UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Introducción al uso de GLUT como interfaz de alto nivel para OpenGL

Apellidos, nombre	**Agustí i Melchor, Manuel** (magusti@disca.upv.es)
Departamento	**Departamento de Informática de Sistemas y Computadores (DISCA)**
Centro	Universitat Politècnica de València

1 Resumen de las ideas clave

Cuando se desarrolla una aplicación multimedia interactiva y se necesita centrarse en la lógica de funcionamiento, es deseable abstraerse de los detalles propios de la **interfaz de usuario**. Si, además, se tiene en cuenta que esta parte es la más cambiante entre diferentes plataformas (a nivel de sistema operativo y hardware), es cuanto más aconsejable disponer de ese nivel de neutralidad o despreocupación de los detalles de este nivel que diferencian a las posibles plataformas en las que queremos desplegar la aplicación.

Así sucede a la hora de desarrollar **aplicaciones con OpenGL**, en estas el objetivo es el *renderizado* de una escena tridimensional sobre un área de pantalla bidimensional. Por ello, se propuso en los inicios de OpenGL, complementarlo con una capa de alto nivel de carácter multiplataforma que permitiera esta separación de tareas, la portabilidad del desarrollo y minimizar el impacto en cuanto a consumo de recursos necesarios. La elección recibió el nombre de **The OpenGL Utility Toolkit** (**GLUT**) [1]. Fue desarrollado por Mark Kilgard [2] allá por el 1994, sobre el sistema gráfico de ventanas X de Unix (*X Window System* y se empezaría llamando GLX). Sería portado a Microsoft Windows (WGL) por Nate Robins y en Mac OS nos encontramos con su propia implementación de GLUT[1].

Además de GLUT, otras alternativas como Freeglut y GLFW se han ido desarrollando[2]. En este artículo se presenta un ejemplo de código de OpenGL con GLUT, centrando la explicación en el uso de las operaciones propias de GLUT y en algunas "extensiones interesantes" que ha aportado Freeglut, Por lo que utilizaremos de manera indistinta los nombres de GLUT y *Freeglut*. El lector encontrará, al leer y experimentar con el contenido de este artículo, una serie de servicios propios de interfaz con el usuario que le permitirán enfocar un buen número de **prototipos de aplicaciones interactivas e inmersivas**, con una sencilla portabilidad a otras plataformas.

2 Objetivos

Por lo expuesto anteriormente, la idea básica de este artículo es ver hasta qué punto la implementación de *Freeglut* de GLUT nos permite realizar acciones de interfaz de usuario. Para ello, de modo más detallado, una vez que el alumno se lea con detenimiento este documento, será capaz de:

- Explorar las posibilidades del API de *Freeglut*.
- Poner en marcha un ejemplo básico y añadirle funcionalidades de GLUT.
- Escoger entre el modo dirigido por eventos y el de consulta de eventos.

3 Introducción

OpenGL es el estándar de facto de API[4] de gráficos 2D y 3D, independiente del sistema operativo y del sistema de ventanas. Esta especificación está implementada en un gran número de plataformas de computadores. Para usar las funciones del API de OpenGL hay que realizar una etapa de iniciali-

[1] Mac OS ha desarrollado Metal como alternativa a OpenGL. Véase más sobre qué es Metal en <https://developer.apple.com/metal/> y <https://es.wikipedia.org/wiki/Metal_(API)>.

[2] El lector puede ampliar estas opciones en la página de Related toolkits and APIs de OpenGL disponible en <https://www.khronos.org/opengl/wiki/Related_toolkits_and_APIs>.

[4] La interfaz de programación de aplicaciones o API (*application programming interface*) es la declaración de las operaciones y estructuras de datos a las que puede acceder un desarrollador de una cierta biblioteca de funciones.

zación y, dado que OpenGL es independiente de plataforma, es una tarea compleja y diferente en cada plataforma que se divide en dos partes [3]:

- La creación del contexto para OpenGL. Esto es, el punto de unión entre el sistema de ventanas (encargado de crear una ventana, gestionar los eventos propios de la entrada del usuario) y OpenGL (la estructura de datos que el servidor de OpenGL gestiona para renderizar una escena).
- La carga de las funciones de OpenGL. Generalmente, para usar bibliotecas de funciones hay que incluir las cabeceras oportunas y enlazarlas en la fase de compilación. En el caso de OpenGL, no es hasta el momento de la ejecución cuando se sabe la forma de encontrarlas (en hardware, en un manejador, por software, ...) y el conjunto de las que están disponible es diferente, puesto que depende de qué versión de OpenGL está implementada por el manejador de la tarjeta gráfica o por un componente instalado en el núcleo del sistema operativo y de la versión instalada.

Estas tareas no forman parte de la especificación de OpenGL; por lo que, a su alrededor, diferentes bibliotecas (*toolkits* en la jerga de OpenGL) han ido apareciendo para dar solución en diferentes plataformas a la **creación de ventanas y a la gestión de la entrada del usuario**. En el sitio web de OpenGL [3], se mencionan tres: GLUT, Freeglut y GLFW. En este artículo nos centramos en Freeglut: una solución multiplataforma (Windows, Mac OS X y *nix, con una versión para Android) para gestión de ventanas y eventos de entrada. Su API es un superconjunto del de GLUT y se mantiene actualizada. Soporta la creación de contextos OpenGL en el modo *core* (véase apartado 4.1).

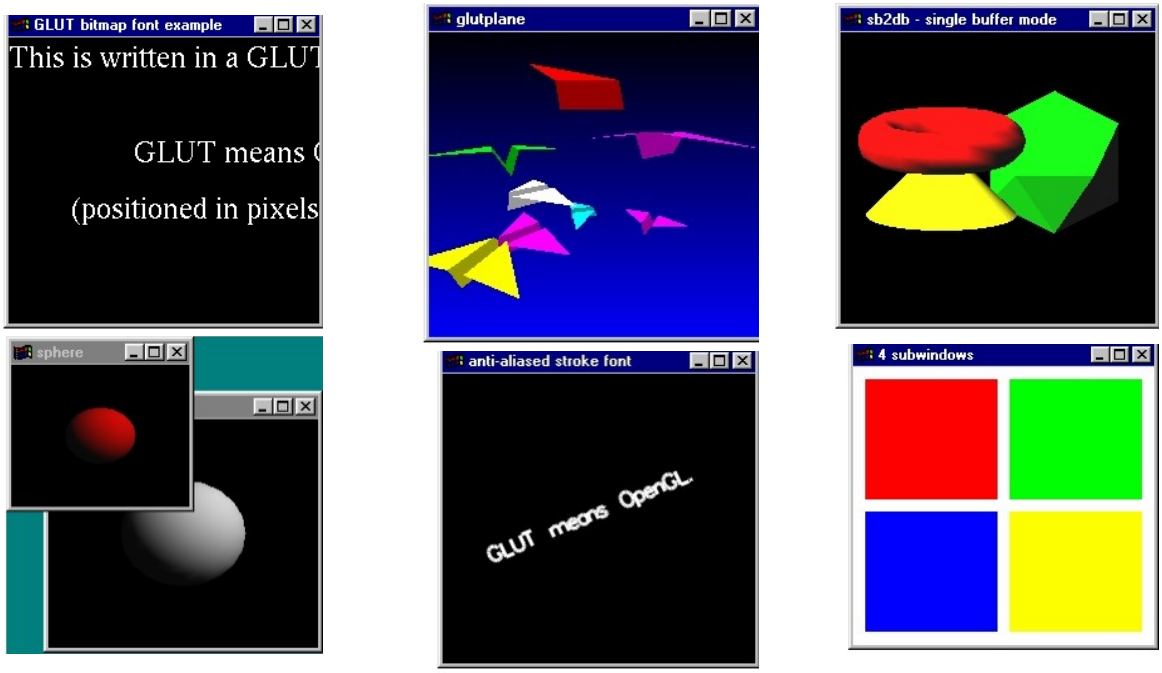

Figura 1: Algunos de los ejemplos de OpenGL con GLUT disponibles en [4] (de izquierda a derecha y de arriba abajo): bitfont, glutplane, sb2db, sphere, stroke y subwind.

4 Desarrollo

Se va a partir de un ejemplo existente, dejando de lado las operaciones propias de dibujado de OpenGL, para centrarnos en el uso de las funciones de GLUT/Freeglut. Para que el contenido sea in-

teresante y dinámico, vamos a escoger el *glutplane* (véase Figura 1). Se encuentra este ejemplo en uno de los de Mark J. Kilgard [4]. Hay unas cuantas interesantísimas "*demos*" del propio Mark en [5].

En este apartado se aborda:

- Cómo poner en marcha el ejemplo de *glutplane*, el entorno de ejecución y desarrollo.
- *Se* describirá qué elementos de GLUT contiene y cómo resolver un error de ejecución que se obtiene en este ejemplo escogido.
- Después se verá cómo incorporar otras características de *Freeglut*.

4.1 El entorno

Este artículo se ha desarrollado sobre plataforma GNU/Linux, pero es portable a otras plataformas por el hecho de utilizar OpenGL y GLUT. En Linux tenemos la implementación de la especificación OpenGL conocida como Mesa[5] [6], que ofrece una implementación de código abierto para OpenGL, OpenGL ES, Vulkan y OpenCL (entre otras). Mesa ofrece, en función del hardware instalado, desde

```
$ glxinfo -B
name of display: :0
display: :0  screen: 0
direct rendering: Yes
Extended renderer info (GLX_MESA_query_renderer):
    Vendor: Intel (0x8086)
    Device: Mesa Intel(R) UHD Graphics 630 (CML GT2) (0x9bc8)
    Version: 21.2.6
    Accelerated: yes
    Video memory: 3072MB
    Unified memory: yes
    Preferred profile: core (0x1)
    Max core profile version: 4.6
    Max compat profile version: 4.6
    Max GLES1 profile version: 1.1
    Max GLES[23] profile version: 3.2
OpenGL vendor string: Intel
OpenGL renderer string: Mesa Intel(R) UHD Graphics 630 (CML GT2)
OpenGL core profile version string: 4.6 (Core Profile) Mesa 21.2.6
OpenGL core profile shading language version string: 4.60
OpenGL core profile context flags: (none)
OpenGL core profile profile mask: core profile

OpenGL version string: 4.6 (Compatibility Profile) Mesa 21.2.6
OpenGL shading language version string: 4.60
OpenGL context flags: (none)
OpenGL profile mask: compatibility profile

OpenGL ES profile version string: OpenGL ES 3.2 Mesa 21.2.6
OpenGL ES profile shading language version string: OpenGL ES GLSL ES
3.20
```

Listado 1: Salida de la orden glxinfo.

[5] El proyecto Mesa fue iniciado por Brian Paul <brian.paul@tungstengraphics.com>.

una implementación totalmente software, hasta la aceleración proporcionada por las modernas GPUs[6].

Al instalar Mesa se obtienen, entre otras cosas, dos aplicaciones: *glxgears* (que cuantifica el rendimiento de nuestro sistema contando el número de cuadros que ofrece en *frames per second* o FPS) y *glxinfo* (que muestra información de nuestro sistema sobre las versiones del estándar OpenGL y del cliente instalado que lo implementa).

Podemos ver la salida abreviada de *glxinfo* en el Listado 1, en el que se observan dos bloques. Por una lado, las líneas 5 a la 17 que identifican la tarjeta gráfica y, por otro, las líneas 18 a la 30 que identifican la versión de Mesa instalada y las versiones que proporciona de los de perfiles que define OpenGL. Los términos *core* y *compatibility* hacen referencia al nivel de características/funcionalidades de OpenGL implementadas, puesto que algunas son declaradas *deprecated* (obsoletas) y se acaban dejando fuera de la especificación. En *core profile* ya no están disponibles las que se ha decidido eliminar, mientras que en *compatibilty profile* todavía estarán disponibles; facilitando así el mantenimiento entre versiones.

4.2 El código del ejemplo

Como se ha dicho, el ejemplo escogido es *glutplane* [4] (véase Figura 1) cuyo código fuente muestra la autoría y la licencia del mismo: "Copyright (c) Mark J. Kilgard, 1994.·This program is freely distributable without licensing fees and is provided without guarantee or warrantee expressed or implied. This program is -not- in the public domain."

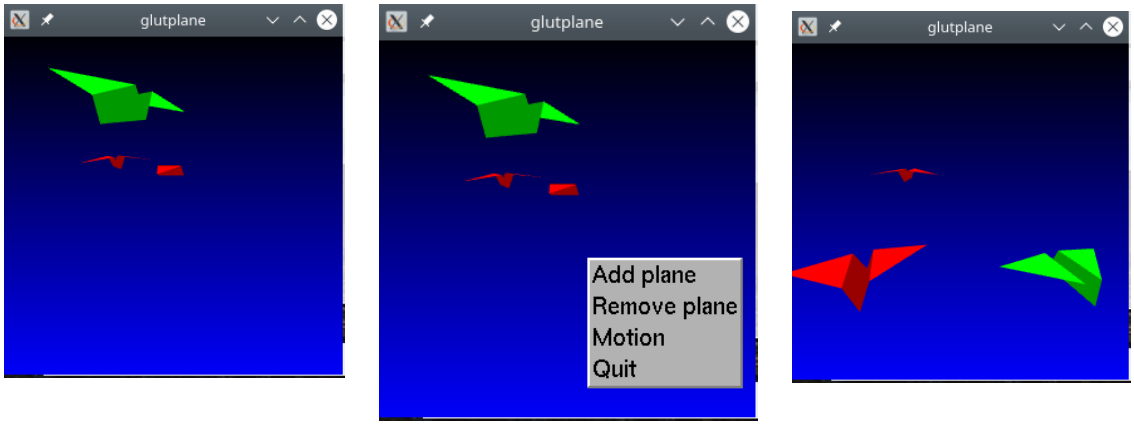

Figura 2: Salida de la versión original de glutplane: de izquierda a derecha, escena inicial, menú asociado al ratón desplegado y objetos en movimiento tras escoger "Motion" en el menú.

Proceda a descargar el código original de [4] o del repositorio que se ha creado[7] para dar soporte a este artículo y compile la aplicación con la orden:

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

Debe obtener una escena como la que muestra la Figura 2. Compruebe el error que sucede al escoger la entrada de menú correspondiente a "Motion" para que empiecen a volar los avioncitos.

[6] Graphics Processing Unit o unidad de procesamiento de gráficos, son los componentes centrales de las tarjetas gráficas actuales.

[7] Encontrará el código a que se refiere en este artículo en el repositorio de Github <https://github.com/magusti/OpenGL_examples/GLUT_API>.

4.3 El código ampliado

La parte inicial del código original no la vamos a modificar, por ello el Listado 2 muestra las funciones encargadas del dibujado en pantalla replegadas, si quiere ver el código está en el repositorio indicado y en [4].

```
1.  #include <stdlib.h>
2.  #include <stdio.h>
3.  #include <unistd.h>
4.  #define random rand
5.  #define srandom srand
6.  #include <math.h>
7.  #include <GL/glut.h>
8.
9.  GLboolean moving = GL_FALSE;
10.
11. #define MAX_PLANES 15
12. struct {
13.   float speed;          /* zero speed means not flying */
14.   GLfloat red, green, blue;
15.   float theta;
16.   float x, y, z, angle;
17. } planes[MAX_PLANES];
18.
19. #define v3f glVertex3f  //v3f was the short IRIS GL name for glVertex3f
20.
21. void draw(void) {... }
22. void tick_per_plane(int i) {  … }
23. void add_plane(void) { ...
24. void remove_plane(void) {... }
25. void tick(void) {... }
    ...
```

Listado 2: Ejemplo de GLUT: glutplane_revisat.c.

En la función *keyboard* (Listado 3) que recoge las pulsaciones de teclado hemos introducido la mayor parte de cambios, relacionados con exponer otros servicios (funcionalidades) propios de *Freeglut*. Para ello hemos declarado la variable *elCursor* en la línea 26.

Como resultado de las modificaciones introducidas en las acciones asociadas al teclado (función *keyboard*, Listado 3), hay que mencionar que se ha incorporado la operativa del menú asociado al ratón con pulsaciones del teclado, como:

- Añadir o eliminar un avión (letras ‘a’ y ‘r’, respectivamente) en las líneas 30 y 31.
- Cambiar el estado de movimiento o no de los aviones (con la tecla ‘m’), en las líneas 32 a la 36, a diferencia de la acción ya existente para la barra espaciadora (líneas 37 a la 39) que solo mueve un paso a todos los aviones.
- Y finalizar la aplicación con la tecla ESC, en la línea 52.

Además, en el código de gestión de los eventos de teclado (función *keyboard*, Listado 3) se han incorporado otras funcionalidades como el cambio de tamaño de pantalla, alternando entre el valor actual y la pantalla completa (utilizando la tecla ‘f’), en la línea 41. También es posible el cambio de la imagen asociada al cursor (tecla ‘c’, líneas 42 a la 46) que va incrementando, de modo cíclico, el

índice para mostrar las diferentes opciones existentes. Se pueden realizar diferentes acciones sobre la ventana como mostrarla, esconderla o minimizarla, entre las líneas 47 a la 49 y cambiar el texto que se muestra en la barra de título o como icono, en las líneas 50 a la 51.

```
...
int elCursor = GLUT_CURSOR_RIGHT_ARROW;

void keyboard(unsigned  char ch, int x, int y) {
  switch (ch) {
   case 'a': add_plane();    break;
   case 'r': remove_plane();    break;
   case 'm':
    moving = !moving;
    if ( moving ) { glutIdleFunc(animate); }
    else  { glutIdleFunc(NULL); }
  break;
  case ' ':
    if (!moving) { tick(); glutPostRedisplay(); }
    break;
   case 'h': printf("Ajuda\n\h\t Esta ajuda ...\n");   break;
   case 'f': glutFullScreenToggle();   break;
   case 'c':
    glutSetCursor( elCursor++ );
    if ( elCursor > GLUT_CURSOR_INHERIT )
     elCursor = GLUT_CURSOR_RIGHT_ARROW ;
    break;
   case 's': glutShowWindow();   break;
   case 'd': glutHideWindow();   break;
   case 'w': glutIconifyWindow();   break;
   case 't': glutSetWindowTitle( "Canviant el títol!");   break;
   case 'i': glutSetIconTitle( "Canviant el titol de l'icono"); break;
   case 27:  /* ESC */ exit(0);     break;
  }
}
...
```

Listado 3: Ejemplo de GLUT: glutplane_revisat.c (2).

Se explicará el contenido del Listado 4 en el apartado 4.4, ya que merece un apartado propio. Así que resta explorar las funciones de GLUT que se utilizan en el programa principal (véase Listado 5) y que son:

- La inicialización de la librería en la línea 94 *conglutInit*.
- La creación del contexto de OpenGL con *glutInitDisplayMode* y *glutCreateWindo*w, en las líneas 96 y 97.
- Las funciones de callback para las acciones de dibujo (*glutDisplayFunc*, en la línea 98) o los eventos de teclado (con *glutKeyboardFunc* en la línea 99).
- La creación del menú y su asignación al ratón con *glutVisibilityFunc*; *glutCreateMenu*, *glutAddMenuEntry*, *glutAttachMenu*(y *glutMenuStatusFunc*, en las líneas 101 a la 107.
- El bucle de eventos, línea 123, con *glutMainLoop*, que revisaremos en el aparatado 4.5.

Y, ya que estamos, si dejamos crecer (línea 11 del Listado 2) el número máximo de aviones de papel de 15 a 150… ¡La diversión aumenta!

```
...
55.  void menu(int item) {
56.   switch (item) {
57.   case ADD_PLANE:
58.     add_plane();      break;
59.   case REMOVE_PLANE:
60.     remove_plane();      break;
61.   case MOTION_ON:
62.     moving = GL_TRUE;
63. //Era:      glutChangeToMenuEntry(3, "Motion off", MOTION_OFF);
64.     glutIdleFunc(animate);
65.     break;
66.   case MOTION_OFF:
67.     moving = GL_FALSE;
68. //Era:      glutChangeToMenuEntry(3, "Motion", MOTION_ON);
69.     glutIdleFunc(NULL);
70.     break;
71.   case QUIT:
72.     exit(0);      break;
73.   }
74. }
75.
76. void menuStatusFunc( int status, int x, int y )  {
77.     if ( status == GLUT_MENU_IN_USE )  {
78.      //
79.     }
80.
81.
82.     if ( status == GLUT_MENU_NOT_IN_USE ) {
83.         if ( moving )
84.             glutChangeToMenuEntry(3, "Motion off", MOTION_OFF);
85.         else
86.             glutChangeToMenuEntry(3, "Motion", MOTION_ON);
87.
88.         glutPostRedisplay();
89.     }
90. }
...
```

Listado 4: Ejemplo de GLUT: glutplane_revisat.c (3).

Recuerde que los compilaremos desde la línea de el terminal con la orden

$ gcc glutplane_revisat.c -o glutplane_revisat -lglut -lGL -lm

4.4 Resolución del cambio de las entradas de menú

Si lo ha comprobado ya, habrá podio observar que al escoger la entrada de menú correspondiente a “Motion” para que empiecen a volar los avioncitos, se obtiene este error en la línea de órdenes:

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

Así se advierte ya en la documentación de GLUT[8]: "Remember that it is illegal to create or destroy menus or change, add, or remove menu items while a menu (and any cascaded sub-menus) are in use (that is, "popped up"). Use the menu status callback to know when to avoid menu manipulation.".

```
    ...
91. int
92. main(int argc, char *argv[])
93. {
94.   glutInit(&argc, argv);
95.   /* use multisampling if available */
96.   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH |
    GLUT_MULTISAMPLE);
97.   glutCreateWindow("glutplane");
98.   glutDisplayFunc(draw);
99.   glutKeyboardFunc(keyboard);
100.
101.  glutVisibilityFunc(visible);
102.  glutCreateMenu(menu);
103.  glutAddMenuEntry("Add plane", ADD_PLANE);
104.  glutAddMenuEntry("Remove plane", REMOVE_PLANE);
105.  glutAddMenuEntry("Motion", MOTION_ON);
106.  glutAddMenuEntry("Quit", QUIT);
107.  glutAttachMenu(GLUT_RIGHT_BUTTON);
108.
109.  glutMenuStatusFunc( menuStatusFunc );
110.
111.  /* setup OpenGL state */
112.  glClearDepth(1.0);
113.  glClearColor(0.0, 0.0, 0.0, 0.0);
114.  glMatrixMode(GL_PROJECTION);
115.  glFrustum(-1.0, 1.0, -1.0, 1.0, 1.0, 20);
116.  glMatrixMode(GL_MODELVIEW);
117.  /* add three initial random planes */
118.  srandom(getpid());
119.  add_plane();
120.  add_plane();
121.  add_plane();
122.  /* start event processing */
123.  glutMainLoop();
124.  return 0;             /* ANSI C requires main to return int. */
125.}
```

Listado 5: Ejemplo de GLUT: glutplane_revisat.c (y 4).

¿Cómo se puede resolver? Como dice la documentación y como hemos hecho ya, en el Listado 4 se muestran comentadas las líneas 63 y 68 de la función *menu* y ha podido ver una función *menuStatusFunc* (líneas 76 a la 90, del Listado 4) que no estaba en la versión original del programa y que en el programa principal (Listado 5), se ha asignado con la función *glutMenuStatusFunc*. La función *menuStatusFunc* es la encargada de realizar los cambios una vez que el menú no esté en uso y, la próxima vez que se despliegue se verán ya aplicados.

[8] Véase en la URL <https://www.opengl.org/resources/libraries/glut/spec3/node90.html>.

4.5 Cambio de estrategia de gestión de eventos

Es posible escoger entre que la aplicación esté funcionando en un modo dirigido por eventos o en un modo de consulta de eventos para la implementación de la misma. Esto depende de las restricciones temporales que pueda tener la aplicación, como por ejemplo la carga de audio desde fichero o la interacción más compleja con el usuario a través del uso de, por ejemplo, la Visión por Computador.

```
1.      while( 1 ) { //appFuncionant )      {
2.       glutMainLoopEvent();
3.       if ( moving ) {   animate();   }
4.       glutPostRedisplay();
5.   }
```

Listado 6: Ejemplo de reescritura del bucle infinito de glutManLoop con glutMainLoopEvent.

La línea 123 del Listado 5, *glutMainLoop*, se puede sustituir por un bucle como el que muestra el Listado 6. El funcionamiento en ambos casos es un bucle, pero la condición se podría poner como una expresión que evalúa cierta condición y establece si se continúa o no el programa. Lo cual no es posible con *glutMainLoo*p, excepto por el cierre de la ventana.

Observe el lector que ha sido necesario incluir la función *animate* que actualiza los parámetros de la aplicación, pero que el resto de funciones registradas con las *callback* (draw, keyboard, o el menú) siguen operativas y que la llamada a *glutPostRedisplay*, dentro de ese bucle se puede hacer al principio del mismo o al final, como está en el listado.

Si no se registra ninguna de las acciones como *callbacks*, se puede decidir en ese bucle modificado del Listado 6, cuándo y cómo ejecutar las acciones indicadas de refresco, redimensionado, consulta de teclas pulsadas o acciones realizadas con el ratón. etc.

5 Conclusión y cierre

Como el lector habrá podido observar al seguir el artículo, si lo ha ido comprobando conforme lo leía, hemos explorado las posibilidades del API de *Freeglut*, *a partir de un e*jemplo básico existente al que se le han añadido funcionalidades de GLUT; llegando a aspectos avanzados como la sincronización de las acciones del programa (que pueden ser totalmente externas al uso de los gráficos como eje central de la aplicación) o la actualización de la escena gráfica (OpenGL y *Freeglut*) ¡y hemos ampliado el número de aviones, véase la Figura 3!

Quiero aprovechar para animar al lector a probar alguna otra función de *Freeglut* como *glutLeaveMainLoop* y otras tantas que se pueden consultar en [7]. Así como las opciones que se han desarrollado para el uso de *Freeglut* sobre Android[9].

Espero que a estas alturas, tenga ejecutándose el código propuesto y comprobando que puede ver volar los aviones de papel. Venga, ¿nos animamos a lanzarles "chicles" para derribarlos y darnos un tiempo máximo para hacerlo? Le faltan dos detalles de este estilo y un poco de sonido de efectos y ya tendrá una aplicación que nos querrá enseñar. Puede descargar el ejemplo del repositorio creado a tal efecto en *GitHub*[9]. Y no cierre este el documento sin haberlo ejecutado antes. ¡¡ÁNIMO!!

[9] El lector interesado las puede encontrar en la URL <https://freeglut.sourceforge.net/docs/android.php>.

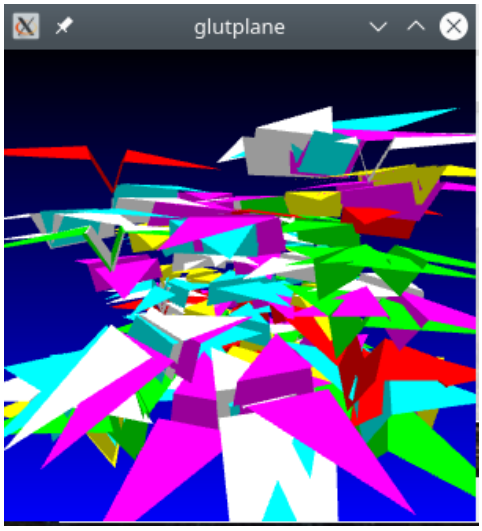

Figura 3: Salida de la versión revisada de glutplane.

6 Bibliografía

[1] Kilgard, M. J. (1996). GLUT - The OpenGL Utility Toolkit. Silicon Graphics, Inc. Disponible en: <https://www.opengl.org/resources/libraries/glut/glut_downloads.php>.

[2] Kilgard, M. (1994). OpenGL and X, Column 1: An OpenGL Toolkit. The X Journal. Disponible en <https://www.opengl.org/resources/libraries/glut/glut.column1.ps.gz>.

[3] Khronos Group. Getting Started. OpenGL Wiki-. Disponible en: <https://www.khronos.org/opengl/wiki/Getting_Started>.

[4] OpenGL examples.Disponible en la URL: <https://www.opengl.org/archives/resources/code/samples/glut_examples/examples/examples.html> .

[5] OpenGL demos. Disponible en: <https://www.opengl.org/archives/resources/code/samples/glut_examples/demos/demos.html>.

[6] The Mesa 3D Graphics Library. Disponible en: <https://www.mesa3d.org/>.

[7] The freeglut Programming Consortium. (2013).The Open-Source OpenGL Utility Toolkit (freeglut 3.0.0) Application Programming Interface Version 4.0. Disponible en: <https://freeglut.sourceforge.net/docs/api.php >.

[9] Encontrará el código a que se refiere en este artículo en el repositorio de Github <https://github.com/magusti/OpenGL_examples/GLUT_API>.