Representación del Sistema Solar en OpenGL

Sebastián Lastra Mena, Amaro Escobar
{sebastian.lastra@usach.cl}, {amaro.escobar@usach.cl}
Departamento de Matemáticas y Ciencia de la Computación
Universidad de Santiago de Chile - Avenida Libertador General Bernardo O'Higgins #3363

***Abstract*—En este trabajo se presenta el modelamiento del sistema solar con cinturón de asteroides y la enana roja Némesis. Se creó en un ambiente gráfico el cual sufrirá los cambios dependiendo de la elipse que se establezca para la enana roja. Otras implementaciones realizadas en este trabajo fueron la aplicación de texturas en el fondo de la escena utilizando la técnica de SkyBox, el posicionamiento de la ViewPort en la orbita del planeta que se requiera y la implementación de un menu para facilitar la navegación entre las funcionalidades del programa. Fue implementado en OpenGL bajo sistema operativo GNU/Linux.**

***Index Terms*—Sistema Solar, OpenGL, Orbita Elíptica, Nube de Oort, Némesis, SkyBox.**

I. Introducción

La idea principal es llevar a cabo la simulación del Sistema Solar el cual incluye un cinturón de asteroides y la presencia de una enana roja. El sistema que simula el fenómeno espacial se activa presionando un botón, lo cual genera una onda expansiva que desplaza a los asteroides de su cinturón haciendo posible la colisión de estos con los distintos planetas del sistema solar.

Considerando que lo que se presenta en esta versión del informe es una modificación a lo que ya está implementado, resulta importante mencionar que, pese a que se puede pensar que resulta más rápido, fácil e intuitivo implementar los nuevos requerimientos, se tomó gran parte del tiempo para leer, comprender, analizar e interactuar con el programa, con el fin de poder acoplar los nuevos requerimientos y no estropear el resto. Se agregan dos nuevas funcionalidades: la opción de acercarse a las órbitas de los planetas y al llegar a ella comenzar a orbitar en ella, a una velocidad controlada por el usuario. Por otro lado y como se había mencionado anteriormente se aplicó la técnica de SkyBox aplicando una textura de estrellas al fondo de la escena.

A. Objetivo General

Generar gráficamente el modelamiento del fenómeno espacial y que se aprecie lo más realista posible.

B. Objetivos Específicos

- Aprender a usar las herramientas necesarias de OpenGL para la realización de sombras, iluminación y texturas.
- Entender los conceptos fundamentales de la Computación Gráfica y cómo se expresan y/o representan en términos computacionales.
- Asociar la teoría aprendida en el curso de Computación Gráfica con la representación práctica en OpenGL.
- Lograr un dominio en el lenguaje de programación y librerías utilizadas.

II. Definición de la problemática

En el año 1984, surge una hipótesis astronómica que sustenta de que nuestro Sol forme parte de un sistema binario [1]. En este sistema, la estrella compañera del sol se llamaría Némesis y cada 26 millones de años desestabiliza la nube de Oort, la cual posee gran cantidad de cometas y estrellas [2]. Esta desestabilización generaría que los cometas que se encuentran en la nube se dirijan hacia el sistema solar causando gran cantidad de colisiones en los planetas.

Skybox es una tecnica que permita que una escena se vea más grande y más impresionante, envolviendo al espectador (usuario) con una textura que es posible apreciar en una cámara de 360° [3]. Dicha técnica utiliza imágenes con baja calidad ya que la cantidad de datos que se procesa en cada frame es mucha y si se necesitara una imagen de mayor calidad y/o resolución, es necesaria una tarjeta con procesamiento gráfico independiente para una mejor representación (Además de una optimización en cómo se aplica la textura).

Finalmente, resulta necesario trabajar con las transformaciones geométricas correctas que permitan posicionar la viewport en un planeta específico, realizando una trayectoría animada desde un punto (x_0, y_0, z_0) hacía un punto (x_1, y_1, z_1), considerando que estos orbitan en torno al punto $(0, 0, 0)$ y sus posiciones varían constantemente.

A. Problemática

Realizando un desgloce final de las problemáticas de nuestro proyecto, se tiene:

1) Calcular la trayectoria de los asteroides encontrados en la nube de Oort que impactarán con los planetas.
2) Encontrar los planetas con los cuales efectivamente harán colisión.
3) Correcta implementación de la nube de Oort, buscando la mejor representación utilizando OpenGL.
4) Implementar un menú que permita navegar entre las distintas funcionalidades presentes en el programa.
5) Aplicar la técnica de SkyBox en la escena.

6) Realizar la traslación de la ViewPort hacía el planeta deseado utilizando el menú.

III. Modelo matemático

A. Ecuación de la elipse

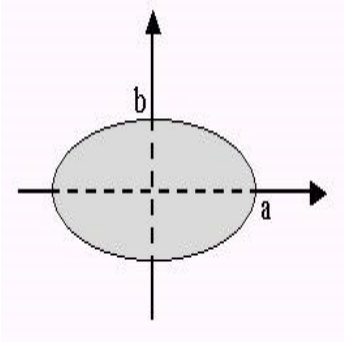

Fig. 1: Elipse

$$\frac{x^2}{a^2} + \frac{y^2}{b^2}$$

Donde $a > 0$ y $b > 0$, las coordenadas de las abscisas. La elipse se encuentra centrada en el origen $(h, k = 0)$ [4]. La ecuación de la elipse modela la ubicación de la nube de Oort y la órbita de los planetas, y la trayectoria de la enana roja Némesis.

$$\frac{x \cdot (x - x_0)}{a^2} + \frac{y \cdot (y - y_0)}{b^2}$$

Luego con la ecuación de la recta tangente a una elipse, se obtiene la dirección a la cual serán lanzadas las estrellas de la nube de Oort, y se observarán las colisiones con los planetas del sistema solar.

B. Matriz de Rotación

Las rotaciones en dos dimensiones está determinado por la siguiente matriz [5]:

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Teniendo esto, es posible generalizar dicha matriz y utilizarla para 3 dimensiones. La diferencia fundamental radica en que la transformación geométrica de rotación es posible aplicarla sobre (x, y, z), es decir, es posible aplicar la rotación con un ángulo θ sobre un eje específico. A continuación se presenta cada una de las matrices, indicando sobre qué eje se efectuará la rotación en θ grados:

Rotación sobre el eje x:

$$R_{xy}(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotación sobre el eje y:

$$R_{yz}(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotación sobre el eje z:

$$R_{zx}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Se describe como rotación en los planos xy, yz y zx, pues resulta más fácil asociar la rotación en 3 dimensiones de esa forma. Por ejemplo, al realizar una rotación yz con θ grados, el objeto realiza un giro en torno al eje y, pero en el sentido del eje z. Es posible apreciar esto gráficamente con la siguiente imagen:

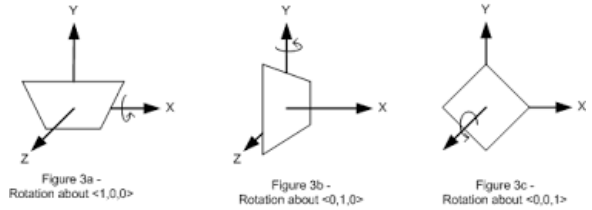

Fig. 2: Rotación sobre cada eje.

C. Matriz de Traslación

La matriz que describe la traslación de los objetos en la escena es la siguiente:

$$T = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} v_x \\ v_y \\ v_z \\ v_w \end{bmatrix} = \begin{pmatrix} v'_x, & v'_y, & v'_z \end{pmatrix}$$

IV. Aproximación utilizada en las simulaciones gráficas

La formulación del modelo en base al sistema solar y a la teoría de la enana roja respecto al diagrama de Hertzsprung-Russell [6], esto es un modelo debido a que los cálculos realizados son a escala respecto al sistema solar real, es decir, X pixeles son equivalentes a algunos años luz. Los tamaños de los planetas también son a escala y la formulación de las estrellas son pixeles arbitrarios dentro de un rango (cinturón de asteroides).

La órbita de los planetas se genera aplicando una rotación sobre el eje y (En el origen, teniendo el solo como referencia)

con un ángulo $0° \leq \theta \leq 360°$, el cual aumenta en cada frame de la escena.

Antes de realizar la rotación de los planetas es necesario posicionarlos en la escena. La traslación de los planetas permite determinar el radio que tendrá la órbita del planeta respecto al sol. Es en torno a esta transformación que se aplica la rotación, por ejemplo, en el primer frame se sitúa el objeto en la posición (x_0, y_0, z_0) y se realiza una rotación en θ_0 grados en torno al eje y, provocando que el objeto se sitúe en la posición (x_1, y_1, z_1).

V. Implementación computacional

OpenGl es el principal entorno de desarrollo, para la creación de aplicaciones graficas portátiles e interactivas en 2D y 3D. Desde su introducción en 1992, OpenGl se ha convertido en la Interfaz de Programación de Aplicaciones (API) graficas más usada por la industria, brindando miles de aplicaciones a una gran variedad de plataformas de computación. A su vez, OpenGL fomenta la innovación y velocidad del desarrollo de aplicaciones incorporando una alta variedad de formas de rendering, mapeo de texturas, efectos especiales y otras poderosas funciones de visualización [7].

Se utilizo el Entorno de Desarrollo Integrado (IDE) llamado NetBeans, el cual permite detectar rápidamente errores de sintáxis. Permite también iniciar en modo debug, herramienta con la cual es posible realizar un seguimiento del valor de cada variable en cada frame de la aplicación. Además, resulta mucho más fácil el proceso de compilación. La instalación y posterior configuración del proyecto se escapan del ámbito de este informe pero se puede encontrar información respecto a esto en nuestra Wiki del proyecto en GitHub[1].

Finalmente, pese a que existe una gran variedad de librerías diferentes a OpenGL, no se utilizaron pues la idea principal de este proyecto es aprender a utilizar al más bajo nivel posible cada uno de los elementos presentes en la computación gráfica (Transformaciones geométricas, primitivas, etc...), cosa que es posible directamente con OpenGL y no con librerías externas que agregan una capa de abstracción en la programación y, por lo tanto, un menor nivel de entendimiento de lo que se realiza. Además, la mayoría de dichas librerías sólo están disponibles para C++.

VI. Análisis de resultados

A. Validación de resultados

El trabajo es realizado es en base a un proyecto anterior del ramo de Computación Gráfica, llamado Sistema Solar (5). Este proyecto ha sido optimizado en líneas de código y utilizado para nuestro proyecto de fin de semestre. El hecho de que la nube de Oort no se pueda apreciar desde nuestro planeta, y de que la estrella Némesis sea una hipótesis astronómica, deja a la "realidad" un poco de lado. Se corresponde sí a las hipótesis científicas, a las distancias normalizadas de los planetas y sus velocidades de rotación.

[1]Más información en: https://www.github.com/delor34n/nemesis_v2/wiki

B. Rendimiento gráfico de la aplicación

La aplicación realizada quedo implementada de tal manera que su visualización es, si no rápida, realista a un nivel a escala del fenómeno real es decir, podría verse tanto en un pc especializado (poseedor de una buena tarjeta de video) o en un pc común corriente. La aplicación es optima respecto al trabajo que tomamos de base, ya que se redujeron las líneas de código, sin perder la efectividad y el fondo de la aplicación destacando estos 2 puntos anteriores, creemos que el rendimiento grafico de la aplicación es prudente(por la sencillez), rápido(por como se aprecia) y optimo(por cómo se realiza).

C. Resultados y Problemáticas

Como problema de implementación, tenemos un equipo al cual se le instaló el SO Linux Ubuntu, el cual no posee aceleración gráfica, por lo que el proyecto no alcanza su apogeo corriendo en esa máquina. No se alcanza a distinguir los movimientos de los planetas y pierde valores al momento de compilar. Por ende, la programación en ese equipo se ha vuelto truncado por problemas que escapan a cualquier tipo de solución que podamos dar.

VII. Experimentos

En la figura 1 se puede apreciar la estrella Némesis (situada en la esquina inferior derecha) sin haber tocado los asteroides de la nube de Oort. Posteriormente en la figura 2 se puede ver como son desplazados los asteroides al perder su órbita normal y ser atraídos por el sol.

VIII. Conclusiones

La realización de este modelamiento fue costosa y ardua, ya que no había conocimiento de cómo plantearnos el problema, tal vez bosquejos o ideas que cualquier persona podría tener, pero no como analistas. Luego de algunas clases y laboratorios empezamos a entender lo que nos pedían y como poder realizar el modelamiento. Después de mucha investigación, comprensión de códigos antiguos, adquisición de conocimientos sobre el lenguaje de programación utilizado y el trabajo de unas funciones, se pudo llevar a cabo el modelamiento final. Nos sentimos orgullosos de haber sido capaces de realizar un buen modelamiento de un fenómeno natural tan sorprendente. De aquí en adelante esperamos que este trabajo pueda ser reutilizado como base para proyectos mas grandes por alumnos que tendrán este ramo más adelante o bien ser capaz de optimizarlo y lograr difundirlo para algún uso particular de buena manera.

References

[1] R. Muller, "Evidence for a solar companion star," *Nature*, vol. 308, pp. 715–117, 1984.
[2] Astromia.com. (Diciembre 15, 2014.) Astronomía educativa: Tierra, sistema solar y universo. [Online]. Available: http://www.astromia.com/solar/nubeoort.htm
[3] J. YANG. (Diciembre 15, 2014.) Treasure hunting. http://www.cs.ucla.edu/~jdyang/CS274C_project_report.pdf.
[4] Enciclopedia. (Diciembre 1, 2009.) Enciclopedia. [Online]. Available: http://enciclopedia.us.es/index.php/Elipse
[5] J. Foley, *Computer graphics : principles and practice*. Reading, Mass: Addison-Wesley, 1995.

[6] A. O. T. Universe. (Diciembre 15, 2014.) Atlas of the universe. [Online]. Available: http://www.atlasoftheuniverse.com/espanol/hr.html
[7] OpenGL. (Diciembre 15, 2014.) Opengl overview. [Online]. Available: https://www.opengl.org/about/