UNIVERSIDAD NACIONAL DEL LITORAL
FACULTAD DE INGENIERÍA Y CIENCIAS HÍDRICAS

Tecnicatura en diseño
y programación de videojuegos

UNL VIRTUAL

MANIPULACION DE OBJETOS EN 2D

Unidad 1: Introducción y puesta en marcha de OpenGL y GLUT

Docente
Walter Sotil

Tutores
Emmanuel Rojas Fredini, Cristian Yones

CONTENIDOS

1. Introducción a la computación gráfica 3

1.1. Antecedentes 3

1.2. Actualidad 4

2. Renderizado vectorial vs. raster 5

2.1. Gráfico rasterizado 5

2.2. Gráfico vectorial 6

3. Rendering 6

4. API´s gráficas 7

5. OpenGL 8

6. Pipeline 9

7. Distintas GUI´s con ventana OpenGL 10

8. GLUT 11

9. Lineamientos generales de un programa gráfico 11

REFERENCIAS 13

1. Introducción a la computación gráfica

Computación gráfica
Estudia la representación y manipulación de la información geométrica o visual de una imagen creada, utilizando técnicas computacionales.

¿Qué estudia la computación gráfica?

Es el área encargada de estudiar la representación y manipulación de la información geométrica o visual de una imagen creada, utilizando técnicas computacionales.

Comenzó a emplearse a principios de los años 60 y hoy en día está completamente establecida. Los gráficos por computadora son utilizados por industrias e instituciones gubernamentales, en la educación y el hogar. La información gráfica transmitida puede ser de cualquier índole, puede ser un videojuego o animación para la industria del entretenimiento, la descripción visual de objetos y sus detalles para comprenderlos o fabricarlos, hasta la visualización científica de complejos conjuntos de datos.

1.1. Antecedentes

El primer gran avance en la gráfica realizada por computadora fue el desarrollo de *Sketchpad,* en 1962, por Ivan Sutherland. Fue el primer programa informático que permitió la manipulación directa de objetos gráficos; o sea, el primer programa de dibujo por computadora. Se trata de un sistema gráfico, creado mucho antes de que el término *interfaz gráfica* fuera concebido.

Figura 1: Ivan Sutherland con el sketchpad.

Douglas Engelbart fue otro de los pioneros en los años 60. Su principal aporte fue lograr que los seres humanos pudiéramos interactuar directamente con las computadoras. Esto lo logró por medio de la creación del *mouse* o ratón, un periférico que permite tener una interacción activa con los dispositivos y con internet. Adelantado a su época, tuvo la fuerte convicción de que las comunicaciones permitirían mejorar la sociedad.

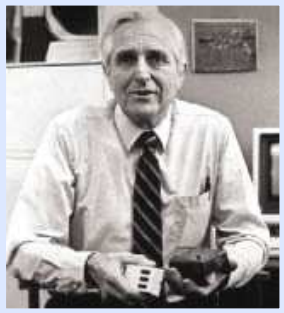

Figura 2: Douglas Engelbart con el mouse.

1.2. Actualidad

La computación gráfica (que a partir de aquí la llamaremos CG) es utilizada en videojuegos, arte, ingeniería y educación.

En lo concerniente a la industria de los videojuegos, en los últimos años hubo una gran evolución en el diseño de las placas gráficas. Este proceso ha ido elevando sostenida y rápidamente el estándar de calidad en las aplicaciones típicas de la CG, en particular los videojuegos y las animaciones.

Las placas gráficas también han evolucionado muchísimo. Comenzaron siendo una interfaz entre la CPU y el monitor, para luego adquirir capacidades de cálculo específicas y la denominación de GPU (por Graphics Processing Unit o Unidad de Procesamiento Gráfico).

En la actualidad son las GPU las encargadas de la mayor parte del cómputo masivo involucrado en la generación de la imagen. En determinado momento, los datos preparados en la CPU pasan a la GPU para seguir un protocolo de cálculos masivos en paralelo. ¿Por qué tanta matemática y cálculo? La generación de gráficos implica el modelado para simular la realidad física que daría lugar a esa imagen, la conformación de los objetos mostrados, su compleja superficie, su interacción entre ellos y con el ambiente, en particular con la luz. Pero también se modela una realidad virtual totalmente irreal, que requiere complejos personajes ficticios en un ambiente complejo que mezcla elementos de la realidad y la fantasía libre del autor. Todos esos procesos involucran una miríada de cálculos, pero permitieron pasar de los “dibujos animados”, hechos cuadro a cuadro por el artista, a la “animación 3D” en la que el artista utiliza complejas herramientas para definir un conjunto de personajes y acciones que se preparan de antemano y luego se procesan en clusters de cálculo masivo y en paralelo. Aun así, tanto las GPU como los modernos algoritmos y programas están capacitados para ayudar también en el dibujo animado y en la presentación interactiva, aquella en la que el usuario interactúa constantemente con la escena que se muestra, el ejemplo más evidente son los videojuegos, donde se pretende cada vez mayor velocidad y potencia de cálculo para generar escenas vívidas de personajes y ambientes complejos en tiempo real (desde unos 25 cuadros o frames por segundo o fps).

En el pipeline o tubería de las placas gráficas se realiza un recorrido virtual de los datos mientras van siendo procesados, primero, como datos vectoriales o geométricos, y luego, como datos raster o gráficos.

Cabe mencionar que hubo grandes avances en la programación del pipeline y también en la utilización de GPU para cálculos de propósito general, en lo que recientemente se denomina GPGPU, o GPU de Propósito General (del inglés General Purpose GPU). Por ejemplo, para invertir matrices de varios millones de coeficientes, la GPU requiere mucho menos tiempo que la CPU.

En este marco, se observa un creciente progreso en el desarrollo de simuladores, la realidad aumentada, el control inalámbrico de la consola y las implementaciones en dispositivos móviles.

Figura 3: Desarrollo de videojuegos.

Figura 4: Animación.

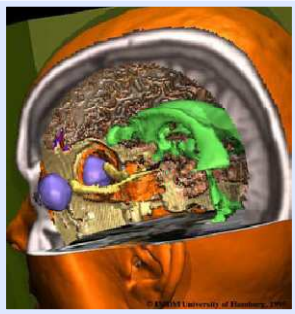

Figura 5: Visualización en medicina.

2. Renderizado vectorial vs. raster

Hay dos modos de obtener gráfica 2D: a través de gráficos vectoriales y raster.

En sus inicios se utilizaron dispositivos de gráficos vectoriales y luego, en las décadas siguientes, éstos fueron reemplazados en gran parte por dispositivos basados en gráficos raster.

2.1. Gráfico rasterizado

Una *imagen rasterizada,* también llamada mapa de bits, imagen matricial o bitmap, es una estructura o fichero de datos que representa una rejilla rectangular de píxeles o puntos de color, denominada *raster,* que se puede visualizar en un monitor de PC, papel u otro dispositivo de representación.

Imagen rasterizada
Estructura o fichero de datos que representa una rejilla rectangular de píxeles o puntos de color, denominada raster.

A las imágenes rasterizadas se las suele caracterizar por su altura y ancho (en píxeles) y por su profundidad de color (en bits por píxel), que determina el número de colores distintos que se pueden almacenar en cada píxel, y por lo tanto, en gran medida, la calidad del color de la imagen.

Figura 6: Gráfico rasterizado.

2.2. Gráfico vectorial

Gráfico vectorial
Imagen digital formada por objetos geométricos independientes, definidos por distintos atributos matemáticos de forma, posición, color, etc.

Un *gráfico vectorial* es una imagen digital formada por objetos geométricos independientes (segmentos, polígonos, arcos, etc.), definidos por distintos atributos matemáticos de forma, posición, color, etc. Por ejemplo, un círculo de color rojo quedaría definido por la posición de su centro, su radio, el grosor de línea y su color.

El interés principal en los gráficos vectoriales radica en la posibilidad que ofrece de ampliar el tamaño de una imagen a voluntad, sin sufrir el efecto de *pixelado* que padecen los gráficos pasterizados (cuando los puntos de color se transforman en cuadrados visibles).

Asimismo, permiten mover, estirar y retorcer imágenes de manera relativamente sencilla. Su uso también está muy extendido en la generación de imágenes en tres dimensiones, tanto dinámicas como estáticas.

Todas las computadoras actuales traducen los gráficos vectoriales a gráficos rasterizados para poder representarlos en pantalla al estar ésta constituida físicamente por píxeles.

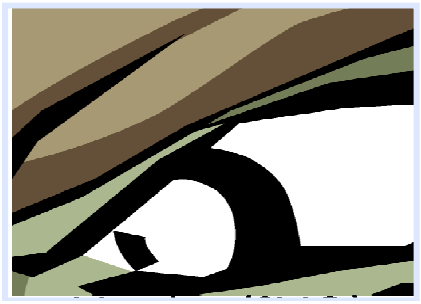

Figura 7: Gráfico vectorial.

3. Rendering

Rendering
Transformación de una escena virtual en una imagen.

Rendering es la transformación de una escena virtual en una imagen. Es el proceso final de generación de la imagen 2D o animación a partir de la escena creada. Esto puede ser comparado con el acto de tomar una foto o, en el caso de la animación, de filmar una escena de la vida real. Los pasos básicos del renderizado definen el llamado pipeline gráfico, que veremos en breve en esta unidad.

En CG se denomina *escena virtual* al conjunto de objetos que se ubican en el espacio *de la escena,* más las luces y el ambiente externo, todo virtual. Montar la escena implica, además, ubicar y apuntar la cámara. La escena es lo que se va a visualizar.

Generalmente se buscan imágenes de calidad fotorrealista, y para este fin, se han desarrollado muchos métodos especiales. Las técnicas van desde las más sencillas, como el render de alambre (en inglés, wireframe rendering), pasando por el render basado en polígonos, hasta las técnicas más modernas como *Ray Tracing, Radiosity* o *Photon Map.*

El software de render puede simular efectos cinematográficos como la profundidad de campo o el *motion blur* (imagen movida). Estos fenómenos son, en realidad, un producto de las imperfecciones mecánicas de la fotografía física, pero como el ser humano está acostumbrado a su presencia, la simulación de dichos efectos incorpora un elemento de realismo a la imagen.

Asimismo, se han desarrollado técnicas con el propósito de simular otros efectos de origen natural, como la interacción de la luz con la atmósfera o el humo. Ejemplos de estas técnicas son los sistemas de partículas que pueden simular lluvia, humo o fuego; el muestreo volumétrico para simular niebla, polvo y otros efectos atmosféricos, y las cáusticas para simular el efecto de la luz al atravesar superficies refractantes (transparentes).

El rendering necesita una gran capacidad de cálculo, debido a que requiere simular gran cantidad de procesos físicos complejos. La capacidad de cálculo se ha incrementado rápidamente a través de los años, permitiendo un mayor grado de realismo en el proceso de rendering.

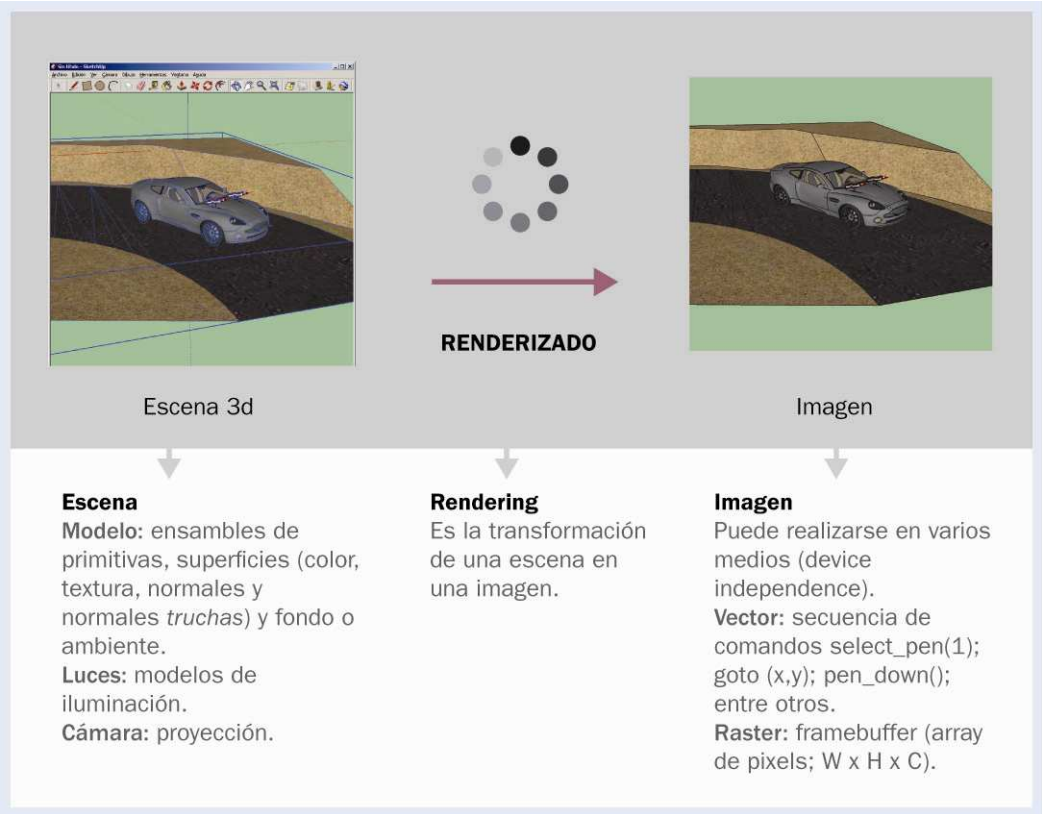

Figura 8: Rendering.

4. API´s gráficas

Una API (del inglés Application Programming Interface) es una biblioteca de rutinas para interactuar con el hardware y una interfaz que permite al programador acceder a las funcionalidades que aloja.

Entre las API´s gráficas de mayor popularidad podemos mencionar a *OpenGL* y *Direct3D,* ambas utilizadas en el desarrollo de videojuegos.

API (Application Programming Interface) Biblioteca de rutinas para interactuar con el hardware e interfaz que permite al programador acceder a las funcionalidades que aloja.

5. OpenGL

OpenGL
Es una API que permite crear aplicaciones gráficas interactivas, bidimensionales y tridimensionales.

OpenGL fue desarrollada por Silicon Graphics Inc. (SGI) en 1992 y es soportada por casi todos los sistemas operativos (Microsoft Windows, Unix, GNU/Linux, Mac OS y PlayStation). Está implementada en hardware, en todas las placas gráficas (la GPU se encarga de hacer las operaciones), y aún por software es eficiente.

Esta API permite crear aplicaciones gráficas interactivas, bidimensionales y tridimensionales. La interfaz contiene más de 250 funciones diferentes que pueden usarse para dibujar escenas tridimensionales complejas, a partir de primitivas geométricas simples como puntos, líneas y triángulos. Esta biblioteca incluye funciones para especificar las características esenciales de una escena, a saber, tipo de proyección, características de la cámara, primitivas geométricas, luces, texturas, transparencias, etc.

OpenGL funciona como una máquina de estados. Es decir, si a una propiedad se le asigna un valor determinado, todo lo que se haga a partir de ese momento se verá afectado por ese valor, hasta que éste se modifique o desactive de forma explícita.

Entre sus principales ventajas podemos destacar que es estable (perdura y se puede reutilizar), portable (como mencionamos, a casi cualquier SO), eficiente y gratuita (tiene copyright de Silicon Graphics, pero hay implementaciones realmente libres como Mesa 3D).

La primera especificación de OpenGL (OpenGL 1.0) fue publicada por Mark Segal y Kurt Akeley en enero de 1992. Posteriormente, fueron desarrollándose nuevas versiones y con cada una se incorporaron nuevas características a la API.

OpenGL 2.0, publicada en septiembre de 2004, incluye soporte para GLSL (del inglés OpenGL Shading Language). Las tarjetas publicadas con OpenGL 2.0 fueron las primeras en ofrecer shaders programables por el usuario, lo cual permite al programador sustituir el pipeline predefinido en OpenGL (llamado fixed pipeline). Más en detalle, con la publicación del 2004 empezaron a ser programables la etapa de procesamiento de vértices (vertex processor) a través del vertex program y la etapa de procesamiento de fragmentos (fragment processor) por medio del fragment program. Luego, en el 2009 se incorporó el geometry processor para procesar las primitivas y otras operaciones y en el 2010 se agregó el tessellation processor para subdividir las primitivas.

La versión más reciente de OpenGL es la 4.3, lanzada en el 2012.

Recientemente ha sido desarrollada WebGL, que es una especificación estándar para desplegar gráficos en 3D acelerados por hardware en navegadores web usando OpenGL desde Javascript. La principal característica de WebGL es que permite activar gráficos 3D en el navegador sin la necesidad de plug-ins, en cualquier plataforma que soporte OpenGL 2.0 u OpenGL ES 2.0.

La popularidad de OpenGL se debe en parte a su detallada documentación oficial. El OpenGL ARB (el comité que lo maneja) fue publicando una serie de manuales actualizados conforme la API iba evolucionando. Son fácilmente reconocibles y conocidos por el color de sus tapas.

El *Libro Rojo,* llamado oficialmente "The Red Book: OpenGl Programming Guide", es el libro de referencia y tutorial. Es considerado un libro de cabecera para programadores de OpenGL.

El *Libro Azul,* cuyo nombre original es "The Blue Book: The OpenGL Reference manual", es –en esencia– un manual de referencia de OpenGL (en los documentos de la cátedra está disponible la versión del *Libro Azul* en español).

6. Pipeline

Pipeline gráfico
Consiste en una serie de pasos más o menos rígidos que se realizan en el hardware gráfico y que transforman la escena en una imagen.

Los gráficos son generados mediante el cómputo de una serie de etapas, que habitualmente es el llamado pipeline gráfico. Consiste en una serie de pasos más o menos rígidos que se realizan en el hardware gráfico y que transforman la escena en una imagen. La mayoría de las funciones de OpenGL emiten primitivas al pipeline gráfico o bien configuran la manera en la cual el pipeline procesa dichas primitivas.

El funcionamiento básico de OpenGL consiste en aceptar entidades geométricas básicas o *primitivas*, tales como puntos, líneas y polígonos, y convertirlas en píxeles. Este proceso es realizado por el pipeline gráfico.

OpenGL es una API basada en procedimientos de bajo nivel que requiere que el programador dicte los pasos exactos necesarios para renderizar una escena. Esto contrasta con las API's descriptivas, donde un programador sólo debe describir la escena y puede dejar que la biblioteca controle los detalles para representarla. El diseño de bajo nivel de OpenGL requiere que los programadores conozcan en profundidad el pipeline gráfico, a cambio de darles libertad para implementar algoritmos gráficos novedosos.

OpenGL ha influido en el desarrollo de las tarjetas gráficas, promocionando un nivel básico de funcionalidad que actualmente es común en el hardware comercial. Algunas de esas contribuciones son:

- Primitivas básicas de puntos, líneas y polígonos rasterizados.
- Un pipeline de transformación e iluminación.
- Z-buffer (se utiliza para resolver los problemas de oclusión visual).
- Mapeado de texturas.
- Alpha blending (se utiliza para realizar mezcla de colores).

Una descripción somera del proceso en el pipeline gráfico podría ser:

1. Evaluación, si procede, de las funciones polinomiales que definen ciertas entradas, como las superficies NURBS, aproximando curvas y la geometría de la superficie.

2. Operaciones por vértices, transformándolos, iluminándolos según su material y recortando partes no visibles de la escena para producir un volumen de visión.

3. Rasterización o conversión de la información previa en píxeles. Los polígonos son representados con el color adecuado mediante algoritmos de interpolación.

4. Operaciones por fragmentos o segmentos, como actualizaciones según valores venideros o ya almacenados de profundidad y de combinaciones de colores, entre otros.

5. Por último, los fragmentos son volcados en el frame-buffer (la matriz de puntos que se va a mostrar).

Muchas tarjetas gráficas actuales proporcionan una funcionalidad superior a la básica aquí expuesta, pero las nuevas características generalmente son mejoras de este pipeline básico más que cambios revolucionarios del mismo.

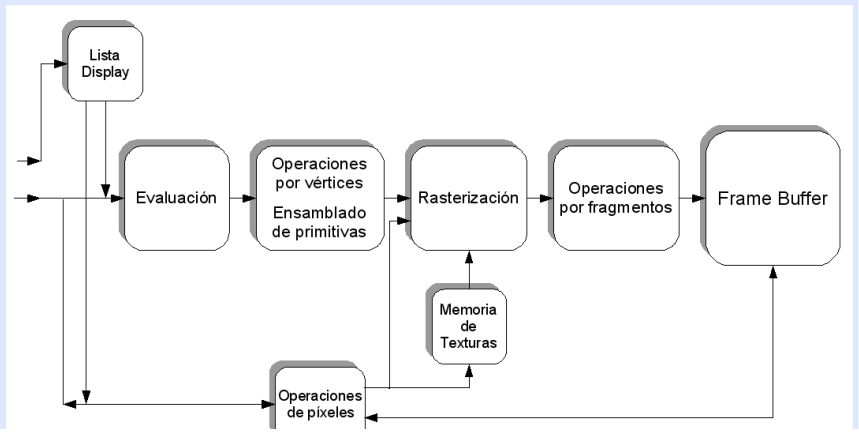

Figura 9: Funcionamiento del pipeline.

7. Distintas GUI´s con ventana OpenGL

La interfaz de usuario, es decir, el conjunto de métodos para interactuar con un ordenador, ha ido evolucionando con el paso del tiempo.

GUI (Graphic User Interface)
Conjunto de técnicas para interactuar con el ordenador.

En un principio, bastaba con una línea de comandos donde el usuario escribía las órdenes y en la pantalla veía las opciones y los resultados, siempre de manera textual.

En el futuro probablemente dispongamos de GUI´s basadas en realidad virtual, donde podamos interactuar con la máquina mediante la voz, el tacto, la dirección de la mirada, o incluso, el pensamiento. Pero, en la actualidad, las GUI´s están basadas en objetos pictóricos como iconos o ventanas, además de texto. La pieza fundamental de las modernas interfases gráficas de usuario son las ventanas.

La *GUI* (del inglés Graphical User Interface) es el conjunto de técnicas para interactuar con el ordenador. En el presente, las estándares proveen funciones tipo *callback* (consistente en suministrar la dirección de la función que deberá llamar cuando se produzca determinado evento) para crear y utilizar ventanas de aplicación y programar la entrada de datos a través del teclado, mouse, cursor, menú (texto, iconos), widgets o artilugios gráficos (slider, radio buttons, checkboxes, comboboxes, scrollbars, cuadros de diálogo, etc.), joystick, tablero digitalizador, trackball, entre otros.

Aunque disponer de una consola de comandos puede ser útil, hoy en día, toda aplicación que pretenda ser competitiva debe tener una GUI, por su comodidad de uso y porque permite un control más versátil de los elementos que interactúan.

Entre las bibliotecas multiplataforma para la implementación de GUI's, con ventanas OpenGL, se encuentran –entre otras– las siguientes:

AUX: su utilización es sólo para aprendizaje de OpenGL. Es excesivamente simple, no tiene manejo de menú y es muy limitada.

GLUT: es más moderna que AUX, muy simple, estable, fácil de aprender y utilizar. Permite callbacks para interacción con el hardware. No tiene widgets (sliders, cuadros de diálogo), no posee selector de archivos y no soporta la rueda del mouse.

Qt: es desarrollada por Nokia y debe gran parte de su popularidad al hecho de ser utilizada en el gestor de ventanas KDE. Qt cuenta actualmente con un sistema de doble licencia: una GPL para el desarrollo de software de código abierto (open source) y software libre, y otra de pago para el desarrollo de aplicaciones comerciales. Qt se encuentra disponible para las siguientes plataformas: Microsoft Windows, Mac OS X, GNU/Linux, PDA. La API de la biblioteca cuenta con manejo de archivos.

FLTK: proporciona una moderna funcionalidad de GUI, soportando gráficos 3D vía OpenGL. Se encuentra disponible para las siguientes plataformas: Microsoft Windows, Mac OS X, GNU/Linux. FLTK es provista según las condiciones de GNU Library Public License, Version 2.

8. GLUT

GLUT (OpenGL Utility Toolkit)
API con formato ANSI C, desarrollada para escribir programas OpenGL, independientes del sistema de ventanas utilizado.

GLUT (del inglés OpenGL Utility Toolkit) es una API con formato ANSI C, pensada para escribir programas OpenGL, independientes del sistema de ventanas utilizado.

GLUT es una biblioteca de utilidades para programas OpenGL, que principalmente proporciona diversas funciones de entrada/salida con el sistema operativo. Entre las funciones que ofrece se incluyen la declaración y el manejo de ventanas y la interacción por medio del teclado y el ratón. También posee rutinas para el dibujado de diversas primitivas geométricas (tanto sólidas como en modo wireframe), que incluyen cubos, esferas y teteras. También tiene soporte para creación de menús emergentes.

La versión original de GLUT fue escrita por Mark J. Kilgard.

Los dos objetivos de GLUT son permitir la creación de código más portable entre diferentes sistemas operativos (GLUT es multiplataforma) y hacer OpenGL más simple.

Todas las funciones de GLUT comienzan con el prefijo *glut.*

9. Lineamientos generales de un programa gráfico

Los lineamientos generales seguidos al desarrollar una aplicación gráfica se pueden dividir en *inicialización de OpenGL* (seteo de múltiples variables como cámara, sistema de proyección, Z-buffer, iluminación, materiales, etc.), definición de callbacks, creación de ventanas (posición, dimensiones, nombre) y loop de ejecución que reaccionará a los distintos eventos para luego refrescar determinadas variables y, finalmente, redibujar la zona de interés.

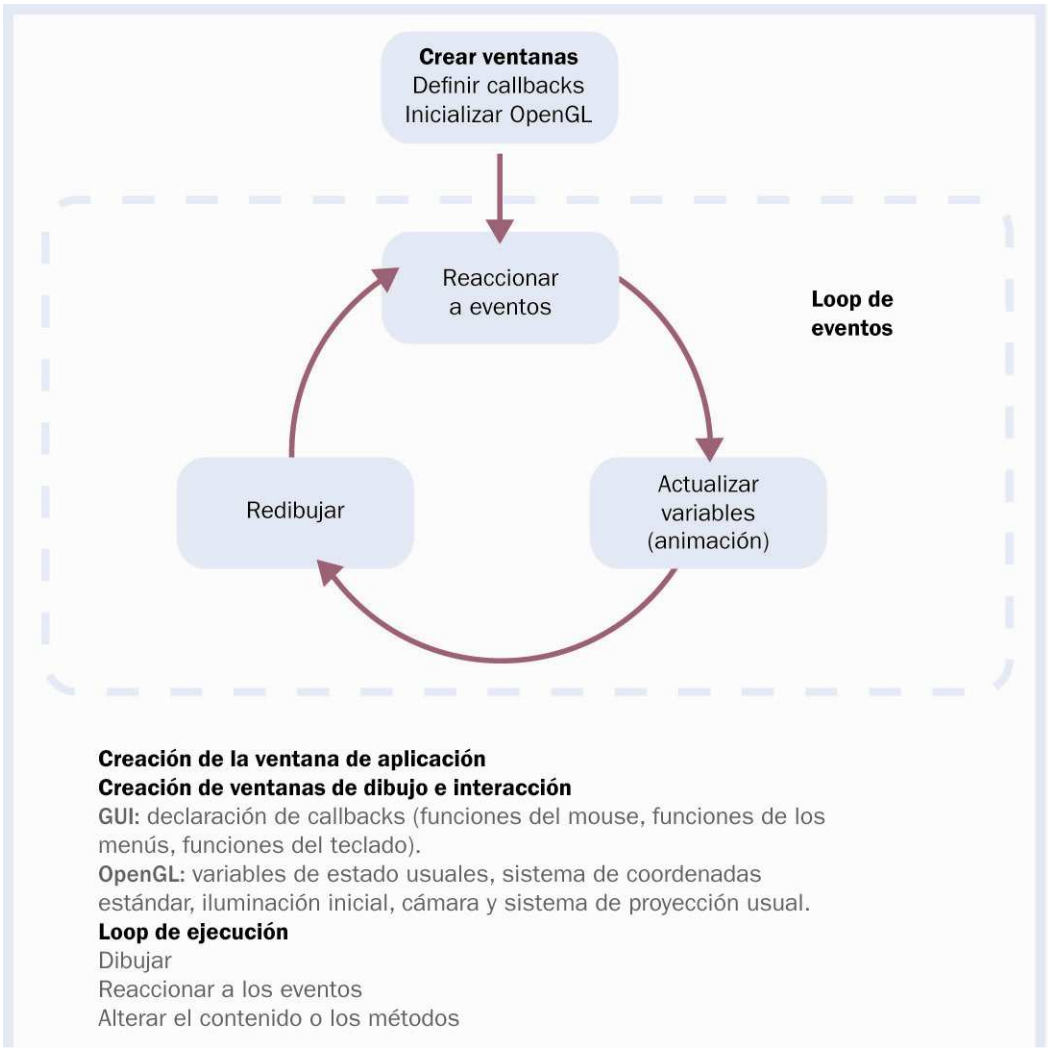

Figura 10: Lineamientos de un programa gráfico.

REFERENCIAS

Cátedra Computación Gráfica, FICH - UNL-, 2010 [en línea] [CIMEC]
www.cimec.org.ar/cg

Eran, D.; Baker, P. *Computer Graphics, C version.* Second Edition. Pearson Prentice Hall, 1997.

Kilgard, M. J. "The OpenGL Utiliy Toolkit (GLUT) Programming Interface". Silicon Graphics, Inc, 1996 [en línea] [OpenGL]
http://www.opengl.org/resources/libraries/glut/glut-3.spec.pdf

Neider J.; Davis,T.; Woo M. "OpenGl Programming Guide: The Official Guide to Learning OpenGL". Addison-Wesley Publishing Company, 1997 [en línea] [OpenGL]
http://www.opengl.org/documentation/red_book/

OpenGL [en línea] http://www.opengl.org

Reimer, J. "A History of the GUI" [en línea] [ars technica]
http://arstechnica.com/articles/paedia/gui.ars/1.

Wikipedia. *Glut* [en línea] [Wikipedia] http://es.wikipedia.org/wiki/GLUT

Wikipedia. *Gráfico rasterizado* [en línea] [Wikipedia]
http://es.wikipedia.org/wiki/Gráfico_rasterizado

Wikipedia. *Gráfico vectorial* [en línea] [Wikipedia]
http://es.wikipedia.org/wiki/Gráfico_vectorial