Project

Choose programming language of your liking (suggested Python/Java/Javascript)

Finished project send to ''dana.skorvankova@fmph.uniba.sk''

=== Stage 1 (3p) ===

''Deadline 11.10.2021 0:00''

Create a simple loader and visualizer of meshes, stored in obj. file format.

- Structures
    - implement IndexedFace which contains ''array of Vec4 vertex positions'' and ''array of Int indices''
    - Mesh import and display
    - load mesh from Wavefront file (.obj). (Download test models here). Assume that all imported models are scaled to box (2x2).
    - display wireframe model in the center of a screen so that the Y-axis of model directs upwards and X-axis directs right. (Z is ignored)

- Examples of the functional program (ignore gui):
    - screen 1
    - screen 2

=== Stage 2 (10p) ===

''Deadline 25.10.2021 0:00''

Add transformation controls to your tool.

- Structures
    - implement math types ''Mat4'' and ''Vec4''.
    - Add functions ''Multiply(Mat4, Mat4)'', ''Multiply(Mat4, Vec4)'' OR override the multiplication operator.
    - Transformations - must be implemented using matrix multiplication!

  - Add buttons that can control the transformations of the model. When a user requests transformation by clicking a button, a model matrix should be constructed. (Optionally added to the previous transformation)
  - Model matrix can be ''reset to identity'' using a Reset button
  - Construct a projection matrix which performs primitive orthogonal projection and transforms the object into the viewport (scale and translate as in the previous stage)
  - Always store the original model. Model transformation and projection is done for each point just before rendering

- Examples of the functional program:
  - screen 1 = Import only
  - screen 2 = Import only
  - screen 3 = Imported, translated by -0.6 in X-axis, rotated by 0.6 rad around Y-axis, rotated by 0.4 rad around Z-axis and scaled by factor 1.1, in order

=== Stage 3 (7p) ===

''Deadline 7.11.2021 23:59''

Enhance existing visualization tool by implementing Blinn-Phong Lightning Model

- Mesh display
  - Implement back-face culling. Faces on the far side of the mesh should not be rendered
  - Display solid model - each face is drawn as a polygon filled with color (no edges)
  - Final face color should be calculated as multiples of base color and intensity calculated by Blinn-Phong
  - Controls
  - Add an interface to control the direction or position of the incoming light

- Examples of the functional program:
  - screen 1
  - screen 2

=== Bonus Points ===

''Submit with Stage 3''

Extend your visualization options by the following features to earn bonus points. Attention: Your point total (including bonus points) will not exceed 25 points for exercises from computer graphics part.

- Material Properties (1p)
    - Expose material properties ka, kd, ks, shininess, and color to GUI. Re-render image when the user changes these properties.

- Lighting Model Selection (2p)
    - Implement a switch that allows the selection between two lighting models (Phong vs Blin-Phong). The difference should be minimal but it is a proof of concept.

- Light Types (3p)
    - Add possibility to select the light type (point light, sun/directional light) - the GUI option should also change from "light direction" to "light position".