UNIVERSIDAD POLITECNICA DE VALENCIA

Integración de medios digitales

Memoria

GlutPlane

Curso 2k14/2k15

Apellidos, Nombre: Montañés Navarro, Edgar - edmonna@inf.upv.es

Titulación: Grado de Ingeniería Informática

20 de abril de 2015

Profesor:

Manuel Agustí

Escola Tècnica
Superior d'Enginyeria
Informàtica

Índice

Índice de figuras

1. Resumen de las ideas clave

En este trabajo se van a presentar una serie de modificaciones de *glutplane.c*, un programa de libre distribución, con las que se pretende ampliar la funcionalidad del mismo.

Por otro lado, el objetivo es que el alumno amplie sus conocimientos sobre la librería *OpenGL* y sus diferentes funciones *GLut*.

2. Introducción

Este trabajo consiste en el desarrollo de un programa, reutilizando el código de un ejemplo básico ya creado: *glutplane.c*, que se puede conseguir en la página web de *openGL*. [1]

Aquí, la faena del alumno comienza ya con la compresión total del ejemplo *glutplane*, para que así después sea capaz de poder ampliarlo y así, ampliar la funcionalidad del mismo.

Llegados a este momento, después de estar un rato sin parar de leer: ejemplo, glutplane... uno se pregunta: ¿Qué es ésto? ¿En que consiste? Este ejemplo del que se viene hablando hasta ahora, es un programa diseñado en lenguaje C, que utiliza la librería openGL y sus funciones para diseñar un escenario, donde podemos observar tres aviones.

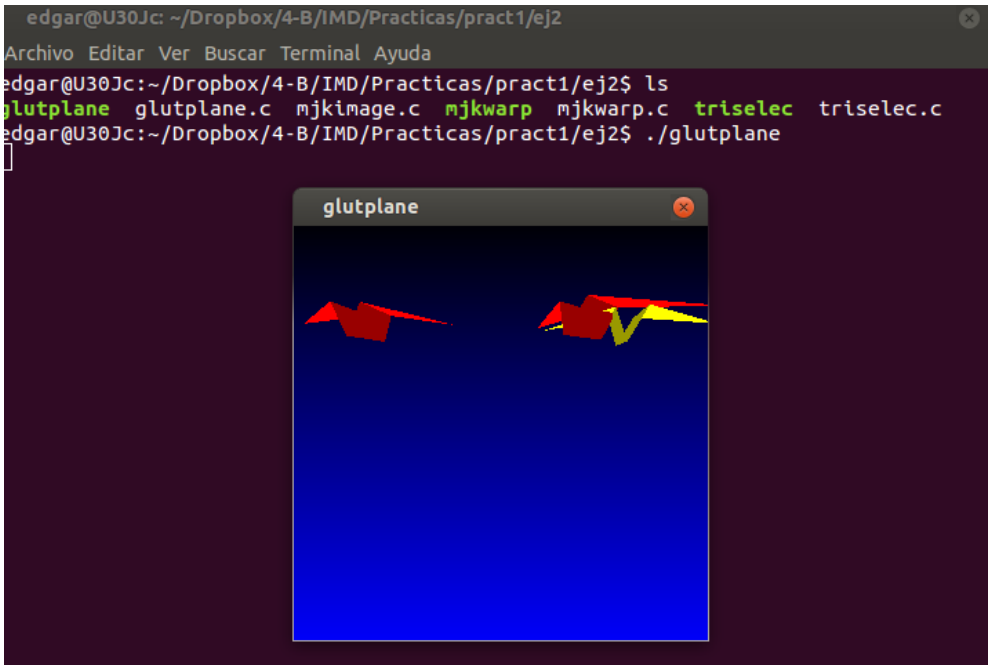

Figura 1: Captura de pantalla que muestra glutplane funcionando.

Antes de seguir hablando, nos apoyamos y hacemos uso del refrán: *Una imagen vale más que mil palabras.* para hacer más fácil y amena la explicación del programa.

[1] Código original de GlutPlane: `https://www.opengl.org/archives/resources/code/samples/glut_examples/examples.zip`

Como se puede visualizar en la imagen, en el escenario tenemos tres aviones. Estos tres aviones vuelan de manera aleatoria, sin que el usuario pueda decidir nada sobre la trayectoria de ninguno de ellos. No solo el vuelo es aleatorio, sino también, el color con el que se dibujan.

El ejemplo viene con un menú muy simple, que podemos visualizar a continuación:

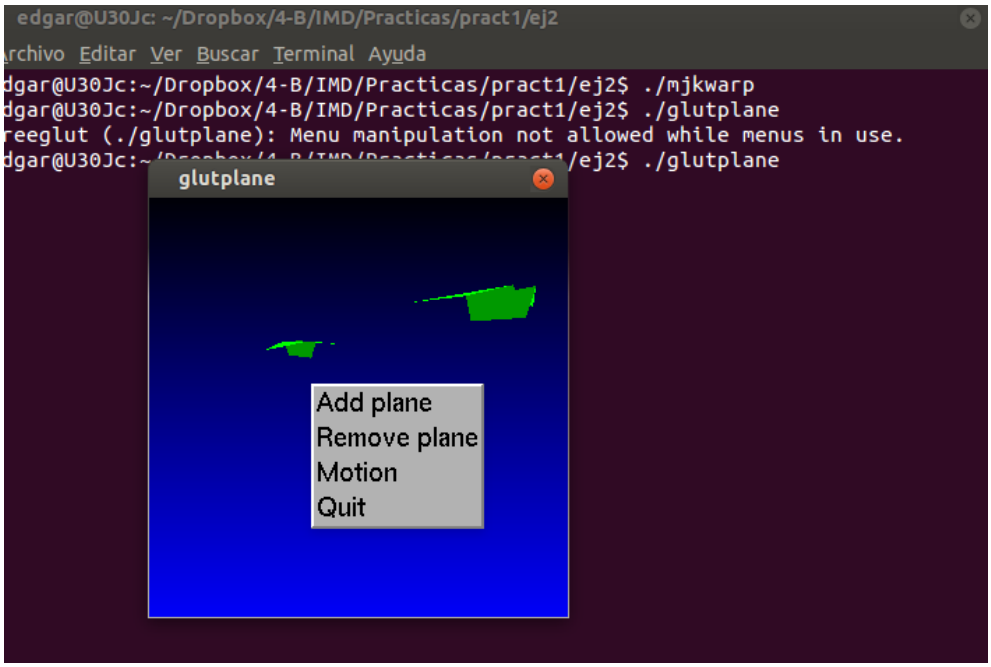

Figura 2: Captura de pantalla que muestra el menú de glutplane.

Como podemos observar en esta nueva imagen, en dicho menú solo podemos añadir nuevos aviones, borrar aviones, darle movimiento automático a los aviones (sin tener que pulsar ninguna tecla) y cerrar el programa.

Si se quiere compilar el ejemplo para probarlo uno mismo en su máquina, se hace la siguiente manera:

```
1 gcc glutplane.c -o glutplane -lglut -lGLU -lGL -lm
```

3. Objetivos

Como ya se ha comentado anteriormente, el principal objetivo, es ampliar la funcionalidad del programa original.

Han surgido bastantes y buenas ideas, que se exponen a continuación, pero aún así no se han implementado todas, debido al justo tiempo del que se ha dispuesto y a causa de diferentes problemas enconctrados durante el desarrollo de las ampliaciones.

Las ampliaciones son:

- Crear un paisaje.
- Poner sonido a los aviones.
- Poder elegir número de aviones o aleatorios.
- Poder elegir color de los aviones.
- Poder mover los aviones con teclas, al menos uno.
- ¿Qué pasa si se chocan entre ellos? Destruirlo.
- Poder disparar con los aviones, al menos el que diriges.
- Disparar a los aviones con el boli (interactuar con la cámara, que detecte la tapa del bolígrafo y dispares).

Una posible ampliación que se podría implementar sería:

- Generar un sonido 3D o 5.1.

Estas ideas, que se han expuesto aquí directamente, han sido revisadas por el profesor, quién debido a su mayor conocimiento de la situación, ha dado una serie de consejos, que son:

- El paisaje pone límites y obstáculos a la escena, con lo que debemos considerar primero el ampliar la ventana al tamaño de la pantalla.
- Introducir "viento" o que el usuario realice alguna acción y modifique así la ruta de los objetos. Esta es una posible ampliación que se debería realizar antes de ponerse con otras...
- Por otro lado, la "posible" ampliación se considera que ya es de obligada incorporación, ya que se ha realizado la práctica de audio y ya se tienen los conocimientos necesarios para realizarlo.
- Mover, al menos un avión, es interesante. Se tiene una vista de la escena que se podría poner como vista en primera persona del avión "conducido".
- La ampliación: "Disparar a los globos con el boli" debe ir precedida de una versión con el ratón.

4. Desarrollo

Una vez visto en que consiste el proyecto y las posibles ampliaciones, es momento de entrar más en detalle y tecnificar el trabajo, es decir, hasta ahora, todo lo que se ha comentado es sobre funcionamiento y ampliaciones, pero sin entrar en detalles de cómo está diseñado: lenguaje, librerías, estructuras... y esto es necesario conocerlo, para poder llevar a cabo las posibles ampliaciones y así acercarnos lo mayor posible al objetivo.

4.1. Lenguaje

Sobre el lenguaje con el que está diseñado y realizado este programa *glutplane.c* es lenguaje c. Este *software* hace uso de diversas librerías, además de la de *openGL*. Utiliza la librería para trabajar con la entrada/salida, para trabajar con operaciones matemáticas...

Las funciones que pertenecen a *stdlib.h* pueden clasificarse en las siguientes categorías: conversión, memoria, control de procesos, ordenación y búsqueda... Las funciones que pertenecen a *stdio.h* se utilizan para hacer operaciones, estándar, de entrada y salida, así como la definición de tipos necesarias para dichas operaciones. Las funciones de *math.h* están diseñadas para realizar operaciones matemáticas básicas.

4.2. Estructura

El programa contiene una función *main()* que realiza llamadas a diferentes funciones, algunas definidas y creadas dentro de nuestro código, y otras que su implementación está ya realizada.

La primera función con la que nos encontramos en la función principal es *glutInit()* que sirve para inicializar la GLUT y debe llamarse antes de cualquier llamada.

La siguiente función es *glutInitDisplayMode()*. Ésta configura la memoria gráfica según los buffers que se vayan a utilizar. Por defecto, si no se llama, la configuración es de un buffer de color con canal alfa en modo RGB. Corresponde a usar como parámetros GLUT_RGBA—GLUT_SINGLE. El parámetro es una máscara de bits por combinación or de constantes predefinidas. Se puede usar GLUT_DOUBLE en vez de GLUT_SINGLE para tener un buffer trasero además del buffer frontal (front y back buffers). GLUT_DEPTH para el uso del Z-buffer, entre otros. La llamada debe hacerse antes de la creación de la ventana.

Con respecto a la función *glutCreateWindow(char* titulo)* crea la ventana, sin mostrarla todavía, y configura todas sus características como tamaño, posición y formato de buffers. El parámetro hace referencia al título que aparece en la barra superior de la ventana (barra de título) si es que existe con el manejador de ventanas usado. No se puede dibujar todavía sobre la ventana hasta que no se ponga en marcha el bucle de eventos con *glutMainLoop()*. La función devuelve un identificador único (manejador de la ventana). El identificador sirve para referirse a esa ventana en concreto, pasarle el foco por ejemplo, cuando se usa multiventana. Cada ventana mantiene su propio contexto de dibujo que ha quedado configurado con la llamada a esta función.

La función *glutDisplayFunc()* permite registrar la función que atenderá el evento de "display". Este evento se produce cuando el contenido de la ventana de dibujo cambia por cualquier razón. La causa más frecuente es que se fuerce desde dentro de la aplicación el redibujo porque algo ha cambiado (por ejemplo en una animación). La forma de registrar la función de atención, llamada callback, es pasar su nombre como parámetro p.e.glutDisplayFunc(draw) siendo draw() la función que se activará cuando se procese el evento de redibujo.

La función *glutReshapeFunc()* como la anterior permite registrar la callback que atiende al

evento de redimensión de la ventana de dibujo. El evento de redimensión también se lanza cuando se crea la ventana.

La función *glutKeyboardFunc(keyboard)* es la función que recoge los eventos generados al pulsar una tecla. Si nos fijamos solo se recoge el evento de la barra espaciadora.

La función *glutVisibilityFunc()* establece el callback visibilidad para la ventana actual. La llamada a la función "visibilidad" de una ventana se llama cuando la visibilidad de una ventana cambia. El parámetro de devolución de llamada estado es GLUT_NOT_VISIBLE o GLUT_VISIBLE dependiendo de la visibilidad actual de la ventana. GLUT_VISIBLE no distingue de una ventana totalmente visible frente a una ventana parcialmente visible.

Las siguientes líneas de código son las que crean el menú de dicha aplicación, además de decir que, el menú aparezca cuando se pulsa sobre el botón derecho del ratón. Las líneas de código son:

```
1  glutCreateMenu(menu);
   glutAddMenuEntry("Add plane", ADD_PLANE); //Add plane
3  glutAddMenuEntry("Remove plane", REMOVE_PLANE); //Remove plane
   glutAddMenuEntry("Motion On", MOTION_ON); //Motion activated
5  glutAddMenuEntry("Motion Off", MOTION_OFF); //Motion is desactived
   glutAddMenuEntry("Quit", QUIT); //Quit program
7  glutAttachMenu(GLUT_RIGHT_BUTTON);
```

La función *glClear()* es la orden de borrado de cualquier buffer, o de reinicialización del mismo.

La función *glMatrixMode()* es la matriz del modelo. Se utiliza dicha matriz cuando se quiere trasladar, escalar y girar los objetos para que tengan el tamaño, posición y orientación que se desea.

La función *glutMainLoop()* activa el bucle de atención de eventos. Cada evento que se origine en el sistema (ratón, teclado, etc.), o sea generado por la propia aplicación (p.e. redibujo), llega a una cola de eventos que se procesa en secuencia. Todos los eventos que tengan asociada una callback por el registro anterior son atendidos por el manejador activando la callback, a la que se le pasan los parámetros del evento si éste los tuviera.

4.3. Modificaciones

Una vez ya se ha explicado qué hace cada función y de que consta el programa es hora de empezar a explicar modificaciones que se han ido realizando.

4.3.1. Entrada de menú

La primera modificación que se ha realizado es añadir una nueva entrada de menú. Esto se ha tenido que hacer porque al clicar sobre la tercera entrada (la entrada *Motion*), daba error y la

aplicación se cerraba de manera inesperada. El error reza:

```
freeglut (glutplane): Menu manipulation not allowed while menus in use.
```

Como vemos este error es debido a que intenta cambiar la entrada de menú. Esto se soluciona pues, añadiendo una nueva entrada en el menú y quitando la función de que cambie la entrada de Off a On. Se ha de añadir también, que este error no sucede en todos los ordenadores, pero sí en la máquina con la que se está desarrollando este trabajo.

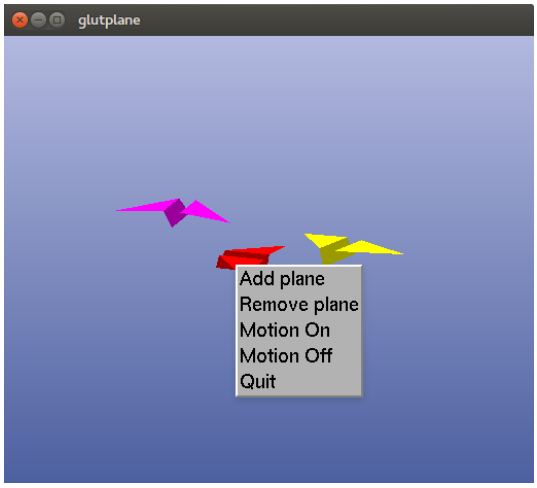

Figura 3: Captura de pantalla que muestra el nuevo Menú.

4.3.2. Modificación del fondo

La siguiente modificación que se ha realizado ha sido modificar el color del fondo. Las funciones que hay que modificar son las *glColor3f()*. Como se puede observar, esta función tiene tres argumentos, que son colores: red, green y blue. Con el número, entre 0 y 1, se define cuanta cantidad de color se quiere de cada componente. Para lograr el color, simplemente se han ido modificando los valores hasta que se ha alcanzado el color deseado. Así ha quedado el código:

```
glBegin(GL_POLYGON);
glColor3f(0.9, 0.9, 1.0);
v3f(-20, 20, -19);
v3f(20, 20, -19);
glColor3f(0.1, 0.2, 0.5);
v3f(20, -20, -19);
v3f(-20, -20, -19);
glEnd();
```

La primera idea era añadir una imagen de fondo, pero al final, se ha visto mejor, modificar el color del fondo, para no tener que trabajar también con las dimensiones de la ventana, ni tener que buscar una imagen suficientemente grande para que cuando la ventana se ampliara, se viese bien.

4.3.3. Añadiendo audio

La siguiente modificación que se ha realizado al programa, es añadir audio posicional a los aviones. Hemos aprovechado el ejemplo de *openal.c* que se nos dio en la práctica 2 de audio de la asignatura.

La primera mejora que se hizo fue dibujar tres fuentes de audio, ya que el programa inicia con tres aviones en pantalla. Dichas fuentes se dibujaban en pantalla con un pequeño cubo, para que se viese como iba variando la posición de la fuente, a la vez que variaba la posición del avión.

Las pistas de audio son tres ficheros *.wav* que se deben leer de disco. [2]

También hemos se ha creado un oyente, el cuál se puede acercar a las diferentes fuentes de audio con las flechas del teclado.

Para activar las fuentes se hace con las teclas numéricas: 1 para activar y 4 para parar la primera fuente. La segunda, con 2 para activar y 5 para parar y la tercera fuente, con 3 y 6, respectivamente.

Se ha añadido una ayuda de ejecución que se muestra al ejecutar el programa, para que le ayude al usuario a ejecutar correctamente el programa. [3]

Aquí se agrega una imagen donde se muestra como cada fuente de audio está dibujada en la posición correspondiente a su avión:

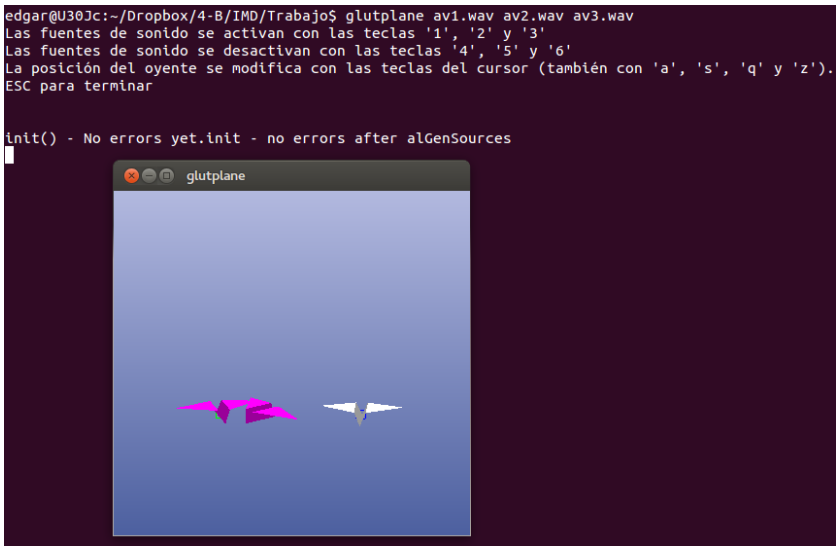

Figura 4: Captura de pantalla que muestra cada fuente de audio unida a su avión.

[2]Se adjuntan para mayor facilidad de ejecución.

[3]Lo que se conoce como: Usage.

Una vez se vio que el programa funcionaba correctamente con 3 fuentes de audio y que éstas realizaban el recorrido correcto, es decir, idéntido al del avión al que estaban asignadas, la siguiente mejora ha sido que se añada una fuente de audio, cuando se añade un avión.

Es por eso que se ha creado un vector de posiciones de fuentes, de dimensión *MAX_PLANES*. Así en el peor de los casos, que habrán 15 aviones en pantalla, habrán 15 fuentes de audio.

Aún así, hay dos pequeños inconveniente y son: por un lado, cada vez que se crea una nueva fuente de audio, se le tiene que asignar un fichero de audio de los que se le ha pasado anteriormente por parámetro. Es decir, que siempre tendremos 3 tipos de sonidos distintos. Por otro lado, otro inconveniente es, que debemos asignar teclas nuevas a cada nueva fuente que se crea. Una solución es crear desde un principio las 15 fuentes y asignarles ya las teclas que harán funcionar cada fuente en concreto.

Finalmente, no se ha realizado esta última ampliación, con lo que sólo sonarán 3 fuentes de audio.

También me gustaría añadir que, esta mejora de añadir un vector de fuentes se ha alargado en tiempo mucho más de lo previsto. Esto ha sido porque *openGL* no admite un vector de posiciones a la hora de posicionar la fuente. Se va a mostrar el código para poder realizar una explicación más entendible:

```
int i;
  ALfloat posicion_i[3];
  for (i = 0; i<MAX_PLANES; i++) {
    posicion_i[0] = Posiciones[i].x;
    posicion_i[1] = Posiciones[i].y;
    posicion_i[2] = Posiciones[i].z;
    alSourcef (source[i], AL_PITCH, 1.0f);
    alSourcef (source[i], AL_GAIN, 1.0f);
    alSourcefv (source[i], AL_POSITION, posicion_i);
    alSourcefv (source[i], AL_VELOCITY, velocidadFuentes);
    alSourcei (source[i], AL_BUFFER, buffer[0]);
    alSourcei (source[i], AL_LOOPING, AL_TRUE);
    ...
  }
```

En un principio la línea de posicionar las fuentes era:

```
alSourcefv (source[i], AL_POSITION, Posiciones[i]);
```

Desde mi punto de vista ésto es totalmente correcto, ya que le pasas una posición que tiene 3 coordinadas, y siendo así, situaría la fuente. Sin embargo, a la hora de compilarlo, éste se queja diciendo que los valores de dicha posición no son constantes y, por ese motivo, no deja compilar

el código. Como se puede visualizar, este error, finalmente, se solucionó creando un vector que solo almacenaba los tres puntos de la posición de cada fuente iésima, y de esta manera, el código compila y ejecuta correctamente.

En el código se puede visualizar que, no se ha creado un vector de velocidades de fuentes, que un principio parecería lo lógico, ya que cada fuente debería tener su velocidad concreta. Sin embargo, se ha hecho simplemente añadiendo un vector que contiene las coordenadas 0.0, 0.0, 0.0, ya que las fuentes no tiene velocidad y solo varía la posición.

Por último, mencionar que, al haber añadido audio a *glutplane*, para poder ejecuarlo ahora se hace de la siguiente manera:

```
gcc glutplane.c -o glutplane -lglut -lGLU -lGL -lalut -lopenal -lm
```

4.3.4. Cambio de color de los aviones

Como su nombre indica, la siguiente modificación que se ha realizado es, que al clicar un avión con el botón izquierdo, éste, cambia de color.

Lo que se ha conseguido es cambiar un avión de color al hacer clic con el ratón en la pantalla, pero siempre cambia de color el mismo avión, ya que se cambia directamente en el código.

En este momento que se entrega el trabajo, se estaba trabajando en identificar el avión que se está clicando.

5. Conclusión

En el momento de entregar el trabajo, las conclusiones que quedan, no son del todo satisfactorias.

En un principio, se pensaron bastantes mejoras y los objetivos eran amplios, pero conforme ha ido pasando el tiempo y se le han dedicado horas, se ha visto imposible llegar a cumplir los objetivos. La cuestión es que, la cantidad de objetivos alcanzados es poca, en relación a la cantidad de horas dedicadas.

Se ha tenido que dedicar horas a realizar otros ejemplos más simples, que no se pueden mostrar en este trabajo, para poder comprender mejor el código de *glutplane*. Ejemplos, tales como, dibujar elementos para comprender como funcionan las funciones de dibujo.

En fin, hablando personalmente, creo que he hecho lo que he podido, aunque no niego que quizás debería haber empezado a dedicarle tiempo al trabajo en fechas anteriores.

Referencias

[1] Transparencias y apuntes de los seminarios de la asignatura: Introducción a los sistemas gráficos interactivos.

[2] Escribir código de programación en nuestro documento LaTeX: `http://minisconlatex.blogspot.com.es/2012/04/escribir-codigo-de-programacion-en.html`