Final submission: Interactive Computer Graphics Exercise

Summer Semester 2023

General description.

The goal of the assignment is to develop a Typescript web application that implements the content learned in the Interactive Computer Graphics course. In particular, this relates to the following topics:

- Scene graph
- Lighting models
- Materials
- Software library for linear algebra
- Interactivity

Assessment

The assignment consists of the project and an individual code review.

To pass, at least the *minimum* requirements of the project must be met. Additional *enhancements* may be met to upgrade the grade.

All minimum requirements combined are worth up to 50 points. The point breakdown can be found in the description of each requirement.

The total score is the sum of all individual requirements. A total score of 100 points is the highest score to be achieved and is equivalent to 1.0 (*very good*).

Submission

The project submission is made via the repository. A Readme file documents your project submission. It explains how your submission is structured, as well as how to use (install and operate) the program.

The delivery is done in groups of maximum 3 members for bachelor students and individual for master students.

The result of the project must be submitted by **September 15, 2023**, 23:59. For this purpose, it has to be uploaded to the GitLab Server of the Institute of Computer Science. Alternative submission methods are not allowed. The submission is done via a given repository. Access to the submission repository will be revoked after the submission deadline.

Non-performing implementations will have points deducted. More on that in the **non functional requirements**.

Exam dates

A code interview will be held on the days **September 18, 2023 - September 22, 2023**. In the event that more students wish to take the exam than there are dates available, additional exam days will be set up after the dates listed.

Further details on exam dates will be announced via WueCampus.

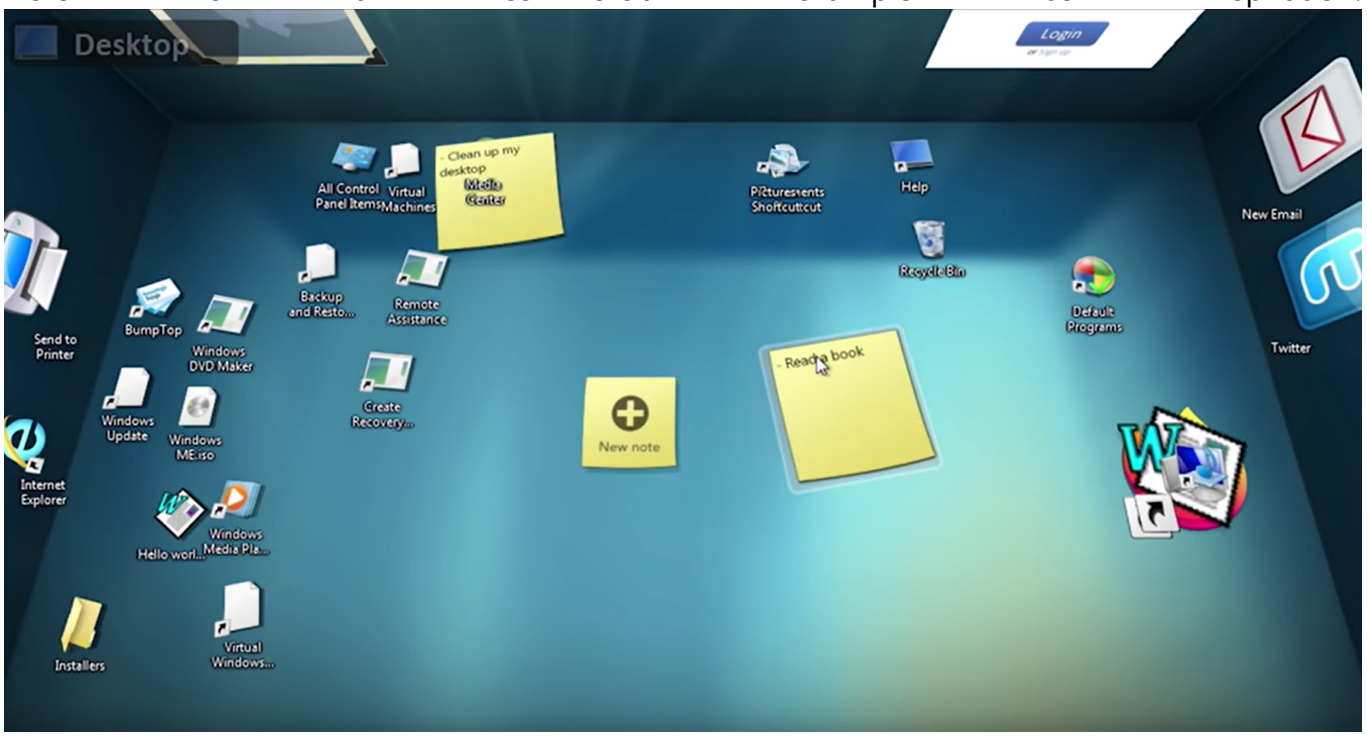
Code Interview

Each group member will be subject to an individual interview during the review. The one-on-one survey ensures that each group member has actively participated in the development of the project. The interview focuses on key aspects of the project implementation. Each group member must be able to navigate through the program code of the project and explain excerpts as well as associated theory on which the code is built. Performance in the individual interview will be scored with a factor on the group presentation of **0.4** to **1.1** in .05 steps.

Project description

The subject of the program to be implemented is a desktop user interface. The desktop consists of a background, taskbar and animated/interactive display windows. By mouse and keyboard it should be possible to use control elements of the taskbar and the application in the application windows. All parts of the user interface are composed of three-dimensional objects in a 3D scene using the framework build from the course.

Here is a commercial example as inspiration:



Bumptop 3D Desktop

Terminology

The requirments below use the following terms:

- **Drawing Area**: the drawing area describes the HTMLCanvas element on the project's webpage. This artboard serves as the target for the raytracer and rasterizer.
- **Application**: The application describes the set of implmented Typescript and WegGL functions that are usable from the project's web page.
- **Application window**: An application window is a set of objects within the interactive scene. This consists of the application pane itself and additional controls for minimizing, closing, and possibly maximizing the application pane. It has an associated icon in the taskbar.
- **Application pane**: The application pane is the area within the application window that is used to display the application's content.
- **Menu Bar**: The bar at the top of a display window that contains the buttons to open, minimize, and maximize the display window, as well as the title of the display window.
- **Background**: The element surrounding the desktop infront of which display windows are placed. Can be a pane, a plane, or a sphere (skybox).
- **Taskbar**: The bar at the bottom of the desktop that contains icons of the application windows as buttons to open, minimize, and maximize the application the corresponding Application Window.
- **Button**: A control that contains text or picture and triggers a specific behavior when clicked.
- **Desktop**: The area that contains the background, taskbar, and application windows.
- Elements not in the Drawing Area:
    - Slider: A slider is a control element that can be used to change a value by dragging a control element.
    - Color Picker: A color picker is a control element that can be used to select a color.
    - Load and Save Button: A button that can be used to load and save the scene.
    - Instructions: A text that describes the functionality of the application.
    - **Control Panel**: An element that is displayed adjacent to the drawing area with all and can be operated with the mouse. This includes buttons, sliders, text boxes, etc.
- Terminology from the lecture and exercise, like scene graph, visitor pattern, etc. are assumed to be known and do not need to be explained.

Minimum Requirements for passing the project

These requirements have to be at least implemented and functional to pass the project.

Deductions will be made for missing, or non-functional requirements.

1. [5 points] The application shall use a scene graph data structure to hierarchically manage the 3D objects of the scene. The contents of the displayed scene must be chosen in such a way that *the advantages of the data structure are apparent* (objects are moved relative to each other in ~at least~ three hierarchical levels). The depth of the scene graph must be changeable in the code (adding further (group) nodes must be possible in principle). Libraries like Three.js must **not** be used! For all lecture-relevant content, the implementation must be done by the group members. Functions of WebGL and Typescript may be used.
2. [10 points] Two renderers must be supported: A raytracer and a WebGL rasterizer. Both must traverse the scene graph using the Visitor pattern to render the scene. The raytracer is allowed to ignore all geometry objects except spheres when rendering. The active renderer must be switchable by keyboard input. Both renderers must support the Phong lighting model presented in the lecture. Switching between the renderers must be possible without restarting the application and preserve the lighting parameter.
3. [3 points] At least three different object types (e.g. cube, sphere, pyramid) must be included. These object types shall support colors defined per vertex. The implementation of the sphere known from the exercises expects only a single color. This may be adopted. All other visible objects must support multiple colors. **This also applies to optional objects such as mesh bodies.** The rule is that an **intentional, controllable and traceable modification** of a single object color must be necessary.
4. [8 points] At least 3 different animation nodes (Scaler - resizes along base axes, Mover - moves an object, Driver - moves along at least two axes in response to keyboard input) must be supported. At least two of the animation nodes must be controllable by the user in a suitable way. At least keyboard input must be supported to start and pause the animation. A constant speed can be assumed for displacements and rotations. The animations must be supported by both renderers.
5. [4 points] The 3D models displayed in the application should be able to be provided with a color texture. Furthermore, at least one of the objects must be provided with a color texture.
6. [5 points] The mathematical operations necessary for the scene graph must be implemented with the help of a math library. All necessary mathematical operations that are not supported by the programming language itself (i.e., without the use of additional libraries) must be implemented by the programmer. It is expected that formulas are not only copied but also understood.
7. [4 points] A shader program is to be written that implements the Phong lighting model. The parameters of the lighting model are to be modified by the Typescript application at runtime and via the user interface, e.g. the coefficients of the individual terms of the lighting model are to be modifiable at runtime of the program. At least one moving light source must be supported.
8. [2 points] At least two application windows, each with an application pane and menu bar. The pane should display at least one texture.
9. [4 points] A taskbar with buttons to show the individual application windows.
10. [5 points] Selection and manipulation of objects by mouse. The implementation must calculate a ray depending on the mouse position. The beam shall calculate the closest

object by intersection test. The manipulation can be either moving the windows with the mouse button pressed or minimizing.

Additional Requirements for a better grade

Additionally, there are points for the implementation of the following extensions (the number of points actually awarded depends on the degree of fulfillment of the respective task):

- [7 points] Materials for phong shading. The shader shall use the material for rendering, but not as color texture. (e.g., to implement bump, normal, or alpha mapping). Multiple primary textures (e.g. per side) do **not** fulfill this requirement! Instead, the intended use of the additional texture must be different from the primary texture.
- [3 points] Vidoes and text as texture. The text to be displayed has to unicode text in your code and rendered using WebGL. The source of the video can be a file or a URL. The video must be played in a loop.
- [8 points] Model Loader (e.g. OBJ format) to read in complex meshes. The source of the model can be a file, a URL, or statically copied into the code. The loader parses the mesh file into a mesh object that can be included as triangle mesh. If the models are self-selected, they must but be included in the repository. The loaded models must behave similar to the other geometry objects.
- [4 points] Multiple (>=3) moving light source nodes as part of the scene graph consisting of a point light source and a yellow sphere. The number of light sources must be variable over the execution time of the program but limited to a maximum of 8.
- [6 points] A magnifying glass effect at the position of the mouse, which can be switched by keystroke. Using a Zoom node is not sufficient. The distortion must be implemented as well.
- [4 points] Animation when clicking on an object within the scene: changing the texture, manipulation of the z value or own suggestion. A color/texture change on click is also sufficient manipulation. It must be possible that the object registering the click is not the object to which the animation is applied. For objects that are not spheres, it is **not** sufficient to test the cut test with the bounding-sphere. After that, the intersection test must be performed with the triangle mesh of the object itself.
- [8 points] Camera nodes as part of the scene graph (incl. animation, e.g. maximizing application windows via camera movement).
- [5 points] Acceleration of the beam shot by bounding spheres or bounding volume hirarchy. Instead of calculating an intersection test directly with the geometry, first test with a simple enclosing geometry.
- [8 points] The scene is to be loaded and saved locally on the computer as a JSON file. Thereby the path/file shall be selectable by the user. All modifications and parameters of the scene shall be included in the save. This includes any changes made to the scene by user interaction.
- [7 points] Ray tracing of all objects constructed from triangle meshes, including the lighting model. [+3 points] if shadows are implemented. [+4 points] if the fps is at least 10.
- [10 points] Shadow mapping in WebGL: for a single light source, objects cast a shadow on other. The scene has to be rendered from the perspective of the light source into a framebuffer and the distance to the light source is to be saved into a texture. Use a depth

test to determine the shadowed area.
- [10 points] Second, interactive scene as display content of the drawing area. The result of the render call of another scene is to be displayed on the drawing area. This scene is to be interacted with (6 points). If the drawing area is clicked, the object in the second scene must be determined and manipulated by beam intersection (4 points).
- [5 points] Simple application in the drawing area: a game like tic tac toe or a drawing application shall be able to be manipulated interactively on the drawing area.
- [up to 5 points] Own suggestion for an extension of the application. The suggestion must be approved by the lecturer before implementation. The number of points awarded depends on the complexity of the implementation as well as code quality and performance of the application..
- [up to 5 points] Exceptional implementation of requirements. The number of points awarded depends on the code quality and performance of the application.

Non Functional Requirements

These requirements lead only to deduction and only if not fulfilled sufficiently, i.e. hinder the evaluation of the functional requirements or the development and usage of the application.

Code Quality [-10 points]

- The code is structured in a meaningful way and is easy to understand.
- The code is commented and the comments are meaningful.
- The code is formatted in a uniform way.
- The code is easy to read and understand.
- The files are hirarchically structured.
- Variables and functions are named in a meaningful way.

Code Performance [DNF / -10 points]

- [Did Not Finish]
  - **The project must be executable on an updated system (Windows, macOS or Linux distribution) with current software or latest LTS variants. At least one configuration must be tested and must be documented with the commit.**.
  - The code is executable without modification.
  - Mandatory requirements are fulfilled to the extent written above.
- [-10 points]
  - The code is performant and does not contain unnecessary loops or calculations.
  - The code is optimized for the browser and does not contain unnecessary calculations.

Documentation [-5 points]

The documentation is complete and understandable. The documentation is written in English.

- Code is commented as necessary.

    - Methods use annotations to describe their purpose and parameters.
    - Variable names are meaningful, self-explanatory and consistent.
- A Readme.md file is included in the repository, which describes:
    - The project
    - The responsible persons for each part of the project
    - The build process with system requirements already tested
    - The authors
    - The used sources (if not already included in the code)
    - The used tools (if not already included in the code)
    - The used tutorials (if not already included in the code)
    - The used code snippets (if not already included in the code)
- Pictures, diagrams and videos are included to illustrate the project as needed.
    - You are allowed to use them during the code insterview. ## Version Control [-5 points]
- Expressive commit messages are used.
- .gitingore is used to exclude unnecessary files from the repository.
- The submission is tagged with the tag ''final'' and on the master branch.
    - "Tagging" refers to the Git functionality of the same name and is not to be equated with a specific naming of a Git commit description.

Build Process [-5 points]

- The build process is structured in a meaningful way. (e.g. build, run, clean)
    - Dont deviate from the given structure without a good reason.
    - Dont rely on the IDE to build the project.

Explanations

Requirements

If parameters of the lighting model in the shader are to be modifiable at runtime, it is expected that the parameters in the shader program are implemented as `uniform` and can be modified from TypeScript side. It is recommended that the HTML page allows modification of the parameters with sliders/buttons. Other, suitable, modifiers may be used.

Using multiple textures for another process does not mean simply adding, multiplying or similar the color information of two textures together. The task requires that an additional texture contains values that do not directly correspond to color values. These can be, for example, material coefficients of the Phong lighting model.

Nodes as a part of the scene graph mean that they are also used like a part of the scene graph. Inserting nodes into the data structure and then ignoring them there and using them as if they were completely independent of the scene graph does not fulfill the requirement. Nodes in a scene graph must pick up the transformations of the nodes preceding them in the hierarchy. It is further expected that all objects in a render run behave uniformly. Neither camera information nor lighting information should behave differently for different objects within the same render

run. Accordingly, when using light sources and camera in the scene graph, it is necessary to extract their information in a separate traversal of the scene graph before rendering all objects in another traversal.

Examples of effects for not meeting a requirement.

The minimum requirements must be implemented, used, and functioning at least to some extent. If parts of it are not present at all, or do not function at all, failure to pass follows.**.

Specific examples of point deductions:

- The rastervisitor does not render spheres correctly in the border area.
- Rayvisitor multiplies matrices incorrectly and all spheres are inside each other. (However, the scene graph is defined correctly).
- The implemented Phong lighting model renders only the ambient part correctly, because there is an error in the cross product(All parts of the model are however calculated)
- three animation nodes exist but the rotor makes the object jump wildly through the scene instead of rotating the object, despite the correct matrix.

Concrete examples for missing minimum requirements which make the project unsuccessful:

- only two animation nodes are implemented or used which change the scene.
- The minimum requirements are not transferred from the exercises to the project.
- A visitor generates a runtime error when traversing and application renders only the first frame correctly.
- No object with texture in the scene
- No three different object types used during rasterization
- The presence of a scene graph but not the consistent use of it (e.g. saving / using light sources outside the graph)
- The sum of points from minimum requirements and optional requirement falls below the minimum passing level.

27.07.2022

Marc Erich Latoschik, Matthias Popp, Christian Schell, Chair for Human-Computer Interaction