Today’s class

- Input and interaction
Interactive input devices

- **Locator**
  - specifies coordinate positions
  - examples: thumbwheels, tablet, mouse, trackball, joystick, touch panel

- **Stroke**
  - specifies a sequence of coordinate positions
  - example: tablets
Interactive input devices (cont.)

- **Pick**
  - selects picture components
  - example: light pen

- **Valuator**
  - specifies scalar values
  - example: dial
Interactive input devices (cont.)

- **String**
  - specifies text input
  - example: keyboard

- **Choice**
  - selects menu options
  - examples: keyboard, mouse buttons
Event driven devices

- CPU must wait on the device before it can do anything
- Examples:
  - keyboard
  - mouse
Example: rubberbanding

- Rubberbanding is the process of having a visual indicator move around the graphics window as the mouse moves.
- Events that need to be recognized are mouse movement and mouse button clicks (to indicate starting and stopping of rubberbanding).
Overlay planes

- While rubberbanding the actual image must be preserved.
- This is best accomplished by using an overlay plane, such that the actual image is drawn in the normal draw plane and the rubberbanding shape is drawn in the overlay plane.
- The image seen on the screen is made up of the layering of these two planes:
  - if a pixel in the overlay plane is transparent, the pixel in the normal draw plane is seen.
  - if a pixel in the overlay plane is a normal color, it is seen and the image’s pixel is hidden.
Overlay planes in OpenGL and GLUT

- `glutEstablishOverlay();` is used to establish an overlay plane
- `glutUseLayer(GLUT_OVERLAY);` is used to draw in the overlay plane
- `glutUseLayer(GLUT_NORMAL);` is used to return drawing to the normal draw plane
- OpenGL keeps separate projection matrices for overlay planes
Example program

- Program `rubberband.c` illustrates mouse events and the overlay plane.
- It runs on the SGI and Sun workstations, but not the laptops as Windows does not support overlay planes.
Overview of picking

- First draw scene into frame buffer as usual
- Now change to selection mode and redraw the scene (contents of frame buffer will not change until selection mode is exited)
- Upon exit of selection mode, OpenGL returns a list of primitives that would have intersected the viewing volume
Overview of picking (cont.)

- Each of these primitives causes a selection hit.
- The list of primitives returned as an array of integer-valued names and related data (hit records) that correspond to the current contents of the name stack.
- Name stack is constructed by loading names onto it as you issue primitive drawing commands while in selection mode.
Basic steps

- Specify array to be used for the returned hit records with `glSelectBuffer()`, which is passed:
  - maximum number of values that can be stored in the array
  - an array capable of holding unsigned integers
- Enter selection mode with `glRenderMode (GL_SELECT);`
Basic steps (cont.)

- Initialise the name stack with `glInitNames();` and `glPushName(-1);`
- Define the viewing volume you want for selection with the help of `gluPickMatrix()` (usually different from rendering, so you will want to use `glPushMatrix();` and `glPopMatrix();` to save and restore rendering transformation)
Basic steps (cont.)

- Alternately issue commands to alter the name stack and primitive drawing commands so that each primitive of interest has an appropriate name assigned.
- Exit selection mode and process the returned selection data (the hit records).
Hit records

Contents, in order, of a hit record:

- Number of names on the name stack when the hit occurred
- Minimum and maximum window-coordinate z values of all vertices of the primitives that intersected the viewing volume since the last recorded hit (they lie in the range 0-1 but are multiplied by $2^{32}-1$ and rounded to the nearest unsigned integer)
Hit records (cont.)

- Contents, in order, of a hit record (cont.):
  - Contents of the name stack at the time of the hit, with the bottommost element first
Example program

- Program `picksquare.cpp` demonstrates the complete technique for picking