Administración y Organización de Computadores

Curso 2020-2021

Práctica: *Painter: herramienta de dibujo*

El principal objetivo de esta práctica es completar una herramienta de dibujo, escrita en *C++* sobre Linux, a través de la implementación en ensamblador de varias de las operaciones que debe proporcionar el programa. La integración de los dos lenguajes se llevará a cabo mediante las facilidades de ensamblado en línea proporcionadas por el compilador *gcc*. La programación en ensamblador se realizará considerando la arquitectura de un procesador Intel o compatible de 64 bits.

A continuación, se detallan diferentes aspectos a tener en cuenta para el desarrollo de esta práctica.

Descripción de la aplicación

El código que deberá ser completado estará incluido en una aplicación *C++*, disponible como proyecto de Qt. La siguiente figura muestra una captura de pantalla en un instante de ejecución de la aplicación:

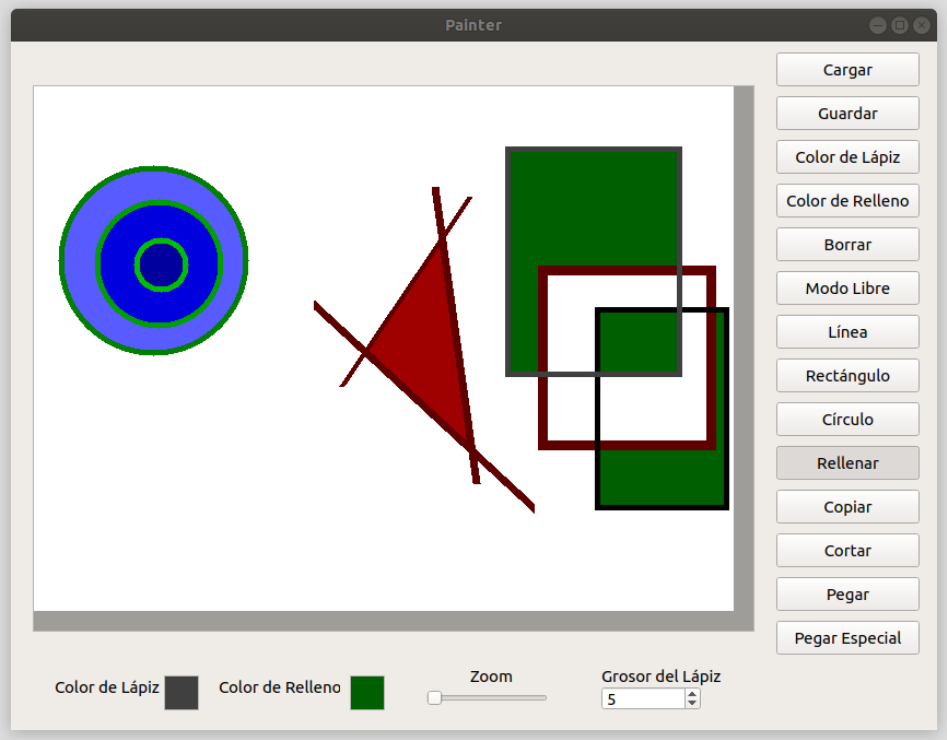

Figura 1: interfaz principal de la aplicación

¿Qué es Qt?

Qt es un marco de desarrollo de aplicaciones que, entre otras aportaciones, proporciona herramientas y librerías de clases para la creación de interfaces de usuario en entornos de escritorio.

Puesta en marcha del proyecto

El fichero que contiene la descripción del proyecto (*pracaoc.pro*) se encuentra disponible en la carpeta principal de la aplicación. Dicho fichero puede ser utilizado para importar el proyecto desde diferentes entornos de desarrollo. No obstante, se recomienda trabajar con Qt Creator (paquete *qtcreator*). Es necesario instalar además los paquetes *qt5-default* y *qttools5-dev-tools*.

Funcionamiento de la aplicación

La interfaz principal (figura 1) está compuesta por una ventana de dibujo y una serie de botones a través de los cuales es posible ejecutar las diferentes operaciones que proporciona la aplicación. El panel de dibujo tiene un tamaño de 640x480 píxeles y constituye el elemento editable de la aplicación. Es decir, es sobre esta ventana sobre la que el usuario aplica las distintas operaciones de dibujo disponibles. Además de las opciones para incluir distintos elementos de un dibujo en la ventana de edición, la aplicación cuenta con opciones que permiten ampliar la vista del panel (*zoom*), cambiar el grosor del lápiz utilizado para dibujar círculos, líneas, etc., modificar el color de dicho lápiz o modificar el color de relleno asociado con las operaciones "*Borrar*" y "*Rellenar*". Se describen a continuación, con más detalle, cada una de las opciones incluidas en el programa:

- ***Cargar***: carga, en la ventana de dibujo, la imagen del fichero indicado. Si la imagen del fichero tiene una resolución superior a la soportada por el programa, ésta es recortada.
- ***Guardar***: guarda el dibujo actual en el fichero indicado con el formato de imagen especificado.
- ***Color de Lápiz***: permite al usuario cambiar el color de lápiz utilizado en las opciones "*Modo Libre*", "*Línea*", "*Rectángulo*" y "*Círculo*"
- ***Color de Relleno***: permite al usuario cambiar el color de relleno utilizado en las opciones "*Borrar*" y "*Rellenar*".
- ***Borrar***: borra el contenido de la ventana de dibujo dándole a cada uno de sus píxeles el color de relleno actual .
- ***Modo Libre***: cuando este modo está seleccionado, el puntero del ratón funcionará como un lápiz sobre la ventana de dibujo. Cuando uno de los botones del ratón se encuentre pulsado, la zona del dibujo situada bajo el puntero tomará el color indicado en "*Color de Lápiz*". El alto y ancho en píxeles de dicha zona será el indicado en "*Grosor de Lápiz*".
- ***Línea***: cuando esta opción esté seleccionada, el usuario podrá dibujar líneas sobre la ventana de dibujo con el grosor y color de lápiz actual. Para ello, deberá indicar los extremos del segmento de línea que desea dibujar trazando dicho segmento con el ratón mientras se mantiene pulsado uno de sus botones. Los puntos indicados al pulsar y soltar el ratón sobre la ventana de dibujo constituirán los extremos del segmento.
- ***Rectángulo***: la selección de esta opción permite al usuario dibujar rectángulos con el color y grosor de lápiz actual. Sólo dibuja los bordes del rectángulo, sin modificar el color de su interior. Para indicar la posición y dimensiones del rectángulo, el usuario marcará con el ratón la zona afectada en la ventana de dibujo.

- ***Círculo***: permite dibujar el borde de un círculo con el color y grosor de lápiz actual. Cuando esta opción esté seleccionada, el usuario deberá indicar el centro y radio del círculo pinchando y arrastrando el ratón sobre la ventana de dibujo hasta marcar la zona afectada. El punto indicado al iniciar la pulsación constituirá el centro de la circunferencia. El radio se obtendrá como la distancia entre los puntos indicados al pulsar y soltar el ratón.
- ***Rellenar***: rellena la zona de la ventana de dibujo indicada expandiendo el color de relleno hasta encontrar bordes que la delimiten. Cuando esta opción se encuentre seleccionada, el usuario marcará con el ratón un píxel de la zona interior del objeto que desea rellenar. A partir de la posición indicada, el color de relleno se extenderá por todos los píxeles del mismo color que el seleccionado. El efecto resultante en zonas uniformes delimitadas por cualquier tipo de borde cerrado será el relleno completo de su interior.
- ***Copiar***: copia el área de la ventada de dibujo seleccionada en un "porta-papeles" interno de la aplicación. Para poder seleccionar una zona, las restantes opciones deberán estar desactivadas.
- ***Cortar***: funciona igual que la opción anterior pero, en este caso, borra la zona seleccionada coloreando sus píxeles con el color de relleno actual.
- ***Pegar***: pega el contenido del "porta-papeles" en la zona del panel de dibujo indicada por el usuario. Una vez seleccionada esta opción, el usuario deberá marcar con el ratón la esquina superior izquierda del área de la ventana donde se copiará el contenido del "porta-papeles".
- ***Pegar Especial***: funciona igual que la opción anterior pero, en este caso, sólo se copian los píxeles del "porta-papeles" cuyo color es distinto del color de relleno actual. Esto equivale a considerar el color de relleno como transparente, permitiendo mantener el fondo original al superponer un área del dibujo sobre otra zona previamente dibujada.

Componentes de la aplicación

El código fuente de la aplicación está formado por 5 ficheros: *main.cpp*, *pracoc.cpp*, *pracoc.h*, *painter.cpp* y *painter.h*. Además, se incluye un formulario de Qt (*mainForm.ui*) y el fichero de descripción del proyecto (*pracaoc.pro*). El contenido de cada fichero fuente es el siguiente:

- *main.cpp*: contiene el procedimiento principal que permite lanzar la aplicación, así como crear y mostrar la ventana principal que actúa como interfaz entre el usuario y la aplicación.
- *pracaoc.h*: fichero que contiene la definición de la clase principal de la aplicación (*pracAOC*). Esta clase contiene los elementos principales de gestión de la aplicación. Entre los atributos, se encuentran la interfaz de usuario incluida en el programa y la variable que permite almacenar el contenido del área de dibujo. Esta última (*img*) está definida como un array de tipo uchar (unsigned char) de 640x480 elementos (640 columnas y 480 filas) asociados con los distintos píxeles del dibujo. Cada uno de sus elementos almacena un índice de color cuyos posibles valores están comprendidos en el rango entre 0 y 255. Dicho índice no es más que una entrada de una tabla de definición de colores que está asociada a la imagen del panel de dibujo. Al cambiar el índice de color de algún elemento del array, el resultado es la variación del color del píxel correspondiente del dibujo. La imagen dentro de este array se encuentra almacenada por filas. Esto implica que el acceso a un determinado píxel situado en una fila *f* y una columna *c* de imagen se realiza a través de la posición del array *f*640+c*.
- *pracaoc.cpp*: Incluye la implementación de los métodos de la clase *pracAOC*. En su mayoría, estos métodos se encargan de responder a los distintos eventos de la interfaz

de usuario incluida en el programa y de llamar a las funciones de dibujo que correspondan en cada caso (disponibles en *painter.cpp y painter.h*).

- *painter.h*: contiene la definición de las funciones implementadas en el fichero painter.*cpp*.
- *painter.cpp*: implementación de las funciones de dibujo que se ejecutan a través de las distintas opciones de la aplicación. La mayoría de estas funciones contienen una implementación vacía. El objetivo de esta práctica es completarlas para que el funcionamiento de la aplicación sea el descrito anteriormente.

Extensiones principales en x86-64

- En relación a la representación de datos en memoria, en 64 bits, los punteros y los datos de tipo *long* ocupan 64 bits. El resto de tipos mantiene el mismo tamaño que en la línea de procesadores de 32 bits (int: 4 bytes, short: 2 bytes, …).
- Todas las instrucciones de 32 bits pueden utilizarse ahora con operandos de 64 bits. En ensamblador, el sufijo de instrucción para operandos de 64 bits es "q".
- Los registros de 32 bits se extienden a 64. Para hacer referencia a ellos desde un programa en lenguaje ensamblador, hay que sustituir la letra inicial "e" por "r" (%rax, %rbx, …). Los registros de 32 bits de la IA32 se corresponden con los 32 bits de menor peso de estos nuevos registros.
- El byte bajo de los registros %rsi, %rdi, %rsp y %rbp es accesible. Desde el lenguaje ensamblador, el acceso a estos registros se realiza a través del nombre del registro de 16 bits finalizado con la letra "l" (%sil, %dil, …).
- Aparecen 8 nuevos registros de propósito general de 64 bits (%r8, %r9, …, %r15). Es posible acceder a los 4, 2 o al último byte de estos registros incluyendo en su nombre el sufijo d, w o b (%r8d – 4 bytes, %r8w – 2 bytes, %r8b – 1 byte).

Ejemplo de implementación en ensamblador x86-64 de una función de C/C++

Dentro del fichero *painter.cpp*, se ha incluido la implementación del procedimiento *modoLibre*. Dicha implementación se describe a continuación.

```
void painter::modoLibre(uchar * img, int cI, int fI, int grosor, uchar color)
{
   asm volatile(
      "mov %0, %%rsi ;"
      "movsxd %1, %%rbx ;"
      "movsxd %2, %%rax ;"
      "movsxd %3, %%rcx ;"
      "mov %4, %%dl ;"
      "imul $640, %%rax ;"
      "add %%rbx, %%rax ;"
      "add %%rax, %%rsi ;"
      "mov $0, %%rax   ;"
"bModoLapizFilas:"
      "mov $0, %%rbx   ;"
   "bModoLapizColumnas:"
         "mov %%dl, (%%rsi, %%rbx) ;"
         "inc %%rbx ;"
         "cmp %%rcx, %%rbx ;"
         "jl bModoLapizColumnas ;"
      "add $640, %%rsi ;"
      "inc %%rax ;"
      "cmp %%rcx, %%rax ;"
      "jl bModoLapizFilas ;"
```

```
        :
        : "m" (img), "m" (cI), "m" (fI), "m" (grosor), "m" (color)
        : "%rax","%rbx", "%rcx", "%rdx", "%rsi", "memory"
    );
}
```

El procedimiento *modoLibre* es invocado por la opción *Modo Libre* cada vez que el usuario pulsa con el ratón sobre el panel de dibujo. Dicho procedimiento se encarga de asignar a los píxeles de la zona indicada por el usuario, teniendo en cuenta grosor de lápiz actual, el color de lápiz actual. Para ello, recibe por parámetro un puntero al bloque de memoria donde está almacenada la imagen del área de dibujo (*img*), la columna (*cI*) y fila (*fI*) del píxel indicado por el usuario, el grosor actual (*grosor*) y el índice del color de lápiz actual (*color*). Estos parámetros están incluidos como operandos de entrada en el bloque de código ensamblador ("*: "m" (img), "m" (cI), "m" (fI), "m" (grosor), "m" (color)"*). Para utilizar los operandos dentro del código ensamblador se deben usar las referencias %0, %1, …, teniendo en cuenta el orden en el que aparecen en la lista. Así, el parámetro *img* sería accesible a través de la referencia %0, el parámetro *cI* a través de %1 y así sucesivamente. Es importante tratar estar referencias teniendo en cuenta el tamaño del parámetro asociado. Por ejemplo, %0 debe tratarse con 64 bits puesto que es un puntero. %1 debe utilizarse con 32 bits, dado que el parámetro *cI* es de tipo *int*.

Tras la definición de los operandos, se incluye la lista de registros utilizados dentro del código. Además de los registros indicados, dado que la memoria es modificada, la lista incluye también la palabra "memory".

Una vez aclaradas estas definiciones, analicemos a continuación paso a paso el código ensamblador incluido en el procedimiento:

- La primera instrucción se encarga de almacenar la dirección inicial en memoria del panel de dibujo (%0 = *img*) en el registro %rsi para su posterior direccionamiento.
- A continuación, se cargan la columna y fila inicial (%1 = *cI*, %2 = *fI*) en los registros %rax y %rcx. Para realizar esta carga, puesto que los registros son de 64 bits y los operandos %1 y %2 son de 32 bits, es necesario utilizar la instrucción *movsxd* que permite hacer una transferencia con conversión de tamaño.
- La siguiente instrucción carga el valor del grosor (%3 = grosor) en el registro %rcx realizando una conversión de tamaño al igual que las 2 instrucciones previas.
- La quinta instrucción almacena el valor de color (%4 = color) en el registro %dl para poder asignarlo posteriormente a cada una de las posiciones del panel de dibujo indicadas por los otros parámetros.
- Tras todas estas asignaciones, se calcula la posición lineal del píxel inicial de la zona indicada por el usuario. Dicha posición se corresponde con la expresión *fI*640+cI* y se resuelve a través de las dos siguientes instrucciones. Una vez calculada, dicha posición se añade a la dirección inicial del panel de dibujo (%rsi) para poder realizar posteriormente la asignación del color a partir de esa posición de memoria.
- A continuación, se incluyen 2 bucles anidados que permiten recorrer un área cuadrada del panel de dibujo definida por el valor del grosor. El bucle más externo realiza un recorrido por filas, controlado por el registro %rax. El más interno recorre el área por columnas utilizando el registro %rbx. Ambos recorridos finalizan cuando el índice de fila (%rax) o el de columna (%rbx) coinciden con el valor del grosor (%rcx). En el bucle más interno, se realiza la asignación del color (%dl) al píxel actual, cuya posición viene determinada por la dirección del píxel inicial (%rsi) y la columna actual (%rbx). Tras recorrer todas las columnas, la posición del píxel inicial se actualiza sumando 640 (ancho del panel de dibujo) a la dirección actual (%rsi). De esta forma, en el siguiente recorrido por columnas, las posiciones modificadas se corresponden con la siguiente fila. Una vez recorrida toda el área afectada, el procedimiento finaliza.

Especificación de los objetivos de la práctica

El principal objetivo de esta práctica es completar el código de la aplicación descrita para que su funcionamiento sea el que se detalla en la sección "*Funcionamiento de la aplicación"*, incluida en esta documentación. Para ello, se deberán implementar, en lenguaje ensamblador, los procedimientos vacíos del módulo "painter.cpp". Cada uno de estos procedimientos está asociado con la opción del programa del mismo nombre. Para cada uno de ellos, se proporciona la estructura inicial del bloque ensamblador, en la que se ha incluido la definición de operandos que afectan a la implementación. La lista de registros utilizados incluye únicamente la palabra "memory", ya que en todos los casos la memoria es modificada. La inclusión de registros dentro de esta lista dependerá de la implementación que se desarrolle en cada caso, por lo que, será necesario completarla para cada uno de los procedimientos.

Se describe a continuación la funcionalidad de cada uno de los procedimientos a completar. Para todos ellos, el parámetro *img* contiene la dirección del bloque de memoria donde está almacenada la imagen del área de dibujo.

- *void* ***borrar****(uchar* **** img****, uchar* ***color)***: asigna a todos los píxeles de *img* (640*480) el color de relleno actual indicado por *color*.
- *void* ***linea****(uchar* **** img****, int* ***c1****, int* ***f1****, int* ***c2****, int* ***f2****, int* ***grosor****, uchar* ***color)***: dibuja el segmento de línea indicado por los puntos (*f1*, *c1*) y (*f2*, *c2*) con el color y grosor especificados. Para ello, asigna el color indicado como parámetro a los píxeles de *img* asociados con las posiciones del segmento. Sin considerar el grosor, dichas posiciones deben estar comprendidas entre (*f1*, *c1*) y (*f2*, *c2*) y cumplir además la siguiente relación:

  $$(c\text{-}c1)/(f\text{-}f1) \approx (c2\text{-}c1)/(f2\text{-}f1)$$

  El símbolo "≈" indica que la relación no es exacta, sino sólo aproximada. No existe más remedio que hacer esta consideración ya que las posiciones de los píxeles son valores discretos y es, por lo tanto, necesario operar con magnitudes enteras.

- *void* ***rectangulo****(uchar* **** img****, int* ***c1****, int* ***f1****, int* ***c2****, int* ***f2****, int* ***grosor****, uchar* ***color)***: dibuja los bordes del rectángulo especificado con el color y grosor indicados. El rectángulo a dibujar estará definido por su esquina superior izquierda, indicada en *c1* y *f1*, y por su esquina inferior derecha, almacenada en *c2* y *f2*. Al igual que en el procedimiento anterior, el dibujo del rectángulo se resolverá asignando el color indicado como parámetro a los píxeles de *img* asociados con las posiciones afectada. Sin tener en cuenta el grosor, estas posiciones cumplen cualquiera de las 4 relaciones siguientes :

  $$\begin{array}{llll} c1 \le c \le c2 & & & f=f1 \\ c1 \le c \le c2 & & & f=f2 \\ c=c1 & & f1 \le f \le f2 & \\ c=c2 & & f1 \le f \le f2 & \end{array}$$

- *void* ***circulo****(uchar* **** img****, int* ***c_c****, int* ***f_c****, int* ***r****, int* ***grosor****, uchar* ***color)***: dibuja el borde del círculo de centro (*f_c*, *c_c*) y radio *r* con el color y grosor indicados. Sin tener en cuenta el grosor, las posiciones afectadas deberán cumplir la siguiente relación:

  $$\text{sqrt}((c\text{-}c\_c)^2 + (f\text{-}f\_c)^2) = r$$

- *void* ***rellenar****(uchar* **** img****, int* ***c****, int* ***f****, uchar* ***color)***: asigna, de manera recursiva, el color indicado en el parámetro *color* al píxel situado en la posición (*f*, *c*) y a cada uno de los píxeles conectados a él cuyos colores coincidan con el de dicha posición. La

solución recursiva de este procedimiento permite expandir el color especificado hasta encontrar bordes que delimiten la zona indicada. Dicha solución se facilita en el pseudocódigo que se incluye junto con este documento.

- *void* ***copiar****(uchar * **img**, uchar * **buffer**, int **c**, int **f**, int **w**, int **h**)*: copia la ventana de *img* especificada por su esquina superior izquierda (*f*, *c*), con ancho *w* y alto *h*, a la variable *buffer* (puntero al porta-papeles interno de la aplicación).
- *void* ***cortar****(uchar * **img**, uchar * **buffer**, int **c**, int **f**, int **w**, int **h**, uchar **cFondo**)*: funciona igual que el procedimiento *copiar*, pero, además, tras la copia de la ventana de imagen en el *buffer*, borra el contenido de dicha ventana asignando a todos sus píxeles el color indicado en *cFondo*.
- *void* ***pegar****(uchar * **img**, uchar * **buffer**, int **c**, int **f**, int **w**, int **h**)*: copia el contenido de la variable *buffer* en la ventana de *img* indicada por su esquina superior izquierda (*f*, *c*), con ancho *w* y alto *h*. El procedimiento deberá controlar que la ventana especificada no exceda los límites de la imagen, recortando, si fuera necesario, el trozo de imagen a pegar.
- *void* ***pegarEspecial****(uchar * **img**, uchar * **buffer**, int **c**, int **f**, int **w**, int **h**, uchar **color**)*: funciona igual que el procedimiento anterior, pero, en este caso, los píxeles del *buffer* que coincidan con el color indicado no deberán copiarse a la posición correspondiente de *img*.

La implementación de los procedimientos *circulo* y *rellenar* será opcional. La nota máxima que podrá obtenerse sin la realización de dichos procedimientos será de NOTABLE(8).

Nota: la práctica se realizará de manera individual.
Fecha de entrega de la práctica: 25/01/2021 hasta las 14:00
Entrega: la entrega se realizará a través de la subida al aula virtual de la asignatura de un archivo .zip que contenga el proyecto completo.