Review of Thursday

- **Scene graphs**
  - Hierarchical scene representation
  - Logical or spatial hierarchy
  - Object-oriented scene model (instancing)

- **Spatial partitioning**
  - Scene pre-processing (of static scenes)
  - Include visibility information for early selection/rejection
  - Faster visibility or collision detections due to hierarchy
  - Octrees, BSP trees, …
What is ‘Interaction’?

- Complex scenes (or complex data in general) cannot be **explored** in their entirety from one static view.

- **Interaction** provides a way to “tell the computer” what you want to see.

- Interaction (inter-action) is a two-way process. The display reacts to your input and you react to what the display shows.
What we will learn about ‘Interaction’

• We have discussed a selection of input devices in Chapter 1 (including devices with 6 DOF)

• Taking a more abstract view:
  • What do we want to achieve through interaction?
  • How do we approach interactivity from a software point-of-view?
  • What are human-computer-interaction techniques used in graphics?
The Purpose of Interaction

- The image generation process (rendering) is governed by a number of parameters
  - Object positions
  - Optical properties (color, etc.)
  - Algorithmic parameters
  - Camera properties
  - ...

- One fixed set of parameter values rarely fits our needs

- Interaction allows us to explore this parameter space
  - Example: Changing camera position by “flying” through the scene
Interaction

- **User interfaces (UI)** represent a system that allows interaction to take place (input and output)

See also: **Graphical User Interface (GUI)**
User Interface APIs

- OpenGL does not provide an interface for user interaction

- Other libraries and APIs are available and can be used to provide a context for OpenGL rendering
  - GLUT (keys, mouse, etc.)
  - Qt (keys, mouse, GUI elements, etc.)
  - Simple DirectMedia Layer (SDL, mouse, keys, audio, etc.)
  - …
Interaction

• APIs abstract from actual hardware by using **logical devices** (hardware can be replaced)

• When triggered, logical devices provide different input or information to the system
  • Numbers (ASCII, letters, etc. – for example: Keyboard)
  • Positions (absolute, relative – for example: Touchpad or Mouse)
Input Modes

• Input provided by logical devices may be handled in different ways

• **Request Mode:**
  • Input is stored in buffer, until device is triggered.
  • Example: Keyboard input on terminal. Program waits until input triggered by return key.

• **Sample Mode:**
  • Program reads contents (sample) of input from buffer. User must have ‘filled’ the buffer with input at this time

• **Event Mode:**
  • Input is stored as event in a queue. Input is independent from what the rest of program is doing.
Event Mode

- (Multiple) logical devices can send events to the UI system
- Data of the event is stored in an event queue for further processing
- UI programs make use of an event loop that handles and dispatches events from logical devices

Diagram:

- Trigger
- Measure
- Event Queue
- Program

Physical input

Await

Event
Events in GLUT

- **Example: GLUT**
  - Main loop is an infinite event loop
  - Every pass looks at events in the queue
  - Calls appropriate callback function (for example: `glutKeyboardFunc()`)

- Different events are recognized by GLUT
  - Display events
  - Mouse events
  - Keyboard events
  - Idle ‘events’, …
GUI Interaction

- **GUI APIs** use events to model graphical input devices
- The use of such **widgets** provides an additional layer of abstraction

- Typical widgets include:
  - Buttons, Radio Buttons, Check Boxes
  - Slidebars, Scrollbars
  - Drop-Down Menus, Menus

- The API provides abstract input about state changes of these widgets to the programmer (“Button was clicked”)
Interaction Techniques

• How can we map input/events to actions?

• Typical actions in computer-graphics:
  • Changing a (scalar) parameter
  • Entering values/text
  • Navigation ("walking, looking, flying")
  • Picking/Selection
  • Drawing/Dragging
GUI Interaction

- Designing a **graphical user interface** allows us to present the user with graphical representations of the possible actions/interaction options.

- Parameter settings are conveyed not only through changes in the scene, but also through the user interface (Checkbox is marked, etc.)
GUI Interaction

• A variety of standard GUI elements/widgets exist
  • Manual implementation is not trivial (mouse-in-element test, rendering, etc.)

  ![Example GUI Interface]

  • Overall design/usability can benefit from custom implementation
  • GUIs are designed to make a certain task as easy as possible, while providing enough flexibility (iPhone example)
Interaction Techniques

• Entering text, changing scalar/vector properties
  • Non-graphical UIs have limited use if not coupled to direct feedback
  • Pre-defined GUI widgets often include preview mode
  • Input characters encoded as ASCII (example: ‘T’ = 84, ‘w’ = 119)

![Color dialog box, Qt](image-url)
Interaction Techniques

- **Text in OpenGL**
  - OpenGL has no built-in text rendering functionality
  - Basic text rendering can be achieved through **bitmap fonts**

- Third-party software provides established font rendering or text processing techniques (GLUT, Qt, …)

Input ASCII code is mapped to texture coordinates in bitmap

<table>
<thead>
<tr>
<th>ASCII Code</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>67</td>
<td>`</td>
</tr>
<tr>
<td>74</td>
<td></td>
</tr>
</tbody>
</table>
Navigation in a Scene

• Navigating through the data/scene is a key action in computer-graphics.

• Most input devices operate in 2D space (mouse, touchpad,…). This requires to perform a mapping from 2D to 3D interaction.

• 3D interaction is significantly more complex due to high DOF of the camera.
Navigation in a Scene

- In 3D rigid bodies have six **degrees of freedom (DOF)**
Navigation in a Scene

- Relative motion in 2D (for example: mouse) can be mapped to six DOF in 3D

- A large number of virtual models can be implemented to define this mapping
  - Flying (including ‘physical’ models)
  - Walking (first-person exploration)
  - Virtual sphere
Navigation in a Scene

• The virtual sphere/trackball/arcball
  • 2D interaction mapped to translation along the surface of a sphere
  • Creates rotation around origin
  • Additional modifiers allow translation of sphere or change of radius
Navigation in a Scene

• The virtual sphere/trackball/arcball
  • Simple implementation:
    • Express camera position in spherical coordinates

\[
\begin{pmatrix}
  x \\
  y \\
  z
\end{pmatrix} = \begin{pmatrix}
  r \sin(\alpha) \cos(\beta) \\
  r \sin(\alpha) \sin(\beta) \\
  r \cos(\alpha)
\end{pmatrix}
\]

• Motion in x,y (mouse) is mapped directly to changes in the two angles (inclination and azimuth)
• Problem: Degeneracies around poles of the sphere
Navigation in a Scene

- The virtual sphere/trackball/arcball
  - Implementation 2:
    - When dragging from \((x_1, y_1)\) to \((x_2, y_2)\), map 2D positions to two positions on the sphere surface \(p_1, p_2\)
    - Compute angle of rotation from
      \[
      \cos(\alpha) = \frac{(p_1 - o) \cdot (p_2 - o)}{\| (p_1 - o) \| \cdot \| (p_2 - o) \|}
      \]
    - Compute axis of rotation as
      \[
      v = (p_1 - o) \times (p_2 - o)
      \]
Picking

• (Graphics) Modeling tools use **picking** to let the user identify/select, and modify objects

• Picking requires that we ‘reverse’ the transformation process that objects undergo in the graphics pipeline
Picking

• **Technique 1**
  • When a click event is registered:
  • Render every object in a distinct color (no shading!)
  • Read color from buffer at selection location

```
red = object 1
```
Picking

• Technique 2
  • OpenGL <3.0: **Selection Mode**
  • When a click event is registered
  • Define a part of the scene (around the cursor) as selection region
  • Draw the scene with IDs (*glLoadName()*) assigned to objects
  • Process the IDs that were determined to have been ‘hit’
Picking

- Custom implementations
  - Ray & Bounding-box intersection test
  - Depth-peeling to determine occlusion

- Common problems:
  - How to handle transparency
  - How to handle selection of multiple objects
Drawing and Dragging

• Copy & Paste, Cut & Paste, Move, (Complex) Selection

• Drawing and dragging
  • Interaction started by trigger (mouse-down)
  • Interaction ended by trigger (mouse-up)
  • Main event consists of a sequence of location data
  • Dragging involves redrawing of active object