UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Introducción al uso de GLFW 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 i**nterfaz de usuario**. Si además se tiene en cuenta que esta parte es la más cambiante entre diferentes plataformas (a nivel de sistema operativo), es cuanto más aconsejable disponer de ese nivel de despreocupación de los detalles de bajo nivel que diferencian a las posibles plataformas en las que queremos desplegar la aplicación. 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 **aplicaciones interactivas e inmersivas**, con una sencilla portabilidad a otras plataformas.

En particular, esto lo podemos apreciar a la hora de desarrollar aplicaciones con **OpenGL**. En estas, el centro del interés 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 (desde 2014) nos encontramos con su propia implementación de GLUT[1].

Además de GLUT, otras alternativas se han ido desarrollando[2]. En este artículo se presenta un ejemplo de código de OpenGL con una de ellas: ***Graphics Library Framework*** (**GLFW**). Este artículo se centrará en la explicación en un ejemplo de aplicación hecha con GLUT, reescrita con GLFW, para ver los cambios típicos al utilizarla en lugar de GLUT.

2 Objetivos

Una vez que el lector se lea con detenimiento este documento y explore el código que se adjunta, podrá ver hasta qué punto GLFW nos permite realizar acciones de interfaz de usuario y, más en concreto, será capaz de:

- Explicar las posibilidades del API de GLFW.
- Explorar un ejemplo de conversión de aplicación realizada inicialmente con GLUT y que se reescribirá con GLFW.
- Poner en marcha un ejemplo básico y añadirle funcionalidades de GLFW.

[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>.

3 Introducción

OpenGL es el estándar de facto de API[3] 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 inicialización y, dado que OpenGL es independiente de plataforma, es una tarea compleja y diferente en cada plataforma que se divide en dos partes [6]:

- 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 (implementadas internamente en el hardware de la tarjeta gráfica, en un manejador genérico, por software, …) y el conjunto de las que están disponible es diferente, puesto que depende de qué versión de OpenGL esté instalada (implementada en la tarjeta internamente, por el manejador de la tarjeta gráfica o por un componente instalado en el núcleo del sistema operativo).

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 [6], se mencionan tres: GLUT, Freeglut y GLFW. En este artículos nos centramos en GLFW: una solución multiplataforma (disponible para las tres plataformas de escritorio, Windows, mac OS X y *nix) para gestión de ventanas y eventos de entrada.

GLFW es [3], Figura 1, una biblioteca escrita en C, multiplataforma y de código abierto para trabajar con OpenGL, OpenGL ES y Vulkan en computadores de escritorio (como Windows, macOS, X11 y Wayland). Ofrece un API sencillo para la creación de ventanas y contexto de renderizado y para la gestión de eventos de usuario de entrada. Su licencia es zlib/libpng .

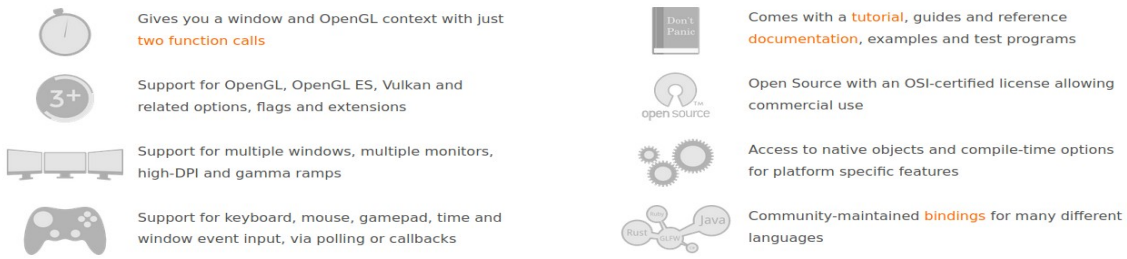

Figura 1: Características de GLFW (imagen de [3]).

4 Desarrollo

Se va a partir de un ejemplo existente, dejando de lado las operaciones propias de dibujo de OpenGL, para centrarnos en el uso de las funciones de GLUT que vamos a reescribir en términos de

[3] 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.

GLFW. Un ejemplo interesante y dinámico es el de *glutplane* (véase Figura 1), que se encuentra en [4] realizado por Mark J. Kilgard. Hay unas cuantas "*demos*" más 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 buscará qué elementos de GLUT contiene y cómo reescribirlos en términos de GLFW.

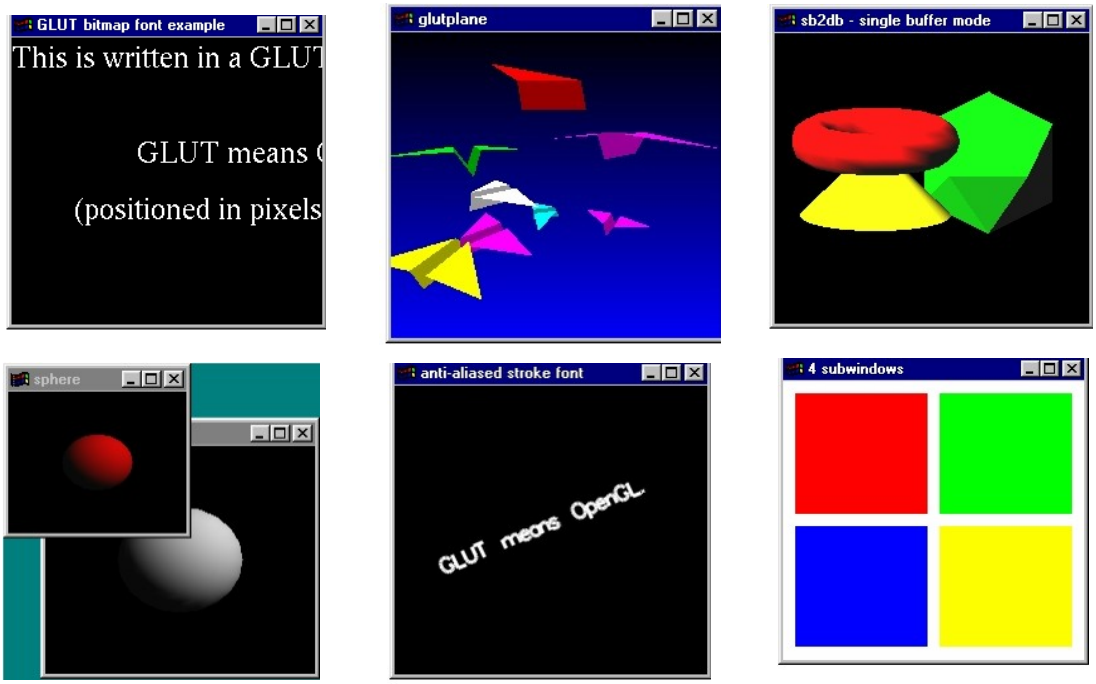

Figura 2: Algunos de los ejemplos de OpenGL con GLUT disponibles en [4] (de izquierda a derecha y de arriba abajo): bitfont, glutplane, sb2db, sphere, stroke y subwind.

El ejemplo de *glutplane* lo podemos compilar con una orden que enlace el resultado obtenido con la librería de OpenGL (*libGL*) y la de GLUT (*libGLUT*), además de la librería matemática que necesitamos (*libm*). Así que podemos hacer uso de la línea de órdenes:

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

4.1 El código

Como se ha dicho, el ejemplo escogido es *glutplane* [4] (véase Figura 3), cuyo código fuente muestra la autoría y cuya licencia permite este uso: "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.".

Veamos de forma comparada las variaciones que hemos introducido en el código original. En el repositorio[4] creado al efecto para este artículo puede encontrar esta versión y el original para compararlos uno al lado del otro. En general, se verá que han desaparecido todas las referencias con el prefijo GLUT a otras similares con GLFW como prefijo, pero veámoslas con más detalle.

En el Listado 1 se puede ver tres cosas. En primer lugar (líneas 1 a la 8), las declaraciones de la cabecera han cambiado para incluir las propias de GLFW: ha sido necesario explicitar las funciones

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

de OpenGL (con *gl.h*), que en la versión de GLUT viene dado por el propio *glut.h* y, necesariamente, la ***glfw3.h*** para declarar las interfaces de GLFW. Se ha comentado la de glad.h[5] que no se va a utilizar en este ejemplo y que se puede ver el ejemplo inicial de GLFW en [3], pero se ha respetado el uso de **GLFW_INCLUDE_NONE**[6]. En segundo lugar (líneas 11 a la 33), aparece la definición de la variable *finestraVisible* para implementar la alternativa a GLUT_VISIBLE, lo que se verá al explorar el programa principal. También aparecen las declaraciones de las interfaces de funciones que va a utilizar este ejemplo más tarde, se ha mantenido así por mantener la distribución del código del ejemplo inicial de [3]. En tercer lugar (líneas 35 a la 51), las funciones que se encargan más propiamente del dibujo no se han modificado. Por ello no se muestra el código de la función *draw*. Estrictamente, se han comentado (para que se vea explícitamente el cambio) las invocaciones a *glutSwapBuffers*, debido a la reescritura del código del bucle de procesado de eventos que se verá en la parte del programa principal. Se describirá entonces, con más detalle, esta modificación. Las demás funciones de este listado no se han modificado o se ha hecho lo mismo que en *draw*, por lo que no se insistirá. A excepción de la función *visible*, que ha quedado vacía, se verá el motivo al hablar del programa principal.

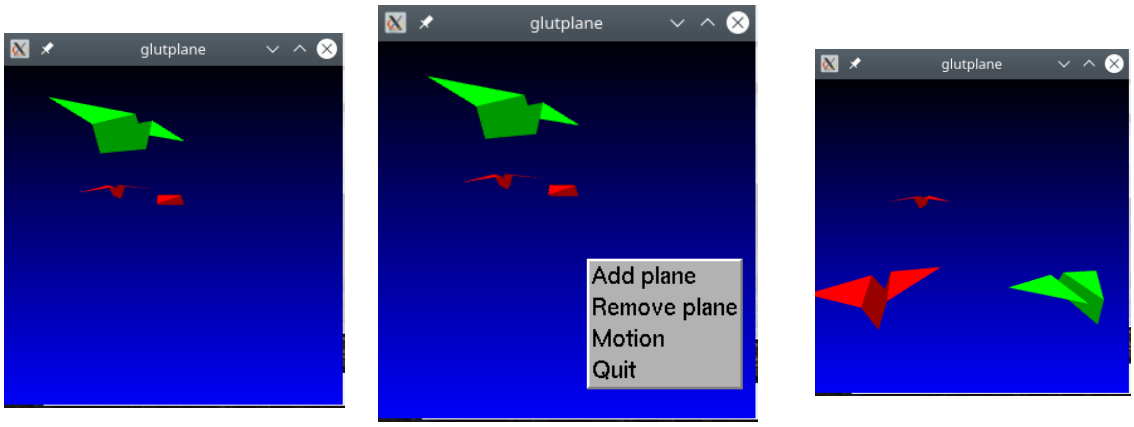

Figura 3: Salida de la versión original de glutplane: de izquierda a derecha, escena inicial, menú asociado al ratón desplegado y objetos en movimiento tras escoger "Motion" en el menú.

En el siguiente bloque de código, Listado 2, hay que destacar como modificaciones incluidas dos cosas. La primera (línea 53), que se han comentado todas las líneas que implementan la gestión de eventos de teclado en el ejemplo con GLUT (véase la función ***keyboard***). Ya que esta tarea tiene una interfaz muy diferente en GLFW y no podemos reaprovechar nada. En esta versión con GLFW todo está centralizado en una única función (***key_callback***) cuyo prototipo ya muestra las diferencias (por la cantidad y tipos de los parámetros que tiene) y de la que hablaremos en el Listado 4. Y la segunda (líneas 54 a la 74), que se han anulado (comentándolas, pero no borrándolas, para que se vea claramente dónde están los cambios) la función que manipula el menú[7] (función **menu**), que aunque no se la llama directamente, se ejecuta como respuesta a ciertos eventos en el programa principal. Así se busca reaprovechar al máximo la implementación del código original con GLUT en esta versión con GLFW.

La mayor diferencia entre ambas versiones (GLUT y GLFW) reside en que el procesado de los eventos de entrada del usuario se ha tenido que implementar explícitamente en el programa principal, así como también las funciones que gestionan cómo lanzar las animaciones (lo que debe

[5] Para más detalles sobre el papel de GLAD en OpenGL se puede consultar en *LearnOpenGL - Creating a window* <https://learnopengl.com/Getting-started/Creating-a-window>.

[6] ¿Para qué? Véalo en <https://www.glfw.org/docs/3.3/quick_guide.html >.

[7] Por cierto, estas funciones han cambiado en la versión actual de GLUT y dan un error en tiempo de ejecución que hace cerrarse la aplicación. Lo resolveremos en otro momento ;-).

modificarse automágicamente ;-) con el paso del tiempo). Se verá que esto ha sido necesario trasladarlo al programa principal por la filosofía de gestión de eventos de GLFW. Con GLFW no va se permite desplegar un menú, pero se va a mantener esta función por facilitar la comparativa y por mostrar cómo se puede reciclar al máximo el código del ejemplo original. Se va a reencaminar las acciones de teclado a esta función y, así, reaprovechar el código original.

En el Listado 3 se aborda la primera parte de los cambios que acontecen en el programa principal en dos bloques. El primero de estos bloques tiene tres etapas. La primera etapa, en las líneas 75 a la 86, muestra los cambios propios de la inicialización de GLUT frente a la de GLFW con ***glfwInit***. Se han dejado unas líneas comentadas (80 a la 83) sobre aspectos que pueden ser necesarios en aplicaciones que requieran servicios de cierta versión de OpenGL, en este caso no es necesario especificar una, debido a que lo que usamos está presente en los valores por defecto. La segunda etapa es la relativa a la creación de la ventana con ***glfwCreateWindow*** (entre las líneas 88 a la 93), esta es la acción de pedir al gestor de ventanas la inicialización de un área gráfica para la aplicación. En la línea 94, ***glfwMakeContextCurrent***, le asignamos un contexto, esto es la conexión entre el servidor OpenGL y la ventana. A partir de ahí ya podemos asignar una función (como sucede en la línea 85, ***glfwSetFramebufferSizeCallback***) que será llamada si se redimensiona la ventana. Otras funciones de gestión de eventos de la ventana no se pueden asignar, como la de dibujo (**draw** con *glutDisplayFunc*, línea 96), cuyo contenido veremos trasladado al bucle correspondiente en el programa principal. Así como, la de recepción de eventos de teclado (línea 97, *glutKeyboardFunc*) que pasa a ser ***glfwSetKeyCallback*** (la línea 98). Y, finalmente, la tercera etapa de este primer bloque, entre las líneas 99 a la 105, que muestra otras operaciones que no se puede utilizar en GLFW; en concreto la creación del menú desplegable que, ya se ha avanzado anteriormente, no es posible con GLFW y que realizaremos con la asignación de sus funciones a ciertas teclas.

El segundo bloque del Listado 3 es el que acomete el bucle de gestión de eventos que sustituye al *glutMainLoop* de GLUT (línea 116) y que ahora es un bucle que comprueba que no se ha pedido cerrar la aplicación (línea 117, ***glfwWindowShouldClose***) y, mientras, puede realizar comprobaciones como que la ventana no está oculta o minimizada (línea 118 a 119), esto es, que el usuario verá los cambios que acontecen en ella. También podría recoger los eventos de teclado (línea 120), pero se ha preferido mantener una función más similar a la de GLUT para mantener la misma forma de actuar y para explorar la mayor capacidad de esta función. Además, la fase de renderizado de OpenGL se realiza entre las líneas 122 y la 123. Y ya solo queda actualizar las áreas de dibujo que gestiona OpenGL (línea 124, ***glfwSwapBuffers***) y refrescar la cola de eventos (línea 125, ***glfwPollEvents***). Al terminar el bucle se han de liberar recursos (línea 129) con ***glfwTerminate***.

En el Listado 4 se puede ver la función de acceso a los eventos de teclado (***key_callback***), precedida por unas declaraciones de variables (líneas 134 a la 135), necesarias para la operativa de esta función. Esta función, ***key_callback,*** recibe todos eventos relacionados con el teclado: pulsaciones o liberaciones de teclas y mantenimiento de teclas pulsadas, por lo que es necesario filtrar la acción que ha desencadenado la llamada. Esto lo realiza la línea 137. Desde la línea 138 y hasta la 178, se puede observar cómo se va comprobando qué tecla ha sido pulsada y, en cada caso, se realiza la acción correspondiente. Así las acciones asociadas en diferentes teclas imitan (como se ha dicho anteriormente) las acciones que realiza el menú desplegable en la versión de GLUT, reencaminando a la función *menu*, todas las acciones que implementa y añadiendo unas nuevas (líneas 153 a la 178) que nos ofrece GLFW para ampliar en este documento las posibilidades de uso de la misma.

```
1.  #include <stdlib.h>
2.  #include <stdio.h>
3.  #include <unistd.h>
4.  #include <math.h>
5.
6.  #include <GL/gl.h> // #include <GL/glut.h>
7.  #define GLFW_INCLUDE_NONE
8.  #include <GLFW/glfw3.h>
9.  // #include <glad/gl.h>
10.
11. int finestraVisible; // per implementar GLUT_VISIBLE [Era: =
    glfwGetWindowAttrib(window, GLFW_VISIBLE);]
12.
13. void framebuffer_size_callback(GLFWwindow* window, int width, int
    height);
14. //void processInput(GLFWwindow *window);
15. void key_callback(GLFWwindow* window, int key, int scancode, int
    action, int mods);
16. void cursor_position_callback(GLFWwindow* window, double xpos, double
    ypos);
17. void mouse_button_callback(GLFWwindow* window, int button, int action,
    int mods);
18. //
19.
20.
21. int moving = GLFW_FALSE;
22.
23. // #define MAX_PLANES 15
24. #define MAX_PLANES 150
25.
26. struct {
27.   float speed;          /* zero speed2 means not flying */
28.   GLfloat red, green, blue;
29.   float theta;
30.   float x, y, z, angle;
31. } planes[MAX_PLANES];
32.
33. #define v3f glVertex3f  //v3f was the short IRIS GL name for glVertex3f
34.
35. void draw(void) {
36. …   // se omite el resto de código por que no se ha modificado
37. //   glutSwapBuffers(); --> glfwSwapBuffers(window); en el bucle ppal.
38. }
39. void tick_per_plane(int i){ ... }
40. void add_plane(void) { ... }
41. void remove_plane(void) { ... }
42. void tick(void) { ... }
43. void animate(void) { ... }
44. void visible(int state) {
45.   if (state == GLFW_TRUE) { // (state == GLUT_VISIBLE) {
46. //     if (moving)
47. //       glutIdleFunc(animate);
48. //   } else {
49. //     if (moving)
50. //       glutIdleFunc(NULL);
51.   }
52. }
...
```

Listado 1: Ejemplo de GLFW: glfwplane.c.

```
...
//void // keyboard(unsigned char ch, int x, int y) { … }
#define ADD_PLANE         1
#define REMOVE_PLANE      2
#define MOTION_ON         3
#define MOTION_OFF        4
#define QUIT              5

void menu(int item) {
  switch (item) {
  case ADD_PLANE:    add_plane();   break;
  case REMOVE_PLANE: remove_plane();   break;
  case MOTION_ON:    moving = GLFW_TRUE;
//      glutChangeToMenuEntry(3, "Motion off", MOTION_OFF);
//      glutIdleFunc(animate);
    break;
  case MOTION_OFF:   moving = GL_FALSE;
//      glutChangeToMenuEntry(3, "Motion", MOTION_ON);
//      glutIdleFunc(NULL);
    break;
  case QUIT:         exit(0);  break;
  }
}
...
```

Listado 2: Ejemplo de GLFW: glfwplane.c (2).

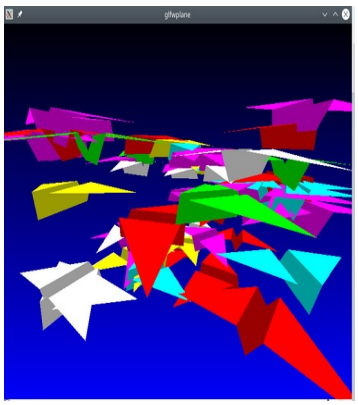

Figura 4: Salida de la versión revisada de glfwplane.

Llegados a este punto solo queda recordar que el ejemplo de GLFW se compila con una orden del estilo:

```
$ gcc glfwplane.c -o glfwplane  -lglfw  -lGL -lm
```

donde se puede ver que se han sustituido las referencias a GLUT por GLFW y que esta versión, ofrece una salida compatible con la versión de GLUT, véase la Figura 4.

```
...
int main(int argc, char *argv[]) {
//   glutInit(&argc, argv);
//   /* use multisampling if available */
//   glutInitDisplayMode(...);
  glfwInit();
  // Anem a deixar-ho per defcte
//   glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
//   glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
//   glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);
#ifdef __APPLE__
    glfwWindowHint(GLFW_OPENGL_FORWARD_COMPAT, GL_TRUE);
#endif
//   glutCreateWindow("glutplane");
  GLFWwindow* window = glfwCreateWindow(800, 600,
                                        "glfwplane", NULL, NULL);
  if (window == NULL) {
        printf("Failed to create GLFW window\n");
        glfwTerminate(); return -1;
  }
  glfwMakeContextCurrent(window);
  glfwSetFramebufferSizeCallback(window, framebuffer_size_callback);
//   glutDisplayFunc(draw);
//   glutKeyboardFunc(keyboard);
  glfwSetKeyCallback(window, key_callback);
//   glutVisibilityFunc(visible);
//   glutCreateMenu(menu);
//   glutAddMenuEntry("Add plane", ADD_PLANE);
//   glutAddMenuEntry("Remove plane", REMOVE_PLANE);
//   glutAddMenuEntry("Motion", MOTION_ON);
//   glutAddMenuEntry("Quit", QUIT);
//   glutAttachMenu(GLUT_RIGHT_BUTTON);
  /* setup OpenGL state */
  glClearDepth(1.0);
  glClearColor(0.0, 0.0, 0.0, 0.0);
  glMatrixMode(GL_PROJECTION);
  glFrustum(-1.0, 1.0, -1.0, 1.0, 1.0, 20);
  glMatrixMode(GL_MODELVIEW);
  /* add three initial random planes */
  srandom(getpid());
  add_plane();   add_plane();   add_plane();
  /* start event processing */
//   glutMainLoop();
    while (!glfwWindowShouldClose(window))     {
        finestraVisible = glfwGetWindowAttrib(window, GLFW_VISIBLE);
        visible( finestraVisible );
   //         processInput(window);
        // render!!!!
        if (moving) { animate(); //tick(); }
        draw();
        glfwSwapBuffers(window);
        glfwPollEvents();
    }
    // glfw: terminate, clearing all previously allocated GLFW
     //  resources.
    glfwTerminate();
    return 0;             /* ANSI C requires main to return int. */
}
...
```

Listado 3: Ejemplo de GLFW: glfwplane.c (3).

```
...
void framebuffer_size_callback(GLFWwindow* window, int width, int height) {...}
// int elCursor = GLUT_CURSOR_RIGHT_ARROW;
int elCursor = GLFW_ARROW_CURSOR;
int pantallaCompleta = 0;
void key_callback(GLFWwindow* window, int key, int scancode, int action, int mods) {
    if ( (action == GLFW_REPEAT) || (action == GLFW_RELEASE) ) return;
    switch ( key ) {
       case GLFW_KEY_ESCAPE:
        glfwSetWindowShouldClose(window, GL_TRUE); //1); //true);
        break;
       case GLFW_KEY_SPACE:  tick();
//             glutPostRedisplay();
        break;
       case GLFW_KEY_A: menu( ADD_PLANE ); break;
       case GLFW_KEY_R: menu( REMOVE_PLANE ); break;
       case GLFW_KEY_M:
        moving = !moving;
        if (moving) menu( MOTION_ON ); else menu( MOTION_OFF ); break;
       case GLFW_KEY_H:
        printf("Ajuda\n\ ...        ESC\t finaliza la aplicación \n");
        break;
       case GLFW_KEY_F:
//      glutFullScreenToggle(); // freeglut v4
        pantallaCompleta = !pantallaCompleta;
        if ( pantallaCompleta )  glfwMaximizeWindow(window);
         else   glfwRestoreWindow(window);
//     glutPostRedisplay();
        break;
       case GLFW_KEY_C:
        if ( elCursor == GLFW_VRESIZE_CURSOR + 1  ) {
         glfwSetCursor(window, NULL); elCursor = GLFW_ARROW_CURSOR; }
         else {
          GLFWcursor *cursor = glfwCreateStandardCursor( elCursor );
         glfwSetCursor(window, cursor ); elCursor++;
        }
//     glutPostRedisplay();
        break;
       case GLFW_KEY_S: //     glutShowWindow();
        glfwShowWindow( window ); break;
       case GLFW_KEY_D:  //     glutHideWindow();
        glfwHideWindow( window ); break;
       case GLFW_KEY_W:  //     glutIconifyWindow();
        glfwIconifyWindow( window ); // glfwRestoreWindow(window);
        break;
       case GLFW_KEY_T: //  glutSetWindowTitle( "Canviant el títol!");
        glfwSetWindowTitle(window, "Canviant el títol");   break;
  }
}
```

Listado 4: Ejemplo de GLFW: glfwplane.c (y 4).

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 GLFW, a partir de un ejemplo básico existente al que se le han sustituido las funcionalidades de GLUT por las de GLFW. ¡Y hemos ampliado el número de aviones, véase la Figura 4!

Hay una funcionalidad que se hecha de menos en GLFW: no es posible desplegar un menú. Pero quizá lo compensa la forma de gestionar el flujo de eventos que es más obvia para algunos desarrolladores y, en concreto, para adaptarlo a plataformas como las videoconsolas, que son muy particulares en estas acciones y que no suelen adaptar la forma de mostrar el menú a cada videojuego o aplicación.

Quiero aprovechar para animar al lector a probar alguna que otra función de GLFW de las que están recogidas en la documentación en línea, como las de los apartados *Text Input*, *Time input*, *Clipboard input and output* y *Path drop input*[8].

Espero que, a estas alturas, tenga ejecutándose el código propuesto y comprobando que puede ver volar los aviones de papel. Puede descargar el ejemplo del repositorio creado a tal efecto en *GitHub*[9] y no cierre este el documento sin haberlo comprobado antes. ¡¡ÁNIMO!!

6 Bibliografía y referencias

[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] Sitio web de GLFW. Disponible en: <https://www.glfw.org/>

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

[8] Las encontrará en <https://www.glfw.org/docs/3.3/input_guide.html >.

[9] Encontrará el código a que se refiere en este artículo en el repositorio de Github <https://github.com/magusti/OpenGL_examples/GLFW_API>, como ya se ha mencionado anteriormente.