .eslintignore
    ├── .eslintrc.json
    ├── .prettierrc.js
    ├── .vscode
    │   └── settings.json
    ├── assets
    │   └── bunny.obj
    ├── README.pdf
    ├── jest.config.js
    ├── package-lock.json
    ├── package.json
    ├── src
    │   ├── geometry
    │   │   └── mesh.ts    <- your code
    │   ├── math
    │   │   ├── utils.ts
    │   │   └── vec4.ts
    │   └── main.ts
    ├── tsconfig.json
    └── webpack.config.js
```