Trabajo Final Computación Gráfica

Autor: Dante Moreno, Natalia Maury, Joaquín Galván

***Abstract*— Informe del trabajo final de computación gráfica, que busca implementar los conceptos aprendidos durante el ciclo.**

I. Introducción

Durante el ciclo académico, se ha aprendido cómo usar OpenGL con el lenguaje de programación de c++ para aplicar los conocimientos adquiridos en el curso como iluminación, texturas, curvas de bezier, etc. Con ello se pueden realizar animaciones 3D que tienen diversas aplicaciones como juegos, simulaciones, etc. Por ello, en este trabajo final se realizará un juego utilizando openGL y aplicando los conceptos aprendidos en clase.

Debido a que todos los estudiantes que conforman este grupo pertenecen a la carrera de ciencias de la computación, se tiene el mismo objetivo al realizar este proyecto: la capacidad de aplicar el diseño de ingeniería para producir soluciones que satisfagan necesidades específicas con consideración de salud pública, seguridad y bienestar, así como factores globales, culturales, sociales, ambientales y económicos.

En este trabajo, realizaremos un juego implementando todos los conceptos aprendidos durante el ciclo para demostrar lo aprendido. El juego consistirá en (juego). Este juego permitirá que el usuario pueda interactuar utilizando el teclado de la computadora, de esta manera podrá ver los efectos de la cámara, las diferentes fuentes de luz que habrá, junto con sus reflejos y otros efectos como rotación, traslación, etc. Además, se aplicará la iluminación a objetos con texturas conforme el personaje se acerque a ellos para dar un efecto más realista. Las curvas de Bézier serán aplicadas con la trayectoria del carrito y su movimiento, que serán controladas por el personaje.

II. Diseño e implementación

Para poder hacer el juego, se utilizaron el siguiente diseño y componentes:

A. OpenGL

Se utilizó la API OpenGL como base para generar nuestras estructuras 3d y procesar la información necesaria para la iluminación, texturas y cámara. Gracias al acceso a la GPU que provee OpenGL, los shaders serán procesados mucho más rápido que otras secciones del código, según cómo esten distribuidas las instrucciones. Esto da una gran flexibilidad en cuanto a la programación de nuestro juego, permitiéndonos hacer los ajustes que queremos para las físicas del sistema al igual que nos da la posibilidad de modificar en tiempo real data de nuestros objetos 3d en base a funciones ya definidas, mediante el uso de variables tipo uniform en los shaders.

B. Programación orientada a objetos

Cuando se trabaja con OpenGL podemos ver que los procesos necesarios para mostrar objetos en la pantalla son increíblemente repetitivos, lo cual haría nuestro código increíblemente largo. Para ello, siguiendo el ejemplo del profesor del curso, decidimos tomar todo lo enseñado en el curso sobre OpengGl, y lo que se enseña en su pagina, y los dividimos en clases que nos permitirían una utilización mucho más ágil. Esto procedimiento conlleva los conceptos aprendidos previamente de encapsulamiento, abstracción, herencia y polimorfismo.

Las clases mas importantes son:

- **CGrafico:** Representa un objeto del escenario. Cuenta con relaciones (punteros) a las direcciones de sus shaders, textura, e incluso su modelado.
- **ProgramShaders:** Clase creada para facilitar el manejo de shaders. En su constructor lee los archivos de código del vertex y fragment shader y notifica si ocurre un error. Cuenta con métodos para usar el shader o verificar errores.

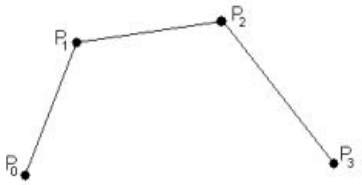

Fig. 1. Puntos de control de ejemplo conectados[5]

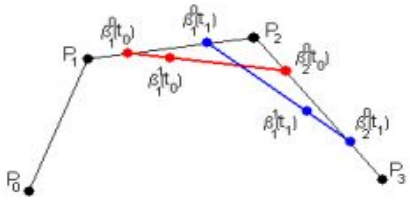

Fig. 2. Conexiones convexas de los puntos de control [5]

- **Controladora:** Esta clase maneja a todo el sistema, incluyendo la ventana en la que se renderizan las imágenes.

III. Estado del Arte

A. Curvas de Bézier

Creadas originalmente por el ingeniero Pierre E. Bézier para la industria automovilística, las curvas de Bézier se implementan actualmente también en el diseño, animación, entre otros. Trascendieron su entorno original gracias a la facilidad en su dibujo, ya que estas curvas están definidas por los *puntos de control*. Con el algoritmo de Casteljau se entiende la idea de los puntos de control. Este permite conocer las combinaciones convexas entre un conjunto de puntos. Una curva de Bézier se dibuja entonces como un punto que sigue la última recta entre puntos que recorren estas conexiones. (Ver Fig. 2)

Este algoritmo resulta en un procedimiento largo, definiendo cada recta entre recta, de grado enésimo. Por eso, las curvas de Bézier se pueden expresar usando el polinomio de Bernstein:

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} = \frac{n!}{i!(n-i)!}(1-t)^{n-i} \tag{1}$$

La fórmula obtenida para dibujar la curva de *n* puntos de control sería:

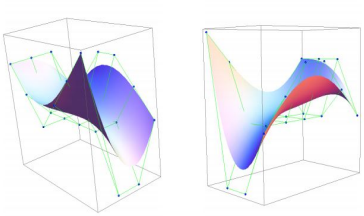

Fig. 3. Superficie de Bézier [5]

$$\alpha(t) = \sum_{i=0}^{n} B_i^n(t) P_i, t \in [0,1] \tag{2}$$

Donde P_i es el i-ésimo punto de control, que permite hallar el punto $\alpha(t)$,en un tiempo t de su recorrido, que dibuja la curva final. Este concepto puede ser aplicado también al ambiente tridimensional. Las superficies de Bézier trabajan con una red de control en lugar de puntos en un plano. Esta red está definida como $P_{i,j}{}_{(i,j)}^{(m,n)}$

$$x(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} B_i^m(u) B_j^n(v) P_{ij}, u \in [0,1]\times[1,0] \tag{3}$$

Con la aplicación de las superficies de Bézier se pueden obtener resultados como los de la Figura 3.

B. Iluminación

En la computación gráfica, la iluminación es otro de los conceptos esenciales para conseguir la fidelidad deseada en la imagen. Para facilitar el trabajo de simular la luz, esta se clasifica según su fuente y dirección. Las más importantes formas de luz son:

- **Luz de ambiente:** No tiene un origen ni dirección específico. Sirve para poder observar objetos aunque no existan fuentes de luz cerca, evitando la oscuridad absoluta. Estrictamente, está modelada por la superficie de los objetos, ya que refleja luz indirecta y determina la intensidad de la luz ambiente.
- **Luz direccional:** No tiene un punto de origen pero sí dirección. Corresponde a una fuente de luz lejana, como el sol o una bengala en el cielo nocturno. Esta fuente de luz existe y se toma en cuenta pero no es parte del escenario simulado.

- **Luz puntual:** Tiene una fuente de luz y su dirección es radial desde este punto de origen. Es parte del escenario, y la dirección de la luz varía dependiendo del objeto iluminado. Puede representar un foco o la llama de una vela.
- **Reflexión especular:** Es la reflexión que ocurre en superficies irregulares o difusas. Esta luz simula el punto brillante de una luz que aparece en objetos brillantes. Utiliza la ley de snell para calcular la luz especular en el objeto que recibe el rayo y aquí se genera un ángulo en el cual se refleja la luz.
- **Reflexión difusa:** Es la reflexión que ocurre en superficies planas. Esto hace que cuando se refleja un rayo de luz en una superficie, este rayo incidente es reflejado en muchas direcciones formando muchos ángulos en vez de uno. Esto se debe por la rugosidad de la superficie.

A la vez, cada objeto refleja luz según su material y la forma de su superficie. La reflexión especular de la luz, que considera a la superficie lisa idealmente, está determinada por la Ley de Snell. Esta ley determina el ángulo del rayo reflactado por el objeto iluminado que afecta a los destellos que se pueden ver. Sin embargo, las superficies normalmente son rugosas, aunque no sea perceptible al ojo humano. La reflexión difusa dicta que un rayo de luz entrante será reflejado en múltiples direcciones desde la superficie. En un reflector ideal, la ley del coseno de Lambert determina claramente la relación proporcional entre la distancia del reflector con la superficie, el ángulo desde su vector normal, y la intensidad de la luz. En términos simples, la luz se distribuye, disminuyendo su intensidad si cubre una mayor área. Trasladar la física de la luz a un modelo computacional que pueda realizar cálculos rápidos resulta difícil. Por eso, una aproximación aceptada ampliamente es el modelo de Phong, con el que se obtiene la intensidad de la luz por reflexión especular.

$$I_{especular} = k_s I_{luz} \cos \phi^s \tag{4}$$

Donde k_s es el coeficiente según el material, igual que s ajusta la luz. La iluminación de Phong se adapta para juntar los múltiples tipos de iluminación en una ecuación:

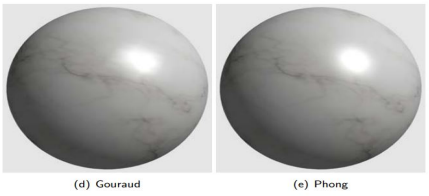

Fig. 4. Comparasión de resultados del Modelo de Phong y el de Gouraud[6]

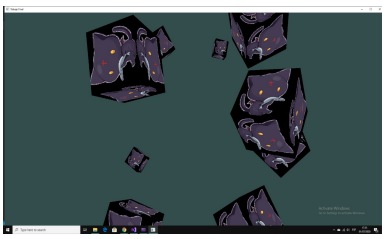

Fig. 5. Renderización de cubos con profundidad

$$I = I_a k_a + \sum_{i=1}^{m} f_{att}(i) I_l uz(i) [k_d (N.L) + k_s (R(i).V)^s] \tag{5}$$

Para un sistema con m luces, donde cada k es un coeficiente según el material y tipo de luz.

IV. Metodología

Los primeros resultados de renderizar estos cubos se muestra en la Figura 5.

V. Resultados

El proceso de implementación fue el siguiente: 1. Definir las fuentes de luz, elementos del proyecto, materiales y texturas. 2. Crear la plantilla a usar del proyecto. 3. Crear la clase del objeto 4. Crear la parte física del objeto 5. Crear la función de curva de Bézier 6. Crear los objetos con texturas y la iluminación 7. Crear la clase controladora con todos sus elementos.

Al crear el proyecto se tuvo diversos problemas y no todo funcionó como se esperaba. El proyecto tiene potencial y con mayor tiempo podría verse. No obstante con el tiempo obtenido este fue el resultado: Se lograrón crear los objetos pero al hacer las luces hubo complicaciones. No se pudo implementar las texturas debido a complicaciones

que hubo. La función de bézier fue difícil de programar pero el rol que cumpliría en el juego siempre fue claro y por qué era importante este elemento. Por estas razones es que el proyecto tiene potencial pero no se pudo terminar completamente por falta de tiempo y se necesita más practica con OpenGL.

VI. Conclusiones

Aunque el proyecto no haya funcionado, se logró entender cuál es el potencial de la computación gráfica y sus aplicaciones. Sí se tenia claro el concepto de qué iba a ser qué elemento y cuál era la función que iba a desempeñar. La curva de Bézier iba a ser utilizada para crear el recorrido que iba a seguir el carrito de mina. Las primitivas gráficas para crear las texturas de los elementos y crear diferentes objetos. La iluminación sigue el modelo de iluminación de Phong. Las fuentes de luz eran el sol cuando se sale de la mina y las antorchas dentro de la mina. Esto tiene un efecto con los demás objetos porque emiten rayos de luz que los otros objetos reciben permitiendo crear reflexiones especulares y difusas. Además, las antorchas al mismo tiempo de ser una fuente de luz se convierte en una luz ambiente y sus rayos son direccionales.

En conclusión, la computación gráfica tiene muchas aplicaciones, una de ellas siendo los videojuegos. Elementos como la iluminación, primitivas gráficas, texturas, etc. permiten crear entornos realistas generados a computadora. No obstante, por falta de tiempo y práctica es que no se pudieron explotar ni realizar para el potencial del proyecto.

References

[1] Academic (2020). Textura (gráficos por computadora). Recuperado de https://esacademic.com/dic.nsf/eswiki/831418[Consultadoel6deJuniodel2020].

[2] OpenGl (S/F). Tutorial 3 : Matrices. Recuperado de http://www.opengl-tutorial.org/beginners-tutorials/tutorial-3-matrices/#the-model-view-and-projection-matrices[Consultadoel6deJuniodel2020].

[3] Wikipedia (2019). Curva de Bézier. Recuperado de https://es.wikipedia.org/wiki/Curva_de_B\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{e\global\mathchardef\accent@spacefactor\spacefactor}\accent19e\egroup\spacefactor\accent@spacefactorzier[Consultadoel6deJuniodel2020].

[4] Wikipedia (2019). Textura (Gráficos por computadora). Recuperado de https://es.wikipedia.org/wiki/Textura_(gr\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{a\global\mathchardef\accent@spacefactor\spacefactor}\accent19a\egroup\spacefactor\accent@spacefactor1ficos_por_computadora)[Consultadoel6Juniodel2020].

[5] Galdames, O. (2011). Modelización con curvas y superficies de Bézier. https://polipapers.upv.es/index.php/MSEL/article/download/3071/3162

[6] Ribelles, J.(2013) *Modelos de iluminación y sombreado*. Institute of New Imaging Technologies, Universitat Jaume I. Recuperado de http://repositori.uji.es/xmlui/bitstream/handle/10234/120644/tema05.pdf?sequence=1&isAllowed=y