LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

INSTITUT FÜR INFORMATIK
MEDIENINFORMATIK
PROF. DR. ANDREAS BUTZ, CHANGKUN OU, DAVID ENGLMEIER
COMPUTERGRAFIK 1, SOMMERSEMESTER 2020

Online-Hausarbeit 1: Build a Solar System

Bearbeitungszeitraum: 06.07.2020 00:00 Uhr - 10.07.2020 23:59Uhr

General Informations

- There are three graded assignments allowing for a total score of 200 regular points and 20 bonus points. You will need 100 points to pass with 4.0, every additional 10 points increase the grade to the next step, while a score of 190 points or greater results in 1.0.
- 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, both parties will be excluded from the course.
- For programming tasks, you must organize and comment your code well by following additional comments/instructions in the code skeleton. Weird coding style or changing the provided template may not be well tested or graded. In addition, any comments in your code must be written in English.
- For non-programming tasks, you can answer in German or English or mix.
- If you have any questions regarding technical issues, please contact the course assistants immediately.

1 Overview

This is the first graded assignment for the CG1. You can achieve a maximum of 55 points from this graded assignment. The estimated completion time for this assignment is about 2 to 4 hours. Depending on your familiarity with the subject and possible coding issues, your time might increase accordingly. If you struggle with one task, get yourself some fresh air and come back to it later. To reduce time pressure, you can submit your work until 10.07.2020 23:59.

For our own information (and without any influence on the grade!), please roughly record the actual time (in hours) you spent on this assignment and let us know on the last page below your signature.

This assignment aims at walking you through all of the basic concepts needed for a computer graphics programming project. While solving the tasks, you will also set up all necessary tools and get familiar with the workflow of creating a graded submission. The third graded assignment will be a programming task again, so all the work you invest here for setting things up will pay off twice.

Creating a solar system is a classic computer graphics project, and you can find a lot of demos online. In this graded assignment, you are going apply the concepts that you have learned from the course to implement a solar system, including transformations, a scene graph, geometry, texture mapping, local illumination, and shader programming.

2 Code Skeleton

The work package of this graded assignment includes a code skeleton that serves as a starting point.

To get started, you need to install NodeJS[1]. Then use **`npm i`** to install all required dependencies in a system console, e.g., Terminal on macOS/Linux, PowerShell on Windows. Next, you can use **`npm start`** to run the code skeleton. The command compiles the entire code skeleton and opens a new web page automatically to demonstrate your results. More conveniently, it also automatically refreshes the web page while you are changing your implementation.

You can use your favorite way to program. To give you some options, we recommend you to use a code editor (e.g., VSCode[2], etc.) or an IDE (e.g., WebStorm[3], etc.) that provide convenient features such as auto-completion, jump to definition, etc.

The code skeleton includes many different files: the `src/assets/` folder contains the textures for creating the solar system and the `src/shaders/` folder contains OpenGL shader files. However, you do not need to read/touch `package.json`, `package-lock.json`, `webpack.config.js`, and `src/assets`.

In the `src/renderer.js` file, the code skeleton provides the necessary rendering facilities, including the WebGL renderer, the scene graph instance, the camera, orbit controls, and the render loop. You can read this file to gain more understanding about the code structure, but you also do not need to change any code inside this file.

For this graded assignment, you work mainly in `src/main.js`, and a tiny amount of coding will be done in `src/shaders/saturn_ring.vs.glsl`, and `src/shaders/saturn_ring.fs.glsl`. All of your tasks are specified within the corresponding `// TODO:` comments.

In the `src/main.js` file, a `SolarSystem` class extends the `Renderer` class. In the `constructor()` method, we provided the needed parameters in `this.params`. `this.sun` and `this.planets` are reserved variables for you to fill. You can read more about them in the Grading Details section.

The OpenGL shaders are located in the `src/shaders/` folder, and the code skeleton has imported them in `main.js` as `ringVS` and `ringFS`, use them whenever you need them. You can find more details about the shader programming tasks in the Grading Details section.

3 Result Reference

Once you completed this graded assignment, you should get a result that is very similar to Figure 1. You can also find an online interactive demo to compare your implementation and an expected reference result at https://www.medien.ifi.lmu.de/lehre/ss20/cg1/demo/grading/solar-system/.

[1] https://nodejs.org/en/download/package-manager/
[2] http://code.visualstudio.com/
[3] https://www.jetbrains.com/webstorm/

Figure 1: A reference result of the implemented solar system.

4 Grading Details (50 Points)

This section lists the grading details regarding the implementation. Note that you can always skip (or postpone) some of the features (or tasks) just as you can skip some of the tasks in a traditional written exam if you have no idea how to implement them correctly.

1. create a starry sky background **(total: 5p)**
   - use the given params `this.params.universe.radius`, `this.params.segment` for geometry (1p)
   - use the texture from `this.params.universe.texture` as the background material for the starry sky (1p)
   - reduce a possible blur effect occuring for the background in all directions (1p)
   - create the background 3D object (1p) then add it to the scene graph (1p)
2. create a sun as the center of the solar system **(total: 5p)**
   - use the given params: `this.params.sun.radius`, `this.params.segment` for geometry (1p)
   - use the texture from `this.params.sun.texture` as the sun material (1p)
   - reduce a possible blur effect occuring at the poles (1p)
   - create the 3D sun object and assign to `this.sun` (1p) then add it to the scene graph (1p)
3. create a light to simulate the sun light **(total: 2p)**
   - create a point light using the given params in `this.params.sun.light` (1p)
   - create ambient light using the given params in `this.params.sun.light` (1p)

4. create the eight planets using provided textures, and move them with a given distance to the sun **(total: 7p)**
    - use the given params in `this.params.planets.<name>.radius`, `this.params.segment` for the planet geometry (1p)
    - use the texture from `this.params.planets.<name>.texture` then create the correct material to show the phong illumination model (1p)
    - reduce a possible blur effect occuring at the poles (1p)
    - move each planet around the sun with a given distance (`this.params.planets.<name>.distance`) (1p)
    - create each planet and assign them to `this.planets.<name>.planet` (2p) then add eight of them to the scene graph (1p)
5. draw a (circular) orbit for each planet **(total: 7p)**
    - use the given params in `this.params.planets.<name>.distance`, `this.params.segment` for the circle geometry (1p)
    - use the given params in `this.params.orbit` then create the line basic material (1p)
    - create each orbit and assign them to `this.planets.<name>.orbit` (2p) then add eight of them to the scene graph (1p)
    - transform each orbit and place it in x-z plane (1p)
    - manipulate the vertices to guarantee the orbit is drawn as a closed circle (1p)
6. create the moon and its orbit **(total: 6p)**
    - create the moon and assign the 3D moon object to `this.planets.earth.moon.planet` using the given params (`this.params.planets.earth.moon`, `this.params.segment`). The moon's material should show the phong illumination model (1p)
    - set the moon to the correct position with a given distance to the earth (1p)
    - add the moon to the scene graph (1p)
    - create the moon orbit and assign it to `this.planets.earth.moon.orbit` using the given params `this.params.planets.earth.moon.distance` and `this.params.orbit` (1p)
    - transform the moon's orbit that is located around the earth in the x-z plane and make sure the orbit is drawn as a closed circle (1p)
    - add the moon's orbit to the scene graph (1p)
7. create the saturn ring and apply corresponing texture mapping with a customized fragment shader **(total: 8p)**
    - use the given params `this.params.planets.saturn.ring` to create the saturn ring using ring geometry and a shader material then assign it to `this.planets.saturn.ring` (1p)
    - make sure the saturn ring is rendered on both sides (1p)

- pass the needed uniforms to the customized shaders (1p)
- write a customized vertex shader that passes the position to fragment shader (1p)
- write a customized fragment shader that performs the UV mapping (2p)
- transform the saturn ring so that is located around the saturn and that it is rotating around x axis with the given angle in `this.params.planets.saturn.ring` and make sure the orbit is drawn in a closed circle (1p)
- add the saturn ring to the scene graph (1p)

8. add self rotation to the sun **(total: 1p)**
   - use the given params `this.params.sun.selfRotate` and add self rotation to the sun around the y-axis. The given parameter is the rotation angle (radians) of each frame. (1p)
9. add self rotation to the eight planets **(total: 2p)**
   - use the given params `this.params.planets.<name>.selfRotate` and add self rotation to each planet around y-axis. The given parameters are the rotation angle (radians) of each frame. (2p)
10. add revolution to the eight planets **(total: 3p)**
    - use the given params `this.params.planets.<name>.revolution` and add revolution (2p) to each planet (1p) that rotates around the sun. The given parameters are the rotation angle (radians) of each frame.
11. add self rotation and revolution to the moon **(total: 3p)**
    - use the given params `this.params.planets.earth.moon.selfRotate` and add self rotation to the moon (1p). The given parameter is the rotation angle (radians) of each frame.
    - use the given params `this.params.planets.earth.moon.revolution` and add revolution to the moon (1p). The given parameter is the rotation angle (radians) of each frame.
    - move the orbit while the earth is moving by changing the position of the moon (1p).
12. move the saturn ring while saturn is moving **(total: 1p)**
    - move the saturn ring while saturn is moving by changing the position of the saturn ring (1p)

5 Submission Instructions (Bonus: 5 Points)

Please use the provided work package. You can use the Markdown editor typora to open and edit a Markdown file for a graphical user interface if you are not familiar with Markdown syntax.

Make sure that your submission is complete. In particular, the signed "Erklärung über die eigenständige Bearbeitung" must be included in your submission. A submission without signature will not be graded, and a re-submission is NOT possible. Your signature must be a non-editable element in the document. Specifically, you can either print the document, sign then rescan the document as a PDF file that replaces this document with your hand signature; Or, you can also use electronic signature by using the Preview app on macOS or Adobe Acrobat Reader DC on Windows (Linux users should be able to find tools for electronic signatures on their own).

Following the additional submission instructions below will reward you with 5 bonus points.

- Submission works without compiling errors using `npm run build` and without runtime errors in the Chrome/Firefox browser's console (1p).
- Filling your implementation directly after the `// TODO:` comments without changing other places of the code skeleton, document your implementation by adding additional comments whenever you think they are helpful (1p).
- Remove all unnecessary constants, variables, and comments in your implementation. Exclude any additional files (e.g. `node_modules`), and your submission folder structure should be exactly the same as the provided work package. (1p).
- Report your implementation in the `CHECKLIST.md` by checking your implemented features among the listed features. Your checked features in the `CHECKLIST.md` should match your actual implementation (1p).
- Rename your folder name to `abgabe1-<your matriculation number>`. Then submit your solution as a `.zip` file via Uni2Work (1p).

As an example, assume your matriculation number is 12345678, then your ZIP filename should be `abgabe1-12345678.zip`, and the decompressed folder structure should look exactly like this:

```
abgabe1-12345678/
├── CHECKLIST.md
├── README.pdf              // with your signature
├── package-lock.json
├── package.json
├── src
│   ├── assets
│   │   ├── earth.jpg
│   │   ├── jupiter.jpg
│   │   ├── mars.jpg
│   │   ├── mercury.jpg
│   │   ├── moon.jpg
│   │   ├── neptune.jpg
```

```
│   │   ├── saturn.jpg
│   │   ├── saturn_ring.png
│   │   ├── sun.jpg
│   │   ├── universe.jpg
│   │   ├── uranus.jpg
│   │   └── venus.jpg
│   ├── main.js
│   ├── renderer.js
│   └── shaders
│       ├── saturn_ring.fs.glsl
│       └── saturn_ring.vs.glsl
└── webpack.config.js
```

Erklärung über die eigenständige Bearbeitung

Online-Hausarbeit 1

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.

München, 10.07.20

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.)