Appendix E

Graphics: Mouse control

E.1 Overview

This appendix explains how to detect events (e.g. display and mouse operations) and use a number of functions that relate to using the mouse.

The appendix is only relevant if you are using the graphics library `graphics_lib.h`.

E.2 Event detection

We will discuss in detail a small program that allows the user to draw on the display using either a filled or unfilled circle. It will detect where the cursor cordinates are and whether the left and right mouse buttons have been pressed. It also will count the number of times the right mouse button is pressed down and allow the selection of various actions depending on that. It will close the graphics window down if the left mouse button is depressed. It employs a while loop to allow the user to draw using circles unless either the left mouse button is depressed or the graphics window is closed.

E.3 Examining the mouse program "mouse.c"

Assuming you have the "clab" project open in Code:: Blocks change the target to "mouse" and open the source code for the mouse program "mouse.c". We will examine this program now.

The program "mouse.c" begins with some include statements which allow the use of the graphics wrapper functions defined in `graphics_lib.h` and a few other libraries in the standard C library. Then inside the `main` function we define some local variables that are used by the program. To use the mouse we have to initialize it with the function `initmouse`.

```
#include <graphics_lib.h>
#include <stdio.h>
#include <conio.h>

int main(void)
{

    int done = 0, overpaint = 0, outlined = 0;
    int hide_cursor;
    int x_pos, y_pos;
    int num_clicks = 0;

    /* initialize the graphics window */
  initwindow(640, 480);
```

```
  setbkcolor(BLACK);
  setcolor(BLUE);

  /* initialize the mouse */
initmouse();

  /* create an event queue */
  create_event_queue();

  /* register display and mouse as event sources */
  reg_display_events();
  reg_mouse_events();

  printf("\nDo you want to hide the mouse cursor? (0 = NO, 1 = YES) ");
  scanf("%d", &hide_cursor);

  if (hide_cursor)
      hide_mouse_cursor();

  initfont();

  x_pos = 320;
  y_pos = 240;

  outtextxy(5,5,"Press left mouse button to quit");

while (!done)
{
   /* wait for event  */
   wait_for_event();

     /* if graphics windows closed then exit loop */
   if (event_close_display())
      done = 1;
   else if (event_mouse_position_changed()) /* mouse moved */
   {
      get_mouse_coordinates();
      x_pos = XMOUSE;
      y_pos = YMOUSE;
   }
   else if (event_mouse_button_down())
   {
      if (event_mouse_left_button_down()) /* if left mouse button down */
          done = 1;

        if (event_mouse_right_button_down()) /* if right mouse button down */
            num_clicks++;
   }
   /* count clicks from 0 to 3 */
   num_clicks = num_clicks % 4;

   if (num_clicks == 0)
   {
      overpaint = 0;
      outlined = 0;
   }
   else if (num_clicks == 1)
   {
```

```
        overpaint = 0;
        outlined = 1;
     }
     else if (num_clicks == 2)
     {
        overpaint = 1;
        outlined = 0;
     }
     else if (num_clicks == 3)
     {
        overpaint = 1;
        outlined = 1;
     }

       if (outlined)
       {
          setcolor(BLUE);
          circle(x_pos, y_pos, 10, 2); /* outlined circle */
       }
       else
          filled_circle(x_pos, y_pos, 10 ,BLUE); /* filled circle */

      update_display();

      if (overpaint)
      {
         filled_circle(x_pos, y_pos, 10 ,BLACK);
      }
 }

 /* Wait for a key press */
 printf("\nPress a key to quit");
   getch();

   /* close the mouse */
   closemouse();

   /* remove the display */
   closegraph();

   return 0;
}
```

To detect events like the mouse cursor position being changed, or whether a mouse button is being pushed down we need to create an event queue. This is done by calling the function `create_event_queue`. Next we register both the display and the mouse state as sources of events, so that changes in them will be detected as events. This is accomplished with the calls to the functions `reg_display_events` and `reg_mouse_events`.

The program allows the user to decide whether they would like to show or hide the mouse cursor while it is *inside the graphics window*. Often graphics application programs have their own cursor or do not show the default cursor. This is accompliashed in the program by calling the function `hide_mouse_cursor`.

After outputting a message to the graphics window the program enters the "game loop" which is the code in the scope of `while` statement. The first statement inside the loop is `wait_for_event`. This waits for the user to do something. In this case, doing something, means either closing the display, moving the mouse, or pressing a mouse button. These events are detected through a number of functions which all begin with "event". These are respectively:

```
event_close_display
event_mouse_position_changed
event_mouse_button_down
event_mouse_left_button_down
event_mouse_right_button_down
```

If the mouse position has changed the program needs to know where to draw the new circle (i.e. it needs the current position of the cursor). This is done by first calling `get_mouse_coordinates`. This function changes two integer global variables, `XMOUSE` and `YMOUSE` which hold these coordinates. The program then assigns these to some local variables, `x_pos` and `y_pos`

The program counts the number of right mouse button "clicks" and according to the remainder on division of this number by four, sets a number of circle drawing actions. The "remainder on division by" operation is given by the % symbol. This is a very useful command. Its role here is to make sure that `num_clicks` has only one of four values: 0, 1, 2, 3. These values are used as "options" which change the nature of the circle drawing operations through the local variables, `outlined` and `overpaint`. It is important to realise that the call to `update_display` copies the screen buffer to the active display, which means that it will draw blue circles at the current values of `x_pos`, `y_pos` but overwrite in black (if `overpaint` is 1) the circle drawn in the *previous* position.

The `main` function ends with some close-down operations.

We give the reference to the new graphics functions that have been used in "mouse.c" below.

Function Reference: `initmouse` — Sets up the mouse for use

```
initmouse();
```

The `initmouse` function initializes the mouse so that mouse actions can be detected and acted on.

`initmouse` is defined in `graphics_lib.h`.

Function Reference: `create_event_queue` — Create an event queue

```
create_event_queue();
```

The void `create_event_queue` function initializes an event queue so that types of events can be detected.

`create_event_queue` is defined in `graphics_lib.h`.

Function Reference: `reg_display_events` — Allow display events to be detected

```
reg_display_events();
```

The void `reg_display_events` function allows display events to be detected.

`reg_display_events` is defined in `graphics_lib.h`.

Function Reference: `reg_mouse_events` — Allow mouse events to be detected

```
reg_mouse_events();
```

The void `reg_mouse_events` function allows mouse events to be detected.

`reg_mouse_events` is defined in `graphics_lib.h`.

Function Reference: `hide_mouse_cursor` — Hide the mouse cursor in the display window

```
hide_mouse_cursor();
```

The void `hide_mouse_cursor` function prevents the mouse cursor from being visible in the display window.

`hide_mouse_cursor` is defined in `graphics_lib.h`.

Function Reference: `event_close_display` — Detect whether the display window has been closed

```
event_close_display();
```

The int `event_close_display` returns 1 if the display window has closed otherwise it returns 0.

`event_close_display` is defined in `graphics_lib.h`.

Function Reference: `event_mouse_position_changed` — Detect whether the mouse position has changed

```
event_mouse_position_changed();
```

The int `event_mouse_position_changed` returns 1 if the mouse has been moved, 0 otherwise.

`event_mouse_position_changed` is defined in `graphics_lib.h`.

Function Reference: `get_mouse_coordinates` — Get the integer coordinates of the mouse cursor

```
get_mouse_coordinates();
```

The void `get_mouse_coordinates` writes the integer coordinates of the mouse cursor to global variables XMOUSE and YMOUSE.

`get_mouse_coordinates` is defined in `graphics_lib.h`.

Function Reference: `event_mouse_button_down` — Detect whether a button on the mouse is pressed

```
event_mouse_button_down();
```

The int `event_mouse_button_down` returns 1 if any mouse button is down, otherwise it returns 0.

`event_mouse_button_down` is defined in `graphics_lib.h`.

Function Reference: `event_mouse_left_button_down` — Detect whether the left button on the mouse is down

```
event_mouse_left_button_down();
```

The int `event_mouse_left_button_down` returns 1 if the left mouse button is down, otherwise it returns 0.

`event_mouse_left_button_down` is defined in `graphics_lib.h`.

Function Reference: `event_mouse_right_button_down` — Detect whether the right button on the mouse is down

```
event_mouse_right_button_down();
```

The int `event_mouse_right_button_down` returns 1 if the right mouse button is down, otherwise it returns 0.

`event_mouse_right_button_down` is defined in `graphics_lib.h`.

Function Reference: `closemouse` — Shut down the mouse reading functions

```
closemouse();
```

The void `closemouse` closes down the mouse as an event device.

`closemouse` is defined in `graphics_lib.h`.

E.4 Running and altering the mouse program

Build and execute the program "mouse.c". Do this from the "command prompt" window. Choose to hide the mouse cursor and try moving the mouse and depressing the right mouse button. It is easy to get interesting looking pictures! Left click the mouse and you will find the graphics window closes and returns the program to the "command prompt". Run the program again and after you have finished drawing close down the graphics window (by clicking the X at the top right corner).

E.5 Summary

After reading through this appendix and running the mouse program, you should have a good idea how to use the mouse in your programs. If you are doing the music option and want to use the mouse in your program you will have to include the graphics library `graphics_lib.h` in your code.