Ampliación Proyecto GlutPlane.

Curso 2k15/2k16

Apellidos, nombre	Moreno Domínguez, José Antonio (jomodom@alumni.upv.es) Slavov Hristov, Vasil (vaslahri@inf.upv.es)
Titulación	Grado en Ingeniería Informática
Fecha	Marzo 2016

Índice.

1. Resumen de las ideas clave

GlutPlane es un programa escrito en C que usa la librería Glut de OpenGL. En la versión básica de este programa podemos observar una ventana con varios aviones con los que podemos interactuar.

Las ampliaciones principales introducidas en el programa son las siguientes:

- Eliminar un avión con el botón central del ratón
- Cambiar el color pinchando en el último avión creado
- Hacer zoom a la visualización de los aviones mediante teclado.
- Poder Visualizar el programa en fullscreen o adaptarlo a las necesidades de visualización mediante el menú contextual del programa.
- Cambiar el modo en el que vuelan los aviones desde el menú contextual del programa.
- Única entrada en el menú para mover/detener la animación con las opciones : MOTION_ON y MOTION_OFF.
- OpenCV - Detectar objeto y mover los aviones

Y Algunas de las ampliaciones que se podrían incluir en la nueva versión del programa son:

- añadir/eliminar varios aviones a la vez.
- seleccionar un avión y poder eliminarlo.
- cambiar el color de los aviones seleccionados.
- aumentar/disminuir la velocidad de un avión seleccionado o sino de todos los aviones.
- seleccionar y mover un avión con teclas.
- desplazamiento de los aviones aleatoriamente a través de gestos captados por una cámara.
- seleccionar y mover un avión con teclas
- aumentar/disminuir la velocidad de un avión seleccionado o sino de todos los aviones
- eliminar el avión seleccionado
- Además de otras ampliaciones que podemos encontrar en la memoria de Edgar en la sección Objetivos en [4].

2. Introducción

Este documento es una memoria de trabajo donde se recogen los apartados más significativos surgidos durante la realización de las modificaciones y mejoras del programa “glutplane.c”. Además, sirve a modo de guía para su posterior entendimiento a la hora de realizar ampliaciones de las distintas partes que se han integrado en la aplicación final.

Por otro lado, esperamos ampliar nuestros conocimientos de la librería OpenGL y sus funciones Glut.

Los pasos seguidos para empezar a trabajar sobre el proyecto, han sido los siguientes:

1-Comprensión del código del ejemplo básico , este se puede descargar en la página web de openGL en [1].

2- Comprensión del código con las nuevas funcionalidades añadidas por el compañero de asignatura Edgar Montañes. Su código y la memoria pueden ser descargado desde la web de la asignatura en poliformat en [2].

3-Aprender a compilar y ejecutar los proyectos para poder realizar los test pertinentes.

4-Programación de las funcionalidades propuestas en el punto 1.

Tieniendo en cuenta todo esto,¡Empezamos!.

Glutplane es un programa diseñado en lenguaje C, que utiliza la librería openGL y sus funciones para diseñar un escenario que podemos observar en la figura1:

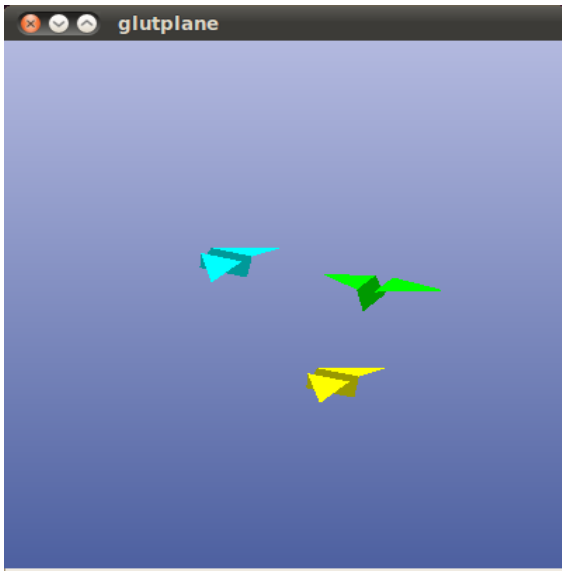

Figura 1.

Como se puede ver en Figura1, en el escenario tenemos tres aviones que vuelan de manera aleatoria, sin que el usuario pueda decidir ni su trayectoria ni el color de los mismos, en principio.

El ejemplo inicial viene con un menú simple el cual se puede ver en la figura 2:

Figura 2.

Para poder compilarlo solo hay que introducir la siguiente orden en el terminal:

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

3. Objetivos

El objetivo principal de este trabajo es intentar mejorar y ampliar la funcionalidad del programa “glutplane.c” ampliado previamente por nuestro compañero Edga Montañés,implementando todas las ideas posibles propuestas en el apartado 1 de este documento.

Todas aquellas ideas que no han podido ser implementadas , por falta de tiempo ,dificultad o problemas encontrados durante el desarrollo de las ampliaciones,se dejan planteadas como reto para el lector.

4. Desarrollo

Para empezar a desarrollar el proyecto hemos estudiado y revisado los códigos de los programas “glutplane.c” y “triselect.c” (programa que se puede encontrar en [5]), para intentar integrar algunas de las funcionalidades de triselect.c en glutplane.c.

El código de triselec.c puede ser compilado y ejecutado con la siguientes ordenes en la terminal para comprobar su funcionamiento:

```
gcc triselect.c -o triselect -lglut -lGLU -lGL

./triselect
```

Una vez visto y entendido el código de triselect.c se han conseguido implementar las siguientes funcionalidades:

- Eliminar un avión con el botón central del ratón
- Cambiar el color pinchando en el último avión creado
- Y hacer zoom a la visualización de los aviones mediante teclado.

Por otra parte:

Para la compilación y ejecución de la ampliación de glutplane.c hay que introducir las siguientes órdenes en la terminal respectivamente :

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

./glutplane x x x

Donde x x x son los ficheros de fuente de audio.
```

Nota:De momento, para ejecutar el programa hay que indicar 3 parámetros, que deberían ser 3 ficheros de audio ya que hemos utilizado partes del código del compañero del curso anterior.

4.1. Lenguaje

El lenguaje en el que está hecho el programa GlutPlane es C. Además, hacemos uso de la librerías OpenGL y otras librerías para trabajar con la entrada/salida.Para hacer operaciones matemáticas(la libreria math.h que puede ser descargada en [7]),además de la librería stdio.h , para trabajar con la entrada /salida, así como la definición de tipos necesarias para dichas operaciones .

Por otra parte se hace uso de las librerias de OpenAL y OpenCV.

4.2. Estructura básica del programa.

Puede encontrarse en [4] en la sección 4.2 de la memoria , pero aquí se documentará un pequeño resumen del mismo.

El programa contiene una función main() que realiza llamadas a diferentes funciones, algunas

para el código que hemos implementado , y otras para la implementación que está realizada en la versión inicial.

A continuación, se comentan brevemente las funciones contenidas en el método main:

glutInit() : sirve para inicializar la GLUT y debe llamarse antes de cualquier llamada.

GlutInitDisplayMode(): configura la memoria gráfica según los buffers que se vayan a utilizar(GLUT_DOUBLE , GLUT_RGB , GLUT_DEPTH o GLUT_MULTISAMPLE).

glutCreateWindow(char* titulo):crea la ventana, , y configura todas sus caracteríısticas 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).

GlutMainLoop(): bucle de eventos.

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(por ejemplo en una animación).

GlutReshapeFunc(): callback que atiende al evento de redimensión de la ventana de dibujo.

glutKeyboardFunc(keyboard):función que recoge los eventos generados al pulsar una tecla.

GlutVisibilityFunc():callback de visibilidad para la ventana actual. Devuelve GLUT NOT VISIBLE o GLUT VISIBLE dependiendo de la visibilidad actual de la ventana.

glutCreateMenu (menu) , glutAddMenuEntry y glutAttachMenu(GLUT RIGHT BUTTON) : crean el menú de dicha aplicación pulsando el botón derecho del ratón.

GlClear(): borrado y reinicialización de buffers.

glMatrixMode():matriz del modelo que se usa para trasladar, escalar y girar los objetos para que tengan el tamaño, posición y orientación que se desea.

glutMainLoop(): bucle de atención de eventos(eventos de ratón, teclado, etc).

4.3. Modificaciones.

Una vez hecho un resumen de las funciones principales de main() comenzamos explicando cada una de las modificaciones realizadas en el código:

- **Eliminar un avión con el botón central del ratón**

  En este caso se puede eliminar el último avión creado , la opción esta implementada tanto para eliminarlo desde el botón central del ratón como con el menú contextual**.**

  Código de eliminación de un avión mediante la pulsación del botón central del ratón(Vease en el cuadro 1).

```
static void
DeletePlane(GLint h)
{
  planes[h] = planes[objectCount - 1];
  cont_Aviones --;
}
//Función mouse que en caso de la pulsación central del ratón llama a DeletePlane(GLint h)
void
Mouse (int button, int state, int mouseX, int mouseY)
{
  GLint hit = 0;
```

```
 //GLint hit = 1;
 //GLint hit = 2;


 if (state == GLUT_DOWN)
  {
   //hit = DoSelect((GLint) mouseX, (GLint) mouseY);
   if (hit != -1)
        {
          if (button == GLUT_LEFT_BUTTON)
           {
             RecolorPlane (hit);
          }
           else if (button == GLUT_MIDDLE_BUTTON)
                    {
     DeletePlane(hit);
                    }



          glutPostRedisplay ();
        }
  }
}
```

Cuadro1.

- **Hacer zoom a la visualización de los aviones mediante teclado.**

  Para ello se ha añadido las siguientes instrucciones(Véase en el cuadro3).Las teclas elegidas han sido la 'T' y 't', y se hará un incremento o decremento progresivo del zoom pudiéndose aumentar o disminuir la velocidad del mismo.El usuario podrá parar el zoom pertinente adaptado a las necesidades de visualización.

En la función Draw:

```
glScalef(zoom, zoom, zoom );
```

Y En la función void keyboard (unsigned char ch, int x, int y) las siguientes opciones en el switch para procesar el zoom mediante teclado con los valores pertinentes:

```
case 't':
  zoom /= 0.99;
  glutPostRedisplay();
  break;
case 'T':
  zoom *= 0.99;
  glutPostRedisplay();
  break;
```

Cuadro 2

- **Poder Visualizar el programa en fullscreen o adaptarlo a las necesidades de visualización mediante el menú contextual del programa**

  La inicialización del programa se hará en full screen , pero podrá ser cambiado con las opciones "Litte Size Screen" y "Big Size Screen" que se muestran en el menú contextual de la figura3 pudiendose conmutar el modo de visualización.

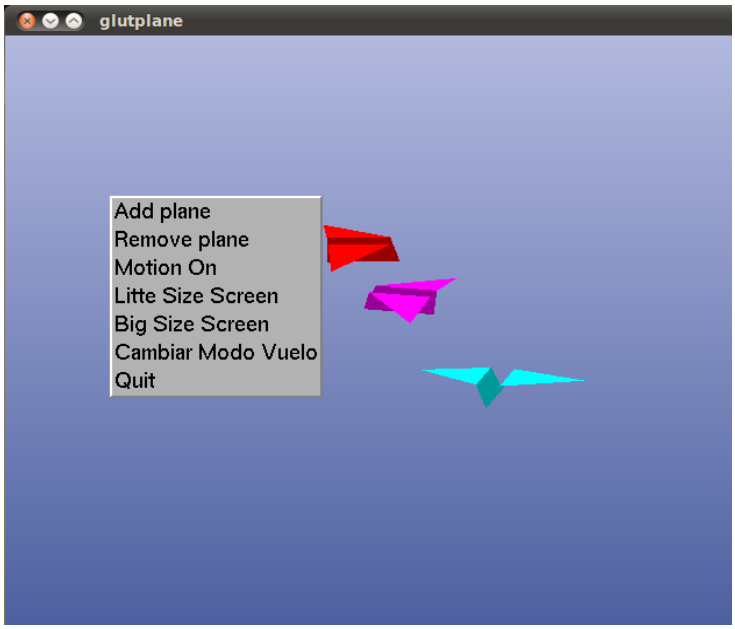

Figura 3.

Véase el código de la implementación en el cuadro 4.

Se ha implementado el siguiente código en la funcion void menu (int item), añadiendo los siguientes case al switch:

```
case SIZE_SCREEN:
```

```
glutReshapeWindow (400,400); // los parámetros son la altura y anchura
break;
case SIZE_SCREEN2:
glutFullScreen();     //Función para poner la ventana en pantalla completa.
break;
```

Y las siguientes entradas en el menú contextual con la definición de las variables pertinentes(en negrita):

```
#define SIZE_SCREEN    5
#define SIZE_SCREEN2  6

glutAddMenuEntry ("Litte Size Screen", SIZE_SCREEN);
glutAddMenuEntry ("Big Size Screen", SIZE_SCREEN2);
```

Cuadro 3.

- **Única entrada en el menú para mover/detener la animación con las opciones : MOTION_ON y MOTION_OFF.**

  Para ello simplemente se han descomentados la siguientes lineas del código de la versión del trabajo del año pasado en [4](Véase en el cuadro 5 las instrucciones en negrita);

```
case MOTION_ON:
  moving = GL_TRUE;
  glutIdleFunc (animate);
  //glutChangeToMenuEntry(3, "Motion off", MOTION_OFF);
  break;
case MOTION_OFF:
  moving = GL_FALSE;
  glutIdleFunc (NULL);
  //glutChangeToMenuEntry(3, "Motion", MOTION_ON);
  break;
```

Cuadro 4.

- **Cambiar el modo en el que vuelan los aviones desde el menú contextual del programa.(Figura 4)**

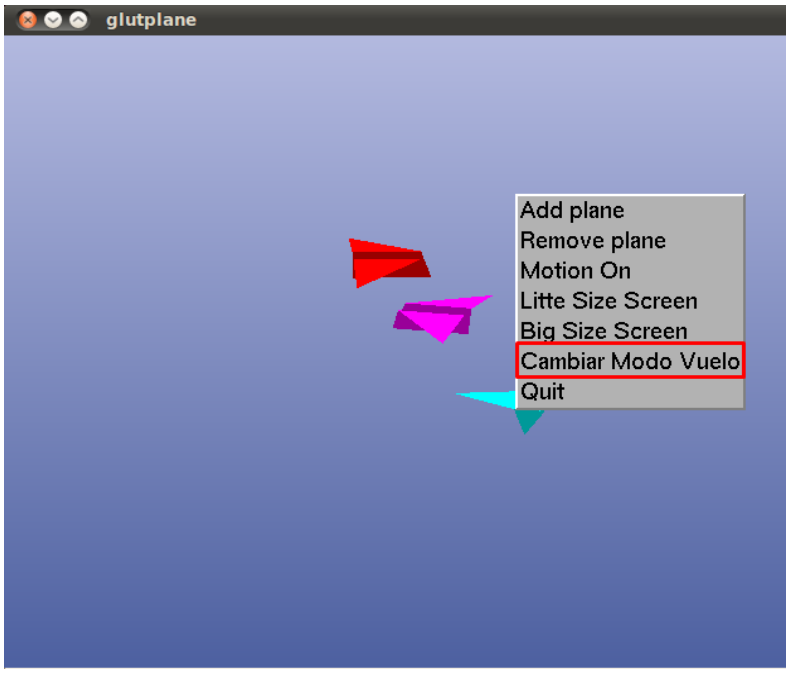

Figura 4.

Con la implementación de esta funcionalidad se ha conseguido que los aviones vuelen de forma distinta. Para ello se han cambiado los parámetros de la siguiente función glTranslatef (29.0, -4.0, -1.5).La implementación y su explicación se puede ver en el cuadro 6.

```
//Se ha añadido la siguiente entrada al menú:
 glutAddMenuEntry ("Cambiar Modo Vuelo", CAMBIAR_MODO);
// Se define la variable del case que hará las llamadas pertinentes dentro del switch de la función menú.Dentro del case se hace la llamada a la función draw2
#define CAMBIAR_MODO 7
case CAMBIAR_MODO:
       glutDisplayFunc (draw2);
     break;
//implementación función draw2 que llama a la función dibujaAvión2
void
draw2 (void) //
{
  GLfloat red, green, blue;
  int i = 0;
```

```
glScalef(zoom, zoom, zoom ); //
glClear (GL_DEPTH_BUFFER_BIT);
/*  Paint background*/
  dibujarFondo ();
/*        Generar Avión en primera persona*/
/*  //dibujarAvion(0);*/
/*  //Definimos cámara*/
/*  gluLookAt(planes[0].x, planes[0].y, planes[0].z,   //Posicion de la cámara*/
/*                          0, 0, -10,           //Punto al que miro*/
/*                          0, 1, 0              //vertical subjetiva*/
/*  );*/
  /*Paint fuentes de sonido */
  //dibujarFuentesSonido ();
  glPopMatrix ();
  /* paint planes */
  glEnable (GL_DEPTH_TEST);
  glShadeModel (GL_FLAT);
  for (i = 0; i < MAX_PLANES; i++)
    if (planes[i].speed != 0.0)
      {
          dibujarAvion2 (i, red, green, blue);
          glPopMatrix ();
      }
  glutSwapBuffers ();
  glutPostRedisplay();
}
```

La llamada a la función de dibujarAvion2 hará que cambie el modo de movimiento de los aviones ,gracias al cambio de los parámetros de la función glTranslatef (29.0, -4.0, -1.5).

Nota: la única diferencia entre las funciones dibujarAvion y dibujarAvion2 son los valores de este parámetro que es lo que produce el cambio en la forma de volar los aviones.

```
void
dibujarAvion2 (int i, GLfloat red, GLfloat green, GLfloat blue)
{

  glPushMatrix ();
  glTranslatef (planes[i].x, planes[i].y, planes[i].z);
  glRotatef (290.0, 1.0, 0.0, 0.0);
  glRotatef (planes[i].angle, 0.0, 0.0, 1.0);
  glScalef (1.0 / 3.0, 1.0 / 4.0, 1.0 / 4.0);
  glTranslatef (29.0, -4.0, -1.5);

  glBegin (GL_TRIANGLE_STRIP);
/*   left wing */
  v3f (-7.0, 0.0, 2.0);
  v3f (-1.0, 0.0, 3.0);
  glColor3f (red = planes[i].red, green = planes[i].green, blue =        planes[i].blue);
  v3f (-1.0, 7.0, 3.0);

/*   left side */
  glColor3f (0.6 * red, 0.6 * green, 0.6 * blue);
  v3f (0.0, 0.0, 0.0);
  v3f (0.0, 8.0, 0.0);

/*   right sid */
  v3f (1.0, 0.0, 3.0);
  v3f (1.0, 7.0, 3.0);

/*   final tip of right wing */
  glColor3f (red, green, blue);
  v3f (7.0, 0.0, 2.0);
  glEnd ();
```

```
}
```

Cuadro 5.

4.3.2. OpenCV - Detectar objeto y mover los aviones

Partiendo del ejemplo de reconocimiento de objetos con C++ y OpenCV [7] hemos conseguido detectar un objeto (bolígrafo rojo en este caso) y seguir su movimiento. El código de este ejemplo está en el archivo llamado detectar_boli.cpp. Para compilar y ejecutarlo hay que ejecutar las siguientes órdenes en el terminal:

```
g++ -o detectar_boli detectar_boli.cpp `pkg-config opencv --cflags --libs`
./detectar_boli
```

Con esto se debería obtener un resultado parecido a la figura 6.

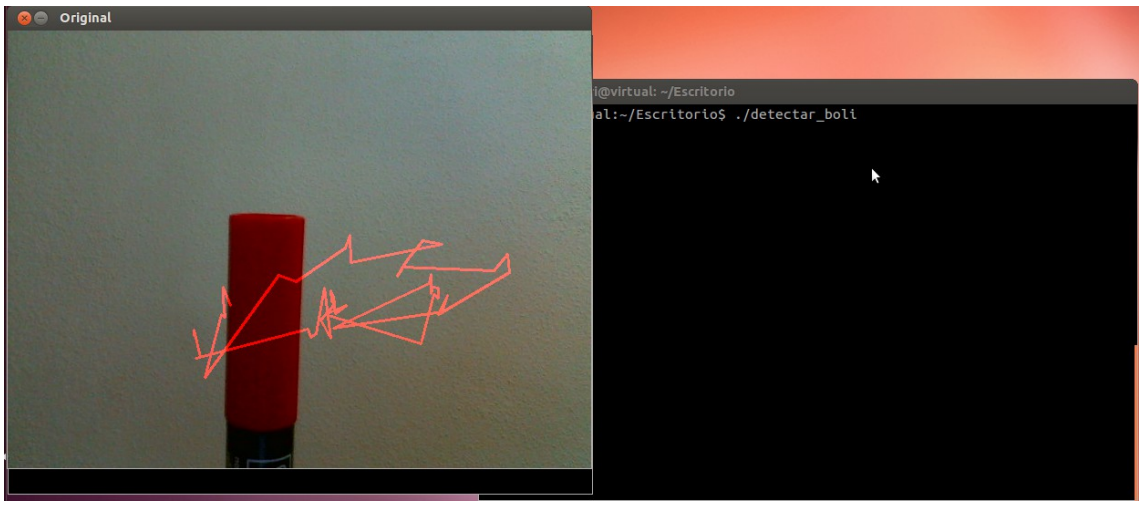

Figura 6: Detectar y seguir un objeto rojo.

En este código se intenta detectar el objeto. Para ello debe haber una diferencia notable en los colores del objeto a buscar y el fondo o resto de objetos.

En nuestro caso, el objeto a rastrear es la tapa de un bolígrafo rojo, mientras que el fondo es una pared de color blanco.

Para diferenciar los colores se procesa el vídeo con algún objeto de color rojo y se crea un nuevo vídeo binario mediante la limitación (treshold) del color rojo. A las áreas de color rojo se les asigna el valor 1 y al resto de áreas se le asigna el valor 0.

Hemos conseguido detectar la tapa del bolígrafo, pero no hemos podido implementar esta mejora en el programa Glutplane ya que nuestros conocimientos no son muy amplios y hemos detectado muchas dificultades en el intento.

5. Conclusión

A la hora de entregar el trabajo, las conclusiones que quedan, no son del todo positivas, quizás porque nos hemos propuesto muchos cambios y mejoras, y debido a la falta de tiempo y de conocimientos no hemos podido cumplir con todo.

Se ha dedicado mucho tiempo a hacer ejemplos básicos en OpenGL y OpenCV que nos han servido para comprender su funcionamiento. Aunque todo el tiempo que hemos dedicado a estos ejemplos no es tiempo perdido ya que nos ha servido para completar nuestros conocimientos en dichas tecnologías.

Como conclusión, podemos decir que quizás no hemos mejorado mucho el programa GlutPlane, pero por lo menos lo hemos intentado, y teniendo en cuenta que nuestros conocimientos de C eran muy básicos, y de OpenGL y OpenCV eran nulos, podemos decir que nuestro trabajo ha sido satisfactorio. De todas maneras, se intentará implementar el resto de mejoras antes de la presentación del proyecto.

6. Bibliografía

[1]https://www.opengl.org/archives/resources/code/samples/glut_examples/examples/glutplane.c

[2]https://poliformat.upv.es/

[3]Documentación oficial de OpenGL: https://www.opengl.org/sdk/docs/man/

[4]Memoria de Edgar Montañés en https://poliformat.upv.es/, asignatura IMD, “Trabajos de años anteriores”.

[5]https://www.opengl.org/archives/resources/code/samples/glut_examples/examples/triselect.c

[6]Documentación oficial de OpenCV: http://docs.opencv.org/

[7]https://sourceforge.net/p/mspgcc/msp430-libc/ci/master/tree/include/math.h

6.4. Compañeros de clase

Edgar Montañés (alumno del curso anterior)