CSE 142
Computer Programming I

Incredibly Useful Stuff

Overview

HW4: The "event loop"

- event = GP142_await_event(&mouse_x, &mouse_y, &key_pressed);
  Huh? (*Think scanf()*)
- Symbolic constants and .h files
- The switch statement
  Textbook sec. 4.8
- Screen coordinates
- Drawing on the screen
  It's NOT printf()
- Animation
  The GP142_PERIODIC event

The HW Event Loop

```
quit = FALSE;
do {
   /* grab an event and deal with it*/
   event = GP142_await_event(&mouse_x, &mouse_y,
                                        &key_pressed);

   •••

}  while (!quit); /* end event (do) loop */
```


GP142_await_event()

- It's like scanf(), in the sense that program execution waits inside this function until the user does something
- It finishes waiting when the user does anything:
  - Clicks with the mouse
  - Types a key
  - Closes the window (to end the program)
  - One other thing we'll get to in a minute
- It returns two things
  - What kind of event happened
  - Particulars about that event

Note: These two are a little imprecise. Full details in a minute.

GP142_await_event()

event = GP142_await_event(&mouse_x, &mouse_y, &key_pressed);

Variable event (an int) gets a value indicating the kind of event that occurred causing the function to return at all:
- Mouse click, or key pressed, or window closed, or that other thing

If it was a mouse click:
- mouse_x and mouse_y receive the x and y screen position of the click (key_pressed is nonsense)

If it was a key press:
- key_pressed (a char) gets the character of that key

If the window was closed:
- None of the output parameters has a useful value

GP142_await_event()

event = GP142_await_event(&mouse_x, &mouse_y, &key_pressed);

Variable event (an int) gets a value indicating the kind of event that occurred causing the function to return at all

What values, exactly?

- Well, they're integers
- But, we don't care which integers, because they're symbolic constants, so we just use their names
  - GP142_QUIT ⇒ window closed, time to terminate
  - GP142_KBD ⇒key pressed
  - GP142_MOUSE ⇒mouse click
  - GP142_PERIODIC ⇒the mystery event…

Where are the symbolic constants defined?

```
event = GP142_await_event(&mouse_x, &mouse_y, &key_pressed);
if (event == GP142_QUIT) {
   …
} else if (event == GP142_KBD) {
   …
```

- They're defined in file gp142.h
- The line "#include <gp142.h>" at the top of your code means:
  - Find file "gp142.h"
  - Before you start compiling, take all of its lines and insert them right here in this file
  - Voila
- YOU DO NOT NEED TO READ gp142.h!!!!!!!!

J-7

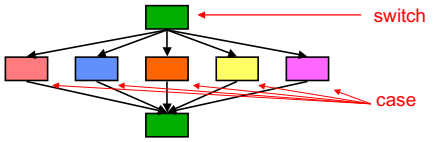
The switch Statement

```
event = GP142_await_event(&mouse_x, &mouse_y, &key_pressed);
if (event == GP142_QUIT) {
   …
} else if (event == GP142_KBD) {
   …
} else if (event == GP142_MOUSE) {
   …
} else if (event == GP142_PERIODIC) {
   …
}
```

Variable event is not modified by any of the "then clauses" ⇒There's a simpler / clearer way to write this

J-8

The switch Statement



J-9

switch Statement

```
switch (int expression) {
   case value1:   stmt1;
                  stmt2;
                  …
                  break;
   case value2:   …
                  break;
   default:       …
                  break;
}
```

switch means "pick one of the following choices"

case value: means "here are the statements if the expression equaled value"

break means "end of statements for this value, skip to end"

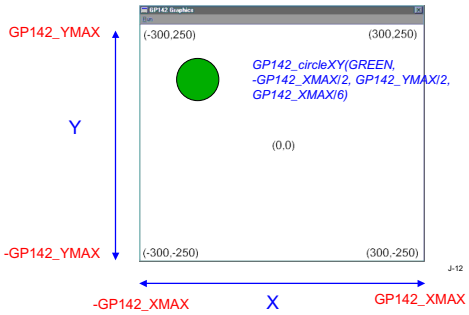
default means "if the expression doesn't equal any of the values in the case's, come here"

```
event = GP142_await_event(&mouse_x, &mouse_y, &key_pressed);
switch (event) {
      case GP142_QUIT:      stmt1;
                            stmt2;
                            …
                            break;
      case GP142_KBD:       stmt;
                            …
                            break;
      case GP142_MOUSE:     stmt;
                            …
                            break;
      case GP142_PERIODIC:  stmt;
                            …
                            break;
      default:              stmt;
                            …
                            break;
} /* end switch */
```

Don't forget the break's!!!

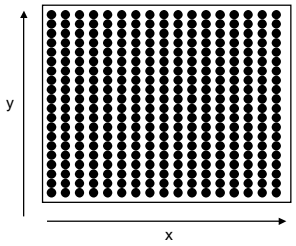
J-11

Screen Coordinates



J-12

Screen Coordinates



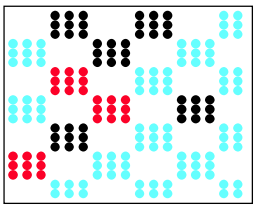
The screen is composed of *pixels* arranged in a grid.

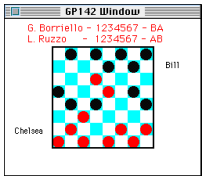
A *screen coordinate* is an (x,y) position naming a pixel.

(*Screen resolution* is the number of pixels your monitor can display. E.g., "800 x 600" or "1280 x 1024")

J-13

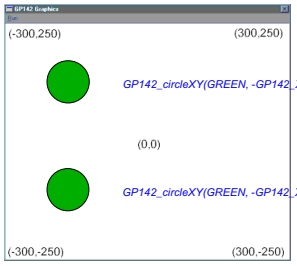
Pixels ⇒Images





J-14

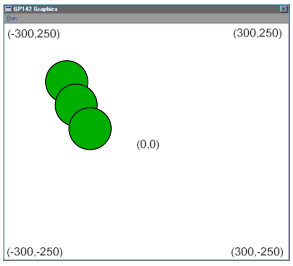
Drawing on the Screen Is Cumulative



J-15

GP142_clear();

So How Do Things Move



J-16

Animation

- To make it appear than an object is moving:
  - Draw it in position (x,y)
  - Wait a little while
  - Clear the screen and draw it at position (x+d, y+e)
  - Repeat
- How do you "wait a little while?"
  - The gp142_periodic event

J-17

gp142_periodic Events

- It's a timer
- Goes off about 10-20 times/second
- The timer can be *on* or *off*
  - GP142_animate(ANI_HALT);
    turns the timer off ⇒no gp142_periodic events occur
  - GP142_animate(ANI_RUN);
    turns the timer on ⇒gp142_periodic events occur 10-20 times / second
  - Bingo, you're quakin'

J-18

Incredibly Important Stuff Wrap-Up

1. The event loop and what it means
2. Using symbolic constants defined by whoever wrote the gp142.c code
3. What "#include <gp142.h>" means
4. The switch statement
5. Screen coordinates
6. Why you have to do an "erase" when writing to the screen, when you didn't when printf'ing
7. Animation: what and how

J-19