CLICKS

CORBALAN CEREZUELA, JUAN JOSE

Famoso y sencillo juego de tablero. Se basa en un tablero de fichas (en este caso bolas o esferas) de distintos colores que se irán eliminando al pinchar sobre ellas con el ratón.

La condición para que las bolas se vayan eliminando es que al menos dos del mismo color sean adyacentes, así, cualquier conjunto de dos o más bolas del mismo color adyacentes todas ellas se eliminará al pinchar sobre una de ellas. El resto de las bolas se recolocan desplazándose verticalmente y hacia abajo ocupando los lugares antes ocupados por las bolas eliminadas. También, cuando una columna queda vacía por completo todas las bolas que queden a la derecha de dicha columna se desplazan una posición hacia la izquierda rellenando la columna vacía y dejando el tablero desocupado a la derecha.

MANUAL DE USO

Una vez compilado el fichero fuente ("clicks.c") y suponiendo que el ejecutable lleva el mismo nombre bastará con teclear "./clicks" en la línea de comandos de una ventana de consola.

Tras ejecutarlo aparece una ventana conteniendo unas líneas introductorias del juego e informando del modo de acceso al menú.

Control

El control del juego se realiza totalmente con el ratón.
Para acceder al menú bastará pulsar el botón derecho del ratón sobre la ventana ("tablero") del juego, de esta forma aparecerá éste y solo será necesario desplazar el puntero del ratón sobre la opción deseada y soltar el botón pulsado (en este caso el derecho).

Menú

Éste consta de 6 opciones distintas:

-Nueva partida: hace desaparecer la pantalla de bienvenida y muestra el tablero del juego que inicialmente estará completamente ocupado por bolas de distintos colores. El tablero es distinto cada vez al crearse aleatoriamente.

-Cargar partida: recupera una partida guardada en un fichero creado automáticamente y llamado "savegame.txt". Esto implica el descarte de la partida actual (si es que hay una partida iniciada) y continuar la partida guardada.
Si por alguna razón no se puede abrir el fichero "savegame.txt" o no existe (por no haberse guardado ninguna partida) se muestra un mensaje de error en la consola.

-Guardar partida: Operación opuesta a la anterior. En cualquier momento del juego invocando esta opción se guarda el tablero actual en el fichero "savegame.txt" de forma que posteriormente pueda ser recuperada la partida. Si el fichero ya existe se descarta su contenido.
Igualmente si por alguna razón no se puede crear el fichero se muestra un error en la consola pero se puede continuar el juego.

-Deshacer: deshace el último movimiento realizado, volviendo a dibujar el tablero en su posición anterior.

-Ayuda: muestra la misma pantalla de bienvenida que contiene información sobre las reglas del juego.

-Salir: termina el juego, cerrando la ventana del mismo.

Partida

Una partida comienza con el tablero completo y se van eliminando bolas. Cuando se eliminan todas las bolas se muestra una pantalla similar a la de ayuda dando la enhorabuena por haber conseguido el objetivo del juego, pudiento entonces comenzar otra partida.
Si no se han eliminado todas las bolas pero las que quedan de distintos no son adyacentes entre sí por colores, no quedarán más movimientos posibles, hecho que hace que se muestre también un mensaje informativo.

DOCUMENTACION

La documentación de este programa no resulta demasiado extensa ya que no se trata de una aplicación muy compleja.

Se ha programado haciendo uso de las librerías gráficas OpenGL.

Así, en esta documentación se presenta como un breve comentario de cada una de las funciones y procedimientos contenidos en el único archivo fuente del programa "clicks.c"

DECLARACIONES E INCLUDES

Se incluyen las librerías estandar "stdlib.h" y "stdio.h", imprescindibles en cualquier programa en C.
Se incluye tambien "glut.h", imprescindible en este caso al tratarse de una aplicación OpenGL.
Y por último se incluye "time.h" ya que será necesario utilizar "time()" para obtener el tiempo del sistema y así generar un conjunto números (utilizados para diferenciar los distintos colores de las bolas) distinto en cada ejecución.

Se definen entonces un conjunto de variables globales necesarias a lo largo de todo el programa

*fichas y deshacer: matrices de dos dimensiones que almacenan los distintos valores para cada una de las posiciones del tablero y que sirven para identificar si esta libre=0, y un valor distinto de 0 para indicar los colores de las bolas (rojo=1, verde=2, azul=3, amarillo=4). "Deshacer" será una copia de "fichas" en cada movimiento para poder deshacer la última jugada.

*ayuda: almacena distintos valores indicando si se muestra el tablero, la ayuda, el mensaje de no mas movimientos o el mensaje de todas las bolas eliminadas.

*bolas y bolas2: con un valor inicial en cada partida de 180 y almacena precisamente eso, el número de bolas que quedan en el tablero en cada instante (bolas2 será una copia del primero en cada movimiento y utilizado para deshacer la última jugada).

*menu: identificador del menú.

*la enumeración opciones_menu: con todas las opciones del menú.

*la estructura punto: que consta de dos enteros que serán los índices de una bola de la pantalla y será utilizada en vector (ver siguiente)

*vector: vector unidimensional de estructuras "punto" que se utiliza para almacenar en cada posición las coordenadas de una bola a eliminar. En su momento se recorre poniendo a 0 en la matriz "fichas" cada una de las celdas indicadas en cada posición de "vector", con lo que se eliminan las bolas al repintar el tablero.

int main(int argc, char *argv[])
****
Función principal que realiza todas las operaciones necesarias para el correcto funcionamiento como son la inicialización de glu y Mesa, generar las fichas del tablero incial (se verá mas adelante), crear la ventana y establecer sus propiedades, establecer las funciones de redimensionado y dibujo, crear el menu activar la eliminación de caras ocultas y el test de profundidad y la iniciacion del bucle principal.

void creaMenu()
****
Sencilla función que como su nombre indica crea el menú en base al identificador "menu" declarado globalmente y se establecen todas las entradas del menú, asignandole también el botón derecho del ratón como método de apertura.

void opcionMenu(int opcion)
****
Realiza las opciones oportunas según la opción de menú seleccionada indicada en el parámetro "opcion".

*Si se ha pulsado la opción de nueva partida: se pone "ayuda" a 0, indicando que debe mostrarse el tablero de juego, se llama a "generaFichas()" que rellena con valores aleatorios la matriz "fichas" (creando así un nuevo tablero de juego), "bolas" se iguala a 180 (número inicial de bolas) y se redibuja.

*Cargar partida: se intenta abrir el fichero "savegame.txt" como lectura (si no se puede se muestra un mensaje de error) y se continua el juego. Si se pudo abrir pada cada posición de "fichas" se lee un carácter del fichero (ya que aunque sean enteros del fichero solo hay que leer un dígito cada vez) y tras pasarlo a entero se asigna a la posición adecuada de fichas (estos valores también se almacenan en "deshacer"). Por último se lee un entero (ya puede que tenga más de un digito) que será el número de bolas y se asigna tanto a "bolas" como a "bolas2". Se cierra el fichero. Con esto se han recuperado los valores de la partida guardada y por último se redibuja.

*Guardar partida: operación contraria a la anterior. En este caso se abre el fichero para escritura y se escribe una a una cada da una de las posiciones de "fichas" y por último el número de bolas almacenado en la variable "bolas".

*Deshacer: simplemente se asignan los valores almacenados en la matriz "deshacer" a las posiciones de la matriz "fichas", y el valor de "bolas2" al de "bolas", con lo que se recupera el estado anterior antes de la última jugada. Redibujando se habrá deshecho la última jugada.

*Salir: simplemente sale de la aplicación.

void genereaFichas()
****
Construye un tablero nuevo. Esta operación consiste en generar para cada posición de "fichas" un valor aleatorio entre 1 y 4 (ambos inclusive) correspondientes a cada uno de los cuatro posibles colores de las bolas del juego.

void printString(void *font, char *string)
****
Escribe "string" carácter a carácter con la fuente "font" haciendo uso de glutBitmapCharacter.

void printhelp(void)
****
Dependiendo del valor de "ayuda" (0->muestra ayuda, 2->muestra mensaje de enhorabuena, 1->no quedan mas movimientos) escribe en pantalla diversos mensajes haciendo uso de la función anterior.

int masMovimientos(vois)
****
Comprueba si quedan mas movimientos posibles o no. Si quedan movimientos devuelve 1 y si no devuelve 0. La operación se realiza comprobando para cada posición de "fichas" (para cada bola) las posiciones superior, izquierda abajo y derecha (siempre que no se salga de rango), si alguna de dichas posiciones coincide en valor numérico (bola del mismo color) a la posición base de la comprobación es que quedan al menos dos bolas adyacentes del mismo color y por tanto queda al menos 1 movimiento, en ese caso de devuelve 1 y no se continua con el bucle. Si el bucle termina si que se cumpla nada de lo anterior es que no quedan movimientos posibles y se devuelve 0.

void reajustaTablero(void)
****

Función utilizada después de cada jugada y cuya misión es primero desplazar las bolas verticalmente para que ocupen los lugares dejados por las eliminadas (estas seran las posiciones en la matriz "fichas" con valor 0).
Y posteriormente realizar una operación similar para desplazar todas las filas horizontalmente y hacia la izquierda en caso de que haya/n quedado una/s columna/s vacías completamente.
La operación se realiza sencillamente mediante comparaciones dentro de bucles for que recorren la matriz "fichas" por columnas o filas según el caso y no reviste más complejidad.

void display(void)
****

Función de dibujo. Hace uso de otra dos funciones construidas para que ésta no resulte demasiado extensa.
En primer lugar se vacían los bufferes de color y de profundidad.
Si ayuda es igual 0 habrá que dibujar el tablero, por lo que se llama a "dibujaTablero" que realiza dicha operación.
Si ayuda tiene un valor distinto de 0, será que hay que mostrar la ayuda o cualquiera de los otros mensajes y para ello se desactivan las luces, y se llama a "printhelp" (ya descrita anteriormente) que realiza la operación oportuna.

void dibujaTablero(GLenum mode)
****

Esta función es la que dibuja el tablero de juego. Recibe el modo de dibujo como parámetro.
En primer lugar se definen el brillo y posición de la luz, y los distintos "materiales" (que en nuestro caso darán lugar a distintos colores) para las bolas del tablero.
Posteriormente se establecen parámetros como la posicion de la luz, la activación de la misma, etc.
Ahora, mediante dos bucles for anidados se recorre cada una de las posiciones de la matriz "fichas" y según el valor almacenado se establecen los parámetros del "material" a utilizar (1->dara lugar a una bola roja, 2-> verde, 3->azul, 4->amarillo), valor seleccionado mediante un switch.
Una vez hecho esto y si la posición actual consultada es distinta de 0 (hay que dibujar la bola) se desplaza a la posición adecuada y mendiante "glutSolidSphere" se apila en la matriz de transformaciones la bola en la posición y color deseados.
En cada ciclo de ambos bucles se usan dos variables "n" y "m" que se van actualizando y que servirán para el posicionamiento adecudo en el modelo.
También dentro de los bucles y si el modo de dibujo (pasado como parámetro) es el de selección, se van apilando dos nombres para cada esfera (dos ya que habrá que saber diferenciar la fila y la columna pinchadas posteriormente).

void reshape(int w, int h)
****

Función de redimensionado de la ventana.
Simplemente y en función de "w" y "h" (anchura y altura respectivamente) varía la vista paralela creada para visualizar el tablero en modo de proyeccion.

void pick(int boton, int estado, int x, int y)
****

Función que controla la pulsación del ratón y por tanto la selección sobre el modelo de las bolas del tablero.
Ésta es análoga a la ya vista por ejemplo en la práctica 4 cuando dibujábamos árboles colocándolos con el ratón, así, se incialmente se comprueba que se hay pinchado con el botón izquierdo y no esté mostrada la ayuda.
En tal caso, se obtiene el viewport y se crea el buffer de selección, se indica a OpenGL que el modo de trabajo será selección, se inicializan los nombres, se fija la transformación de proyección y se establece la ventana de selección te 1x1 pixels. Tras esto se vuelve a modo de proyección y al modo GL_RENDER, obteniendo así el numero de objetos seleccionados.

Si no hay ninguno (hits==0) no se hará nada. En caso contrario se obtienen los dos identificadores necesarios (identificadores de la fila y la columna en la que está la bola pinchada). Antes de realizar ninguna otra operación se copia la matriz "fichas" en la matriz "deshacer" y se asigna el valor de "bolas2" a "bolas", con lo que tenemos el estado actual guardado para poder restaurarlo mediante la opción de deshacer.
Se llama a la función "jugada" pasándole como argumento la fila y columna de que acabamos de obtener como identificadores que realizará precisamente eso, la jugada, eliminando en su caso las bolas correspondientes.
Si dicha función devuelve 0 es porque no se ha modificado el tablero (no habia jugada posible) y por tanto no se hace nada más. En caso contrario se llama a "reajustaTablero" que elimina las bolas quitadas y desplaza el resto. Tras esto se llama a "masMovimientos" que comprueba si puede continuar el juego o no. Si no hay mas movimientos se mostrará el mensaje oportuno (para ello se iguala ayuda a 2) y si "bolas" es 0, no quedan bolas y se muestra el mensaje de enhorabuena (igualando ayuda a 3).
Finalmente se redibuja, con lo que se mostrará la el tablero tras la jugada realizada si quedan bolas y quedan movimientos o el mensaje oportuno en caso contrario.

int jugada(GLuint ii, GLuint jj)
****

A partir de la bola seleccionada se obtienen en "vector" la secuencia de coordenadas correspondientes a las bolas que deberán ser eliminadas por ser del mismo color a la indicada por "ii" y "jj".
Esta operación se lleva a cabo realizando lo que podría llamarse un "recorrido en anchura", es decir, dada la bola pulsada (primera que se introduce en "vector"), se comprueba la bola que hay encima, a la derecha, abajo, y a la izquierda (en ese orden y siempre que no nos salgamos de los límites). Para cada una de las cuatro direcciones si la bola consultada es del mismo color a la inicial, sus coordenadas se menten en la siguiente posicion del vector. Tras comprobar las cuatro posiciones anteriormente comentadas para una bola, se avanza a la siguiente posición del vector y se repite el proceso mientras queden bolas en dicho vector que comprobar.
En cada caso antes de introducir una "bola" en el vector de bolas a eliminar se comprueba si no está ya, ya que si por ejempo meto una bola y luego la que está encima, cuando esté comprobando la segunda y compruebe la que tiene debajo podría meterla por ser del mismo color pero ésta es la bola "padre" de la actual y no debe volver a introducirse, esto se comprueba llamando a la función llamada "repetido" que comprueba precisamente eso.
Así tras terminar el proceso en vector se tiene el conjunto de coordenadas de bolas a eliminar, así que por último se igualan a 9 (9 indica que una bola ha sido eliminada en una jugada, esto se hace para que cuando se llame a "reajustaTablero" solo se pongan de nuevo a 0 las ultimas eliminadas y para marcar los límites de las bolas que deben desplazarse. Dentro de la función "reajustaTablero" las posiciones con valor 9 se igualan definitivamente a 0 indicando posición libre) todas las posiciones de "fichas" cuyas coordenadas estén almacenadas en "vector"; y a "bolas" se le resta el índice de "vector+1" (número de bolas eliminadas.

int repetido(int actual, int dir)
****

Función que es utilizada por la anterior y que comprueba si las cooredanas almacenadas arriba, a izquierda, abajo o derecha, según "dir", de la posición "actual" están o no en el vector, es decir, comprueba si dada una bola y una dirección, la bola en dicha dirección ya está en el vector de bolas eliminadas para no volver a incluirla.
El proceso es sencillo y consiste en una secuencia de if-else anidados devolviendo 0 o 1 según el caso.