Practical work

Summary

This work aims to apply the main concepts of 3D programming discussed in classes. Thus, students, organized into **groups of 4 elements**, must develop a program in C++ that, through the use of the OpenGL, GLM and GLFW libraries, performs the rendering of a 3D scenario, applying the respective textures, lighting, and performs an animation in response to a keyboard event. This work will culminate in the delivery, via Moodle, of the source code files, as well as an oral presentation/defense of the work.

Practical Work

The " **P3D-TP.zip** " file contains:

- Statement of work;
- 3D models of billiard balls (.obj and .mtl files).
- Texture images for the 15 balls (in .jpg format).

The work should be carried out in groups, and the **groups must be made up of 4 elements**.

The work must also be "defended" by the students, in an oral presentation to be carried out in class.

Goal

This work aims to develop a program in **C++** (respecting the Object Oriented Programming paradigm) that performs the rendering of a 3D scenario, applying the respective textures and lighting, and performs an animation (movement of one of the balls) in response to a keyboard event.

The requirements are grouped here in 4 steps, in order to guide students towards solving the work. Note, however, that **you should only submit a single version of your program (the final version), which meets all the requirements**.

- Step 1 :

- Implement window management and user interface through the GLFW library;
- Implement the manipulation of matrices and vectors through the GLM library;
- ([1]) Implement the rendering of a parallelepiped through the OpenGL library (including GLEW);
- ([2]) The coloring of the parallelepiped fragments must be carried out so that each face presents a distinct color (note that it is not expected that the lighting effects will be implemented at this stage);
- The application should allow the user to zoom through the *scroll* mouse *wheel;*
- The application should allow navigating (rotating) around the center of the parallelepiped, through mouse movements.

- Step 2 :

- Students should develop a C++ library (respecting the Object Oriented Programming paradigm) capable of:
  - Load vertex position data, normals, and texture coordinates from .obj;
  - Reorganize vertex information according to face information;
  - Send the vertex data to the GPU memory (VAO and respective VBO);

[1]It is not a requirement for evaluation. It serves only as a guide for the development of the work.

- Identify in file .obj the name of the file .mtl;
- Load material properties (.mtl file), including the texture image;
- Load the texture image into an OpenGL texture unit;
- Provide function for rendering the object.

Although the .mtl supports multiple materials, for this work assume that a .mtl will only have a single material to apply to the entire object.

- The application must make use of this C++ library to load the models, compose and render a 3D scenario that presents 15 balls arranged on a snooker table.

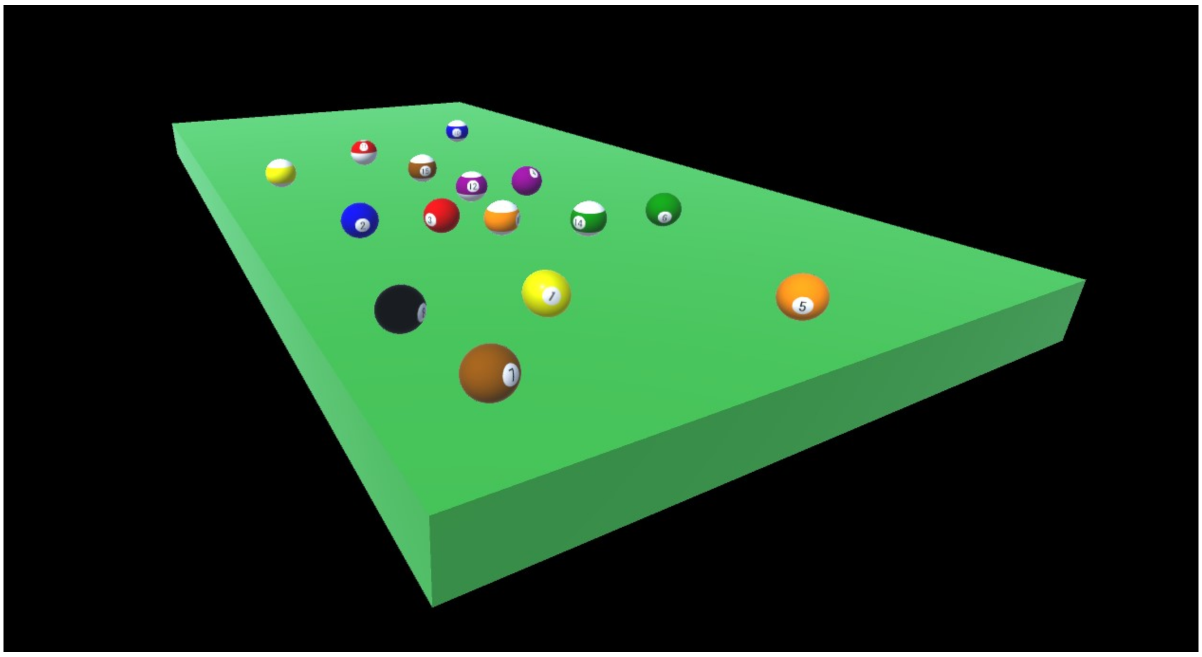

Figure 1: Example of composition with pool table and balls.

- The camera must remain motionless in a position that allows viewing the entire table, being oriented towards the center of the table.
- The library must have its own namespace in which a class (and its respective member functions) will be defined to manage the rendering process. The following member functions are mandatory:
  - **`Read(const std::string obj_model_filepath)`**
    - This function aims to load the data from the file into the CPU memory .obj, whose path is passed as an argument.
    - The materials and textures associated with the model must also be loaded into the CPU's memory.
  - **`Send(void)`**
    - Generates the necessary VAO and VBO and sends the model data (vertices, texture coordinates and normals) to GPU memory.
  - **`Draw(glm::vec3 position, glm::vec3 orientation)`**
    - Function to render the model in the position defined by **`position`** and orientation defined by **`orientation`**.
  - Functions should also be created that allow the association between model data (previously loaded to the GPU through the **`Send()`** `function`) and the attributes of a shader program.
  - In addition to these mandatory functions, others may be created as deemed necessary.

- Step 3 :

- It should be possible to apply any combination of 4 light sources that fall on the object:
  - Ambient light;
  - Directional light;
  - Spot light;
  - Conical light.
- The parameters of each of the light sources are at the discretion of each group;
- The application should allow the user to activate/deactivate each light source, using a key:
  - '1' – Enable/disable ambient light source;
  - '2' – Enable/disable directional light source;
  - '3' – Enable/disable point light source;
  - '4' – Enable/disable cone light source.

- Step 4 :

- Implement the motion animation (translation and rotation) of one of the billiard balls. This animation should be triggered as soon as the user presses the “Space” key. The animation should stop when the ball hits another pool ball or the edge of the table.

Format of 3D Models

The OBJ format allows representing 3D objects in text files that follow a certain structure. For this work, the 3D models of each ball are defined in two text files (.obj and .mtl) and a texture image (.jpg). Example:

• Ball1.obj	→Information regarding vertices, texture coordinates, normals and faces.
• Ball1.mtl	→Information related to the material.
• PoolBalluv1.jpg\	→3D model texture image.

The library should receive the path to the .obj and extract the information related to:

- File name with extension mtl
- Vertex position data (v) →Each line represents the coordinates in **x**, **y** and **z**.
- Texture coordinate data (vt)   Each line represents the coordinate in **s** and **t**.
- Normal data (vn) → In each line the values of the normal vector in **x**, **y** and **z**.
- faces (f)

With file name .mtl, the library must read the parameters that define the material:

- Ka – Coefficient of reflection of ambient light
- Kd - Coefficient of reflection of scattered light
- Ks - Coefficient of reflection of specular light
- Ns – Specular exponent (represents the brightness of the object. Same as *material shineness*.)

You should also read the filename of the texture image:

- map_kd

Ignore any other parameters that the file may contain.

For more information on the OBJ format, consult:

`https://en.wikipedia.org/wiki/Wavefront_.obj_file`

Assessment

The following factors will be taken as evaluation criteria:

- Respect for the rules of delivery of work;
- Program quality:
    - organization, clarity and quality of the source code;
    - use of the C++ language and respect for the OOP paradigm;
    - development of the functionalities described in the work statement;
    - level of optimization of implemented functionalities;
    - correct functioning of the program;
    - added value [2].
- Quality of the code and respective comments, as well as the oral presentation:
    - correct and complete description of the structure of the program;
    - description of the techniques applied in the development of the functionalities.
- Knowledge that each student demonstrates regarding the presented code.

The collective nature of carrying out a group work does not affect the fact that the evaluation is individual for each of the elements of the group.

Deadlines

Carrying out the work presupposes the delivery of a ZIP file containing the files with the source code.

The work must be sent to the professor, via Moodle, by the defined date and time (also available on the Moodle page of the UC). The teacher reserves the right not to evaluate the work delivered after that date and time.

The delivery of the practical work must comply with **the** following requirements:

- The files with the source code (**do not** include the Visual Studio project or the executable) must be placed in a **ZIP file** with the name "**P3D-TP-xxxx-xxxx-xxxx-xxxx.zip**" (where **xxxx** must be filled in with the number of students of each of the elements of the group) and sent to the teacher through the Moodle platform.
- Only 1 (one) member of each group must submit the work.

The delivery period ends on the **9th of June** at **23:00** . **Works submitted after this date will not be considered** . The **Practical Work** will be defended during the classes on **June 13th and 15th**.

Ethical Conduct

The lack of transparency in assessments, whether face-to-face or not, is naturally illegal and immoral. All sources used to support the work must be mandatorily and clearly referenced. Any plagiarism, copying or improper academic conduct will be penalized with the annulment of the work. If the existence of notoriously similar works between groups is verified, all similar works will be annulled.

[2]Added value means how the work stands out (positively) from the rest.