CS 349 - User Interfaces **Winter 2018**

Lecture 5: January 12, 2018

Lecturer: Keiko Katsuragawa *Notes By: Harsh Mistry*

5.1 Events

5.1.1 Event Driven Programming

In event driven programming nothing happens unless some other event occurs first. i.e

- User presses a key,
- User moves the mouse
- Window is resized/closed/covered
- Timer expires

5.1.1.1 Events Defined

- English : An observable occurrence, often extraordinary occurrence
- User Interface Architecture : A message to notify an application that something happened
- Examples
  - Keyboard
  - Pointer Events
  - Window crossing
  - Input focus
  - Window events
  - Timer

5.1.1.2 Role of the Base Window System

- Collect event information
- Put relevant information in a known structure
- Order teh events by time
- Decide which application/window should get event
- Deliver the event
- Some events come from teh user via the underlying hardware; some from the window manager

5.1.1.3 Receiving events

- In X windows, applications get the next event using : `XNextEvent(Display* display, XEvent* evt)`
  - Gets and removes teh next event in the queue
  - **If empty, it blocks until another event arrives**
- To avoid blocking you can use `XPending(Display* display)`

5.1.1.4 Selecting input events to listen to

Selecting Input Events to "listen to"

```
// Tell the window manager what input events you want.
XSelectInput( display, window,
              ButtonPressMask | KeyPressMask |
              ButtonMotionMask );
```

- Defined masks:

  ```
  NoEventMask, KeyPressMask, KeyReleaseMask,
  ButtonPressMask, ButtonReleaseMask, EnterWindowMask,
  LeaveWindowMask, PointerMotionMask,
  PointerMotionHintMask, Button1MotionMask,
  Button2MotionMask, ..., ButtonMotionMask,
  KeymapStateMask, ExposureMask, VisibilityChangeMask,
  ...
  ```

- See
  - http://www.tronche.com/gui/x/xlib/events/types.html
  - http://www.tronche.com/gui/x/xlib/events/mask.html

Slide Taken From Lecture

5.1.1.5 Event Structure

- X uses a union

```
typedef union {
   int type;
   XKeyEvent xkey;
   XButtonEvent xbutton;
   XMotionEvent xmotion;
   // etc. ...
} XEvent;
```

- Each structure contains at least the following

```
typedef struct {
   int type;
   unsigned long serial;  // sequential #
   Bool send_end;   // from SendEvent request?
   Display* display; // display event was read from
   Window window; // window which event is relative to
} X___Event;
```

Slide Taken From Lecture

5.1.2 Animation

Animation is the simulation of movement created by displaying a series of pictures, or frames

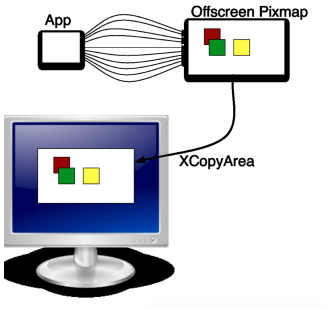
Animation Timing and Responding to Events (non-blocking)

```
while( true ) {

   if (XPending(display) > 0) { // any events pending?
      XNextEvent(display, &event ); // yes, process them
      switch( event.type ) {
        // handle event cases here ...
      }
   }

   // now() is a helper function I made
   unsigned long end = now(); // time in microseconds

   if (end - lastRepaint > 1000000/FPS) { // repaint at FPS
      handleAnimation(xinfo); // update animation objects
      repaint(xinfo); // my repaint
      lastRepaint = now(); // remember when the paint happened
   }

   // IMPORTANT: sleep for a bit to let other processes work
   if (XPending(xinfo.display) == 0) {
      usleep(1000000 / FPS - (end - lastRepaint));
   }
}
```

Slide Taken From Lecture

5.1.3 Double Buffering

Flickering can occur when an intermediate image is on the display. To resolve this we can use double buffering, which involves rendering images off screen to a buffer and then fast copying the buffer to the screen



Graphic Taken From Lecture

5.1.4 Painting Advice

- Keep it simple
  - Clear everything and redraw each frame
  - Use advanced methods, only if you really need them for performance
- Don't repaint too often. Consider adding a "someChanged" bool flag
- Don't flush too often