Manual técnico

Developer's Guide

Proyecto Computación Gráfica

Elaboró: Saúl Abraham Esparza Rivera
Fecha: 12/05/2022

Índice / Index

Introducción

El presente documento tiene la intención de mostrar a grande escala la experiencia que se vivió al momento de desarrollar el presente proyecto. Desde los momentos en que todo salía bien hasta aquellas fechas en las que parecía que nada quedaba bien. Revisaremos algunas secciones críticas del código usado en el entregable y podremos entender mejor qué se logró hacer y qué no pude ser plasmado, sin más, comencemos

Objetivos

El proyecto tenía como objetivo llevar a OpenGL un edificio de nuestra elección, teniendo en cuenta la mayor cantidad de elementos presentes en la referencia y pensando en que lo que sea que planeáramos poner en el tenía que se posible dadas las capacidades y limitaciones de OpenGL y del tiempo y habilidad con los cuales contábamos para desarrollarlo.

La idea era desarrollar el escenario con el paso de las prácticas de laboratorio, teniendo en cuenta los avances y aplicando lo aprendido para poder ir avanzando de manera eficiente. Esto permitiría a nosotros los desarrolladores tener mejor control sobre los avances y a poder proceder de la manera que mejor viéramos posible.

El escenario debía contar con elementos que en el momento de la definición del proyecto aun o habíamos visto, por lo que la promesa de avanzar a nuestro ritmo iba ligada al avance de laboratorio y a todas esas variables que permutan nuestra vida.

En este documento están plasmadas las experiencias de desarrollo que permitirán conocer mejor como fue el mismo y cómo fue que llegamos hasta aquí.

A continuación presento un diagrama de Gantt a modo de resumen del desarrollo del proyecto, las etapas del mismo y los periodos que costó llevar acabo (o no) una actividad.

Proyecto computación gráfica

Periodos son semanas, inicié las actividades desde el 13 de marzo y el deadline es el día 12 de mayo.

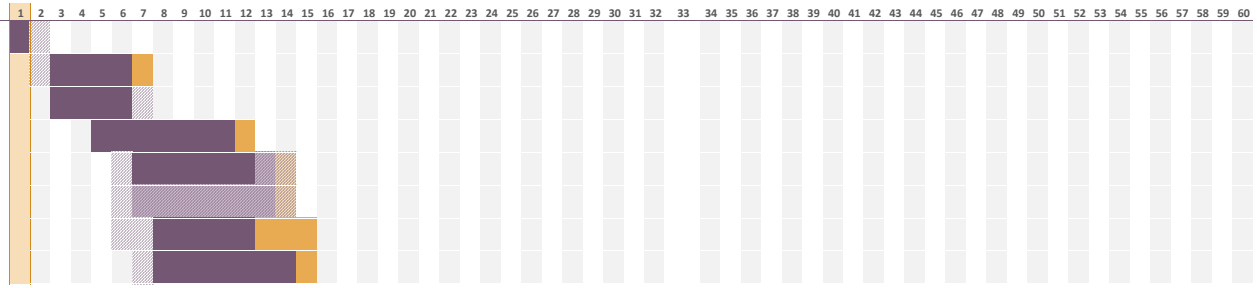
Período resaltado: 1

Duración del plan | Inicio real | % Completado | Real (fuera del plan) | % Completado (fuera del plan)

ACTIVIDAD	INICIO DEL PLAN	DURACIÓN DEL PLAN	INICIO REAL	DURACIÓN REAL	PORCENTAJE COMPLETADO
Imagen de referencia	1	2	1	1	100%
Preparación de objetos	2	5	3	5	100%
Preparación del codigo base	3	5	3	4	100%
Depuración de objetos	5	7	5	8	100%
Animaciones	6	8	7	8	75%
Iluminación	6	8	7	8	0%
Redacción de manuales	6	7	8	8	100%
Preparación del producto final	7	8	8	8	100%

PERÍODOS

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60



Limitaciones

Si bien ya antes había trabajado en un semestre pasado con OpenGL, todo el desarrollo fue totalmente distinto a aquella interacción. Por un lado, teníamos que los temas, tanto de teoría como de laboratorio se desarrollaban de manera diferente a como fue en su momento para mi y por el otro que ya conocía de lo que era capaz OpenGL y a pesar de ser un framework bastante competente, también actuaba como uno bastante molesto y limitante, por ejemplo, el manejo de la memoria al momento de cargar un elemento suele ser algo volátil, causando excepciones bastantes molestas y algunas veces sin explicación alguna.

También está la complejidad que suele tomar al momento de manejar cabeceras, y al estar desarrollando enteramente en código, siempre era complicado revisar los cambios a un objeto, dado que teníamos que recompilar y ejecutar cada vez, la experiencia se volvió bastante bizarra al usar herramientas mucho más avanzadas, en mi caso el software 3DS Max me resulta super sencillo de manejar, al menos al nivel que manejo. Se siente como usar un programa pensado para humanos, no digo que OpenGL sea anticuado, pero lo es.

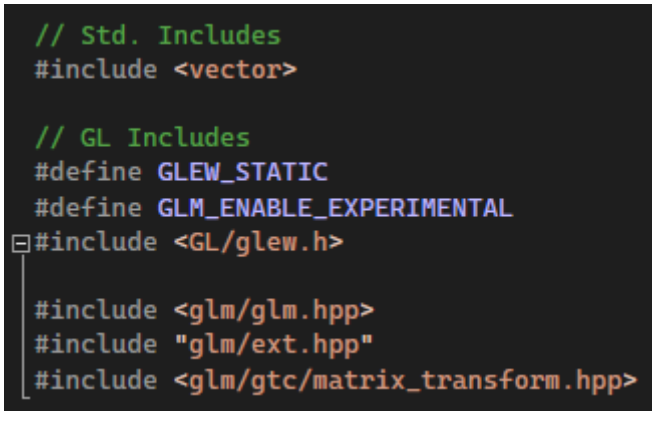
Estas limitantes llevaban a consumir mucho tiempo cargando elementos, realizando transformaciones sobre el código e ir compilando entre cada intento. Si a algo le he de achacar la tardanza que se presentó en este proyecto fue a OpenGL y sus metodologías de desarrollo del siglo pasado.

Pero bueno más allá de eso, la única gran limitante fue la escasa cantidad de tiempo libre, al trabajar y llevar otras materias, el tiempo se vuelve sumamente escaso. Por suerte creo que entregaré este proyecto a tiempo.

Un vistazo al código

Reutilicé gran parte del código de una de las prácticas que desarrollé en el laboratorio, haciendo adecuaciones y quitando todo aquello que no tenía utilidad para mi proyecto. La carga de modelos, la aplicación de transformaciones y de secuencias de movimiento predeterminadas son cosas que vimos durante todo el semestre y por suerte aquí me sirvieron, aunque eso no significa que no tuve que investigar por mi cuenta. A continuación, revisaré solo algunas instancias del código que considero dignas de mostrar, dado que el resto de mi código es bastante estándar.

Algo difícil de investigar soluciones para OpenGL es que cada quien tiene su propio código, osea, cada quien importa modelos de X o Y forma, cada quien aplica transformaciones de manera distinta. Rara vez encontré algo de utilidad en ese mar de amargura llamado StackOverflow, aunque las veces que sí son de destacarse, por ejemplo, tenemos el caso de obtener en consola la posición de la cámara, técnica que tenía planeado usar para colocar elementos mucho más rápido. Leyendo soluciones en StackOverflow encontré como activar una función “experimental” de OpenGL con la que pude ver la posición actual de la cámara y subsecuentemente usar esa posición para colocar mis objetos de manera más rápida.



Lo primero que había que hacer era definir la función experimental y posteriormente incluir el archivo ext.hpp dentro de camera.h. Una vez hecho esto bastaba con colocar la siguiente línea en alguna parte del código.

```
glm::mat4 GetViewMatrix()
{
    //Imprimir la posición de la cámara
    //std::cout << "La posicion actual de la camara es: " << glm::to_string(position) << "\n" << std::endl; //F por los acentos

    return glm::lookAt(this->position, this->position + this->front, this->up);
}
```

Con esto la consola mostraba la posición de la cámara, lo que me facilitó sobre manera mi trabajo.

Otro conocimiento adquirido desde la internet fue la manera de poner un skybox diferente al que nos proporcionó el profesor, hubo muchos intentos, encontré páginas que generaban “cube maps” pero ninguno se veía en OpenGL, hasta que encontré una página que nos e vía muy reciente, aun así explicaban el proceso y al final logré poner mi propio skybox, incluso descubrí algunas características algo raras, como que el skybox anterior estaba girado en dos caras, entonces tuve que desactivar la línea que realizaba esa inversión.

```
GLuint texture;
glGenTextures(1, &texture);
glBindTexture(GL_TEXTURE_2D, texture);
int textureWidth, textureHeight, nrChannels;
//stbi_set_flip_vertically_on_load(true);
unsigned char* image;
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST_MIPMAP_NEAREST);
```

La línea comentada era la culpable.

Estas experiencias de tener que investigar acerca de un problema que me apareció y lograr encontrar una solución entre una montaña de personas que poco o nada aportan fue muy refrescante.

Conclusión

Personalmente me divertí mucho este semestre realizando todas estas prácticas y descubrí algo bastante tétrico, por decir algo. Me refiero a la importancia del manejo del tiempo y a las repercusiones de quedarse sin él. Mi proyecto carece de muchas características que me habrían encantado integrar, pero por X o Y motivo no pude, algunas veces fue por intervenciones de otras materias y sus eternos proyectos los que me evitaron de poder avanzar aquí a mi paso. Otras veces fue mi trabajo y también fue mi culpa totalmente por distraerme tanto y aplazar las cosas.

Una repercusión que tengo muy presente es la incapacidad de incorporar luces a mi escenario, todo porque esa práctica no la pude realizar en su momento, a pesar de intentar realizarla fuera de tiempo, me di cuenta que no es tan fácil recuperar el tiempo perdido y hoy lo viví.

Me llevó muchos aprendizajes de la materia, tanto del temario como fuera de este. Fue un semestre bastante pesado, pero igualmente muy educativo y agradable.

Introduction

This document intends to show, on a large scale, the experience that was lived at the time of developing this project. From the moments when everything went nice and easy to those times when it seemed like nothing was going well. We will review some critical sections of the code used in the deliverable and we will be able to better understand what was achieved and what could not be done, without further ado, let's start

Objectives

The project aimed to bring a building of our choice to OpenGL, considering the greater number of elements present in the reference and thinking that whatever we planned to put in it had to be possible given the capabilities and limitations of OpenGL and the time and skill we had to develop it.

The idea was to develop the scenario with the passage of laboratory practices, considering the progress and applying what was learned in order to move forward efficiently. This would allow us developers to have better control over the progress and to be able to proceed in the way that we see best as possible.

The scenario had to have elements that at the time of defining the project we had not yet seen, so the promise of advancing at our own pace was linked to the progress of the laboratory and all those variables that change our lives.

This document contains the development experiences that will allow us to better understand what it was like and how we got here.

Below I present a Gantt chart as a summary of the development of the project, its stages and the periods it took to carry out (or not) an activity.

Proyecto computación gráfica

Periodos son semanas, inicié las actividades desde el 13 de marzo y el deadline es el día 12 de mayo.

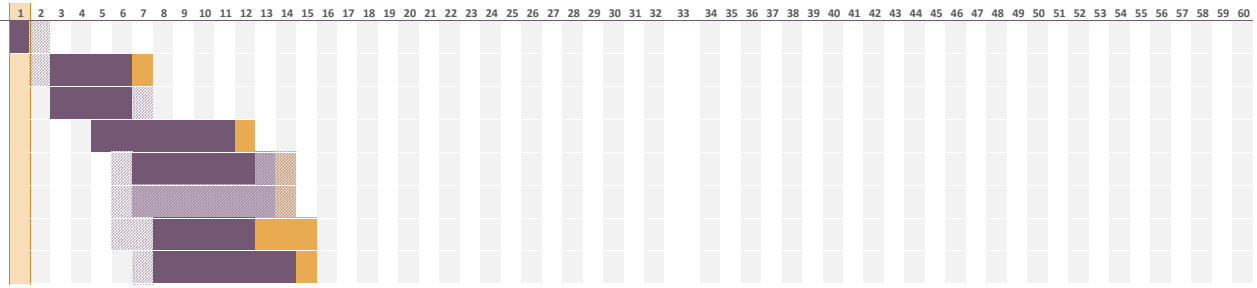
Período resaltado: 1

Duración del plan | Inicio real | % Completado | Real (fuera del plan) | % Completado (fuera del plan)

ACTIVIDAD	INICIO DEL PLAN	DURACIÓN DEL PLAN	INICIO REAL	DURACIÓN REAL	PORCENTAJE COMPLETADO
Imagen de referencia	1	2	1	1	100%
Preparación de objetos	2	5	3	5	100%
Preparación del codigo base	3	5	3	4	100%
Depuración de objetos	5	7	5	8	100%
Animaciones	6	8	7	8	75%
Iluminación	6	8	7	8	0%
Redacción de manuales	6	7	8	8	100%
Preparación del producto final	7	8	8	8	100%

PERÍODOS

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60



Limitations

Although I had already worked with OpenGL for a past semester, the whole development was totally different from that interaction. On the one hand, we had that the topics, both theory and laboratory, were developed in a different way than it was at the time for me and on the other hand, that I already knew what OpenGL was capable of and despite being a fairly competent framework, it also acted as a quite annoying and limiting one, for example, the memory handling when loading an element is usually somewhat volatile, causing quite annoying and sometimes unexplained exceptions.

There is also the complexity that it usually takes when managing headers, and since we were developing entirely in code, it was always difficult to review the changes to an object, since we had to recompile and execute each time, the experience became quite bizarre when using tools much more advanced, in my case the 3DS Max software is super easy to handle, at least at the level I handle it. It feels like using a program made for humans, I'm not saying OpenGL is old fashioned, but it is.

These limitations led to consuming a lot of time loading elements, performing transformations on the code, and compiling between each attempt. If I must blame anything for the delay that occurred in this project, it was OpenGL and its development methodologies of the last century.

But well beyond that, the only major limitation was the limited amount of free time, when working and taking other subjects, time becomes extremely scarce. Luckily, I think I will deliver this project on time.

A look into the code

I reused a large amount of code from one of the practices that I developed in the laboratory, making adaptations, and removing everything that was not useful for my project. Loading models, applying transformations, and applying predetermined motion sequences are things we covered throughout the semester and luckily, they worked for me here, though that doesn't mean I didn't have to do some research on my own. Below I will review only a few instances of the code that I feel are worth showing, since the rest of my code is fairly standard.

Something difficult that occurs while investigating for solutions for OpenGL is that everyone has their own code, that is, everyone imports models of X or Y shape, everyone applies transformations in a different way. I rarely found anything useful in that sea of bitterness called StackOverflow, although the times that did stand out, for example, we have the case of obtaining the position of the camera in the console, a technique that I had planned to use to place elements much faster. Reading solutions on StackOverflow I found how to activate an "experimental" OpenGL function with which I could see the current position of the camera and subsequently use that position to place my objects faster.

```
// Std. Includes
#include <vector>

// GL Includes
#define GLEW_STATIC
#define GLM_ENABLE_EXPERIMENTAL
#include <GL/glew.h>

#include <glm/glm.hpp>
#include "glm/ext.hpp"
#include <glm/gtc/matrix_transform.hpp>
```

The first thing to do was define the experimental function and then include the ext.hpp file inside camera.h. Once this was done, it was enough to place the following line somewhere in the code.

```
glm::mat4 GetViewMatrix()
{
    //Imprimir la posición de la cámara
    //std::cout << "La posicion actual de la camara es: " << glm::to_string(position) << "\n" << std::endl; //F por los acentos

    return glm::lookAt(this->position, this->position + this->front, this->up);
}
```

With this, the console showed the position of the camera, which greatly facilitated my work. Although it was kind of messy because the function was called almost every second and the console was flooded with messages in a matter of seconds

Another knowledge acquired from the internet was how to put a skybox different from the one provided by the teacher, there were many attempts, I found pages that generated "cube maps" but none was seen in OpenGL, until I found a page that sent us very recently, even so they explained the process and in the end I managed to put my own skybox, I even discovered some strange features, like that the previous skybox was turned on two sides, so I had to disable the line that made that inversion.

```
GLuint texture;
glGenTextures(1, &texture);
glBindTexture(GL_TEXTURE_2D, texture);
int textureWidth, textureHeight, nrChannels;
//stbi_set_flip_vertically_on_load(true);
unsigned char* image;
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST_MIPMAP_NEAREST);
```

The commented line was the main suspect.

These experiences of having to investigate a problem that appeared to me and being able to find a solution among a mountain of people who contribute little, or nothing was very refreshing.