

Final project.

Martín Alberto Fernández Lorenzo

A01270963

Computer Graphics

TC 3022

12/08/2017

**Research**

Open Graphics Library (OpenGL) provides the programmer with an interface to graphics hardware. It is a low-level rendering library, available on major platforms. It is used in any graphics application. OpenGL provides low-level routines, intentionally for control and flexibility. Most distributions include the OpenGL Utility Library (GLU), a high-level rendering and modeling library. At the core of OpenGL is the rendering pipeline. Most of the steps are handled by OpenGL. It does not provide any means for windowing, menu or input. The OpenGL Utility Toolkit (GLUT), is a set of support libraries for this purposes [1]. OpenGL just provides rendering capabilities, thus animation is not possible.

The Computer Graphics Rendering Pipeline is the sequence of steps that a graphics library takes when rendering objects [2]. OpenGL works in the following order.

1. Prepares vertex array data, and renders it.
2. Each vertex is processed by a shader. Optional tessellation (vertex are subdivided into primitives)[3] or geometry shading (an input primitive is outputted as zero or more primitives)[4] may happen.
3. Post-processing of the vertex. Output is adjusted.
4. Primitive Assembly. Primitives are divided into base primitives and passed along to the rasterizer to be rendered [5].
5. Interpolation of primitive parameters and scan conversion, this is called rasterization. Generates fragments, a discrete element broken down from a Primitive.
6. Each fragment is processed by a shader. This will produce a set of colors and a depth value.
7. Per-sample processing. Data is written to buffers.



In short, a vertex has different attributes (color, depth, texture, etc.) This vertices may be transformed or lit. Primitives are formed by vertices. Then they are rasterized into fragments. Each fragment is processed using, its attributes (color, depth, texture), texture data/image data, tests are made to ensure fragment correctness, if all tests are passed, the pixel data is replaced in the frame buffer.

Concepts:

Primitive. The result of the interpretation of a vertex stream. Sometimes called base primitive [6].

Vertex Stream. A list of ordered vertices [6].

Rasterization. 3d objects seen from a 2d perspective [7].

**Development plan**

* Chess:
  + Game modes: PvP, PvAI
  + [Chess variants](https://www.wikiwand.com/en/List_of_chess_variants)

Regardless of the project, work will be subdivided into the following categories:

* Assets
  + Environment
  + Characters/Objects
  + Lightning/Texturing
* Gameplay/Interaction
* GUI (Splash screen, start page, settings) (Not achieved.)
* Sound (Not achieved.)

Working alone I can work at my own pace, this also means that there is more for me to do. None of the possible projects is technically demanding. But, will be long projects. I expect the assets modeling or acquiring to be fairly simple. GUI won’t be much of an issue due to not a lot of functionality. Gameplay or interaction will be the most demanding, taking the majority of the time. The chess project will be demanding due to the AI implementation and the several, if convenient, game variants to be added. For the visualizer I would expect to be able to upload an mp3 file and be able to play it. Additional functionality such as youtube playback may be considered, if API compatibility proves to be not very complicated. For the final game, it’s based on a previous unfinished project. Implementation and design would start anew. Would be 3d-based, survival style with waves of enemies coming until the death of the player.

For the second advance, a technical demo should be functional. Placeholder assets may be used. Functionality would not be implemented fully. But should be enough for a proof of concept.

Final delivery:

* Chess board textured and rendered.
* Pieces rendered.
* Complete layout of pieces based on game.
* Pieces are able to make valid moves on turns.
* Camera able to move around and zoom in and out.

Uncompleted goals:

* GUI was not implemented.
* Full gameplay (check, checkmate, stalemate, castling, promotion) was not implemented.
* Sound was not added to the game, as it was not part of the course.
* Additional game modes were not added.
* AI was not implemented.

As a conclusion, chess proved to be a difficult challenge; regarding graphics and rules. Most of what was not implemented was regarding to the rules of chess. Graphically the game was completed, besides the addition of a main menu. Future development of the project, may be interesting as a side project. Currently several chess engines exist, so it may be easier to implement one of those instead of trying to implement everything from the ground up. Time constraints were a challenge to the main development besides external issues that occurred during the semester.

Sources

1. Astle, Dave, and Hawkins, Kevin. Beginning OpenGL Game Programming. Boston: Course PTR, 2004. Accessed September 11, 2017. ProQuest Ebook Central.
2. The Khronos Group. (2017, September 12). Retrieved September 12, 2017, from <https://www.khronos.org/opengl/wiki/Rendering_Pipeline_Overview>
3. The Khronos Group. (2017, September 12). Retrieved September 12, 2017, from [https://www.khronos.org/opengl/wiki/](https://www.khronos.org/opengl/wiki/Rendering_Pipeline_Overview)[Vertex\_Processing](https://www.khronos.org/opengl/wiki/Vertex_Processing)
4. The Khronos Group. (2017, September 12). Retrieved September 12, 2017, from [https://www.khronos.org/opengl/wiki/](https://www.khronos.org/opengl/wiki/Rendering_Pipeline_Overview)[Geometry\_Shader](https://www.khronos.org/opengl/wiki/Geometry_Shader)
5. The Khronos Group. (2017, September 12). Retrieved September 12, 2017, from [https://www.khronos.org/opengl/wiki/](https://www.khronos.org/opengl/wiki/Rendering_Pipeline_Overview)[Primitive\_Assembly](https://www.khronos.org/opengl/wiki/Primitive_Assembly)
6. The Khronos Group. (2017, September 12). Retrieved September 12, 2017, from [https://www.khronos.org/opengl/wiki/](https://www.khronos.org/opengl/wiki/Rendering_Pipeline_Overview)[Primitive#Vertex\_stream](https://www.khronos.org/opengl/wiki/Primitive#Vertex_stream)
7. James D. Foley et al., Computer Graphics: Principles and Practice, 2nd ed. (1990, reprinted 1997; originally published as Fundamentals of Interactive Computer Graphics, 1982)
8. Davis, M. J. (Ed.). (2011). Computer graphics. Retrieved from <http://0-ebookcentral.proquest.com.millenium.itesm.mx>