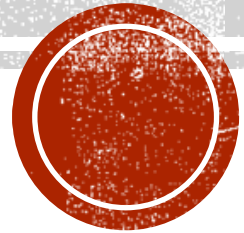
COMPUTER GRAPHICS AND FUNDAMENTALS OF IMAGE PROCESSING



MODULE III

SYLLABUS

Interactive Input Methods and Graphical User Interfaces: Graphical Input Data ,Logical Classification of Input Devices, Input Functions for Graphical Data , Interactive Picture-Construction Techniques, Virtual-Reality Environments, OpenGL Interactive Input-Device Functions, OpenGL Menu Functions , Designing a Graphical User Interface.

Computer Animation :Design of Animation Sequences, Traditional Animation Techniques, General Computer-Animation Functions, Computer-Animation Languages, Character Animation, Periodic Motions, OpenGL Animation Procedures.

WHAT IS INTERACTIVE INPUT METHODS

- We can construct graphics programs and provide input data using the methods and program commands.
- Better than that is useful to be able to specify graphical input interactively.
- For E.g. :

1. We can change the location of an object in a scene by pointing to a screen position.
2. We might want to change animation parameters using menu selections.

GRAPHICAL INPUT DATA

Graphics programs use several kind of input data

For Example :

- Coordinate positions
- Attribute values
- Character-string specifications
- Geometric-transformation values
- Viewing conditions
- Illumination parameter

GRAPHICAL INPUT DATA

- Many graphics packages provide extensive set of input functions for processing such data.
- But input procedures require interaction with display-window managers and specific hardware devices.
- Therefore, some graphics systems, particularly those that provide mainly device-independent functions, often include relatively few interactive procedures for dealing with input data.
- A standard organization for input procedures in a graphics package is to classify the functions according to the type of data that is to be processed by each function
- This scheme allows any physical device, such as a keyboard or a mouse, to input any data class

LOGICAL CLASSIFICATION OF INPUT DEVICES

- According to the type of data the input device provides we can classify input devices.
- The standard logical input-device classifications are:

LOCATOR	A device for specifying one coordinate position.
STROKE	A device for specifying a set of coordinate positions.
STRING	A device for specifying text input.
VALUATOR	A device for specifying a scalar value.
CHOICE	A device for selecting a menu option.
PICK	A device for selecting a component of a picture.

LOGICAL CLASSIFICATION OF INPUT DEVICES

I. Locator Devices

- locator devices allow users to specify a single coordinate position(x,y)

Examples:

1. Mouse Cursor: Users choose a location by clicking a button.
2. Cursor Keys on Keyboard: Users move the cursor using arrow keys (up, down, left, and right).
3. Data Tablets: These devices provide precise input for graphic design and drawing.
4. Touch Panels: Commonly used in touchscreens and mobile devices.
5. 2D Joysticks and Trackballs: These allow users to control both 2D translations and rotations

LOGICAL CLASSIFICATION OF INPUT DEVICES

II. Stroke Devices

- This class of logical devices is used to input a sequence of coordinate positions,
- The physical devices used for generating locator input are also used as stroke devices.
- Continuous movement of a mouse, trackball, joystick, or hand cursor is translated into a series of input coordinate values.
- The graphics tablet is one of the more common stroke devices
- As the cursor is moved across the tablet surface, a stream of coordinate values is generated.

LOGICAL CLASSIFICATION OF INPUT DEVICES

III. String Devices

- The primary physical device used for string input is the keyboard.
- These are the device used for specifying text input.
- Other physical devices can be used for generating character patterns for special applications.
- Individual characters can be sketched on the screen using a stroke or locator-type device.

LOGICAL CLASSIFICATION OF INPUT DEVICES

IV. Valuator

- A device for specifying scalar values.
- Valuator input refers to any kind of input where you set a value that can change, like adjusting a volume knob or changing the brightness on a screen.
- In graphics programs, these values might control things like the size of shapes, camera angles, lighting, or even physical properties like temperature.

Example : Control dial

- Think of these like the volume knob on a stereo. When you turn the dial, it changes a number within a certain range.

Example :Joysticks and Trackballs:

- These devices can be used to adjust values by moving them in different directions. For example, moving a joystick left might decrease a value, while moving it right might increase it.

LOGICAL CLASSIFICATION OF INPUT DEVICES

V. Choice Devices

- These are devices for selecting menu options.
- Commonly used choice devices for selecting a menu option are cursor-positioning devices such as a mouse, trackball, keyboard, touch panel, or button box.
- For screen selection of listed menu options, we use a cursor-positioning device.
- When a screen-cursor position (x, y) is selected, it is compared to the coordinate extents of each listed menu item.
- A menu item with vertical and horizontal boundaries at the coordinate values x_{min}, x_{max}, y_{min}, and y_{max} is selected if the input coordinates satisfy the inequalities

$$x_{min} \leq x \leq x_{max}, \quad y_{min} \leq y \leq y_{max}$$

LOGICAL CLASSIFICATION OF INPUT DEVICES

V. Choice Devices

Imagine you have a vertical list of menu items:

1. File
2. Edit
3. View
4. Help

- Each of these items occupies a rectangular area on the screen. Suppose the "Edit" item has the following boundaries:
- xmin = 10x xmax=100
- ymin=50 ymax=80

LOGICAL CLASSIFICATION OF INPUT DEVICES

V. Choice Devices

- If you position the cursor at coordinates (x, y) = (50, 60) and click, the system will check if these coordinates fall within the boundaries of "Edit":
- $10 \leq 50 \leq 100$ (true)
- $50 \leq 60 \leq 80$ (true)
- Since both conditions are satisfied, "Edit" is selected.

LOGICAL CLASSIFICATION OF INPUT DEVICES

VI. Pick Devices

- A pick device is any input device or method used to select parts of a scene displayed on a screen.

How pick devices works

1. **Cursor Positioning**:

- You use a device like a mouse, joystick, or keyboard to move a cursor on the screen.
- When you click, the program records the cursor's position (x, y coordinates)

2. **Matching Screen Position to Objects**:

- The recorded position is compared to the locations of objects in the scene.
- Each object has boundaries defined by minimum and maximum x and y values (xmin, xmax, ymin, ymax).
- If the cursor's position falls within these boundaries, that object is selected

LOGICAL CLASSIFICATION OF INPUT DEVICES

VI. Pick Devices

Detailed Selection Methods

1)Basic Picking:

- The program checks if the cursor's position (x, y) is within the boundaries of any object.
- If yes, the object is selected for editing or transformation

2)World-Coordinate Mapping:

- The cursor's screen position can be converted to the world coordinates (the actual coordinates used to define the objects in the scene).
- This helps in accurately selecting objects based on their true positions.

LOGICAL CLASSIFICATION OF INPUT DEVICES

VI. Pick Devices

Detailed Selection Methods

3)Handling Overlaps:

- If the cursor's position overlaps multiple objects, additional tests determine which specific object was intended.
- This might involve checking the individual parts of objects, like facets or edges, and comparing distances from the cursor's position

4) Pick Windows:

- A small rectangular area (pick window) is centered around the cursor position.
- Objects intersecting this window are considered for selection.
- For fine selections, the pick window can be made very small to isolate specific parts like line segments.

LOGICAL CLASSIFICATION OF INPUT DEVICES

VI. Pick Devices

Detailed Selection Methods

5) Highlighting:

- Objects overlapping the cursor's position can be highlighted one by one.
- The user can accept or reject each highlighted object until the desired one is selected.

6) **Named Selection**:

- Objects can be selected by typing their names.
- This method is less interactive but can be useful when dealing with complex scenes where objects have descriptive names

INPUT FUNCTIONS FOR GRAPHICS DATA

- Graphics packages that use the logical classification for input devices provide several functions for selecting devices and data classes.
- These functions allow a user to specify the following options:

- The input interaction mode for the graphics program and the input devices. Either the program or the devices can initiate data entry, or both can operate simultaneously.
- Selection of a physical device that is to provide input within a particular logical classification (for example, a tablet used as a stroke device) for a particular set of data values.

INPUT FUNCTIONS FOR GRAPHICS DATA

Input Modes

Some input functions in an interactive graphics system are used to specify how the program and input devices should interact.

There are 3modes.

1. Request Mode
2. Sample Mode
3. Event Mode

INPUT FUNCTIONS FOR GRAPHICS DATA

Input Modes

I. Request Mode

- The application program initiates data entry.
- When input values are requested, processing is suspended until the required values are received.
- Here the program and the input devices operate alternately
- Devices are put into a wait state until an input request is made; then the program waits until the data are delivered

INPUT FUNCTIONS FOR GRAPHICS DATA

Input Modes

I. Sample Mode

- The application program and input devices operate independently.
- Input devices may be operating at the same time that the program is processing other data.
- New values obtained from the input devices replace previously input data values.
- When the program requires new data, it samples the current values that have been stored from the device input.

INPUT FUNCTIONS FOR GRAPHICS DATA

Input Modes

I. Event Mode

- The input devices initiate data input to the application program.
- The program and the input devices again operate concurrently, but now the input devices deliver data to an input queue, also called an event queue
- All input data is saved. When the program requires new data, it goes to the data queue.

Typically, any number of devices can be operating at the same time in sample and event modes. Some can be operating in sample mode, while others are operating in event mode. But only one device at a time can deliver input in request mode.

INPUT FUNCTIONS FOR GRAPHICS DATA

Echo Feed Back

- This is a feature in graphical programs that shows users what they are doing or the settings they are using as they interact with the system. Imagine it as a mirror reflecting back your actions and settings on the screen.

Here are the main things echo feedback might show you:

- **Size of the Pick Window**: If you're clicking or selecting items on the screen, the program might show you the area around your cursor where your click will be recognized.
- **Minimum Pick Distance**: This tells you how close you need to click to an item for the program to consider it a selection. Echo feedback shows this distance to help you understand how precise you need to be.
- **Cursor Type and Size**: The program can change the cursor (like an arrow, crosshair, or hand) depending on what you're doing. Echo feedback shows you the current cursor type and size.
- **Highlighting During Pick Operations**: When you select something, it might change color or get an outline. Echo feedback shows how items will look when selected, so you know they're selected

INPUT FUNCTIONS FOR GRAPHICS DATA

Call Back Functions

- These are special functions in a program that are automatically executed in response to certain events.
- In graphical programs, these events can include things like moving a mouse, clicking a button, or pressing a key on the keyboard.

OPENGL INTERACTIVE INPUT-DEVICE FUNCTIONS

GLUT Mouse Functions

- The glutMouseFunc is a function which specifies a procedure that is to be called when the mouse pointer is in a display window and a mouse button is pressed or released:

```
glutMouseFunc (mouseFcn);

void mouseFcn (GLint button, GLint action, GLint xMouse,
                                           GLint yMouse)
```

- **mouseFcn :** It is a callback function which will handle mouse events.

OPENGL INTERACTIVE INPUT-DEVICE FUNCTIONS

GLUT Mouse Functions

- void mouseFcn (GLint button, GLint action, GLint xMouse, GLint yMouse);

button : Indicates which mouse button was pressed or released.

It can be one of the following GLUT constants.

- ✓ GLUT_LEFT_BUTTON
- ✓GLUT_MIDDLE_BUTTON
- ✓GLUT_MIDDLE_BUTTON

action : Indicates the state of the button. It can be either:

- ✓ GLUT_DOWN (when the button is pressed)
- ✓ GLUT_UP(when the button is released)

OPENGL INTERACTIVE INPUT-DEVICE FUNCTIONS

GLUT Mouse Functions

xMouse,yMouse : These are the coordinates of the mouse cursor relative to the top-left corner of the display window at the time of the event.

OPENGL INTERACTIVE INPUT-DEVICE FUNCTIONS

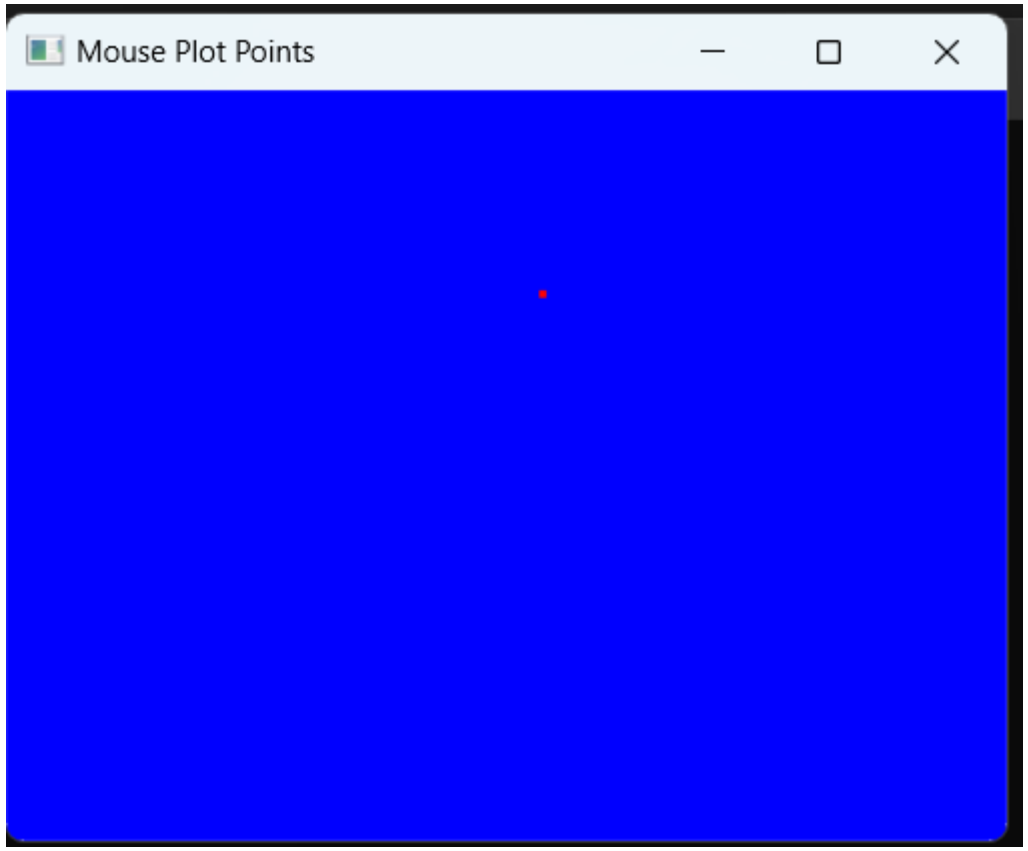
GLUT Mouse Functions

Example : The following program plots a red point, with a point size equal to 3, at the position of the mouse cursor in the display window, each time that we press the left mouse button

```
#include <GL/glut.h>
GLsizei winWidth = 400, winHeight = 300; // Initial display-window size.
void init(void)
{
    glClearColor(0.0, 0.0, 1.0, 1.0); // Set display-window color to blue.
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0, 200.0, 0.0, 150.0);
}
void displayFcn(void)
{
    glClear(GL_COLOR_BUFFER_BIT); // Clear display window.
    glColor3f(1.0, 0.0, 0.0); // Set point color to red.
    glPointSize(3.0); // Set point size to 3.0.
}
```

```
void winReshapeFcn(GLint newWidth, GLint newHeight)
{
    /* Reset viewport and projection parameters */
    glViewport(0, 0, newWidth, newHeight);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, GLdouble(newWidth), 0.0, GLdouble(newHeight));
    /* Reset display-window size parameters. */
    winWidth = newWidth;
    winHeight = newHeight;
}
void plotPoint(GLint x, GLint y)
{
    glBegin(GL_POINTS);
    glVertex2i(x, y);
    glEnd();
}
```

```
void mousePtPlot(GLint button, GLint action, GLint xMouse, GLint yMouse)
{
    if (button == GLUT_LEFT_BUTTON && action == GLUT_DOWN)
        plotPoint(xMouse, winHeight - yMouse);
    glFlush();
}
void main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(winWidth, winHeight);
    glutCreateWindow("Mouse Plot Points");
    init();
    glutDisplayFunc(displayFcn);
    glutReshapeFunc(winReshapeFcn);
    glutMouseFunc(mousePtPlot);
    glutMainLoop();
}
```


Mouse Plot Points

OPENGL INTERACTIVE INPUT-DEVICE FUNCTIONS

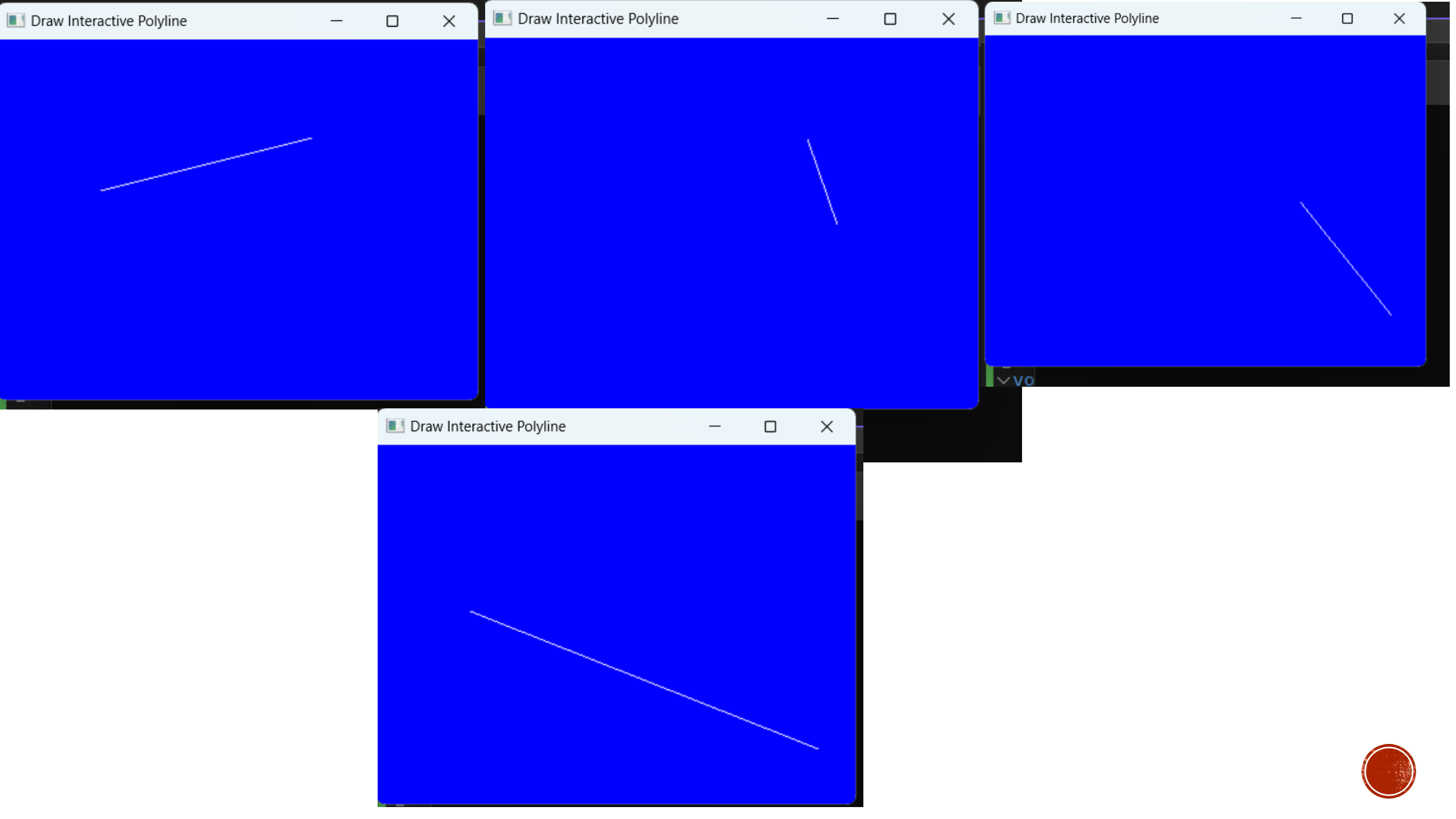
GLUT Mouse Functions

Example :

- The next program example uses mouse input to select an endpoint position for a straight-line segment. Selected line segments are connected to demonstrate interactive construction of a polyline.
- Initially, two display-window locations must be selected with the left mouse button to generate the first line segment. Each subsequent position that we select adds another segment to the polyline.

```
void drawLineSegment(scrPt endPt1, scrPt endPt2)
{
    glBegin(GL_LINES);
    glVertex2i(endPt1.x, endPt1.y);
    glVertex2i(endPt2.x, endPt2.y);
    glEnd();
}
```

```
void polyline(GLint button, GLint action, GLint xMouse, GLint yMouse)
{
    static scrPt endPt1, endPt2;
    if (endPtCtr == 0) {
        if (button == GLUT_LEFT_BUTTON && action == GLUT_DOWN) {
            endPt1.x = xMouse;
            endPt1.y = winHeight - yMouse;
            endPtCtr = 1;
        }
        else
            if (button == GLUT_RIGHT_BUTTON) // Quit the program.
                exit(0);
    }
    else
        if (button == GLUT_LEFT_BUTTON && action == GLUT_DOWN) {
            endPt2.x = xMouse;
            endPt2.y = winHeight - yMouse;
            drawLineSegment(endPt1, endPt2);
            endPt1 = endPt2;
        }
        else
            if (button == GLUT_RIGHT_BUTTON) // Quit the program.
                exit(0);
    glFlush();
}
```


Draw Interactive Polyline
Draw Interactive Polyline
Draw Interactive Polyline
Draw Interactive Polyline

OPENGL INTERACTIVE INPUT-DEVICE FUNCTIONS

GLUT Keyboard Functions

- With keyboard input, we use the following function to specify a procedure that is to be invoked when a key is pressed:

```
glutKeyboardFunc (keyFcn);

void keyFcn (GLubyte key, GLint xMouse, GLint yMouse)
```

- **key :** is assigned a character value or the corresponding ASCII code.
- The display-window mouse location is returned as position **(xMouse, yMouse)** relative to the top-left corner of the display window.
- When a designated key is pressed, we can use the mouse location to initiate some action, independently of whether any mouse buttons are pressed.

OPENGL INTERACTIVE INPUT-DEVICE FUNCTIONS

GLUT Keyboard Functions

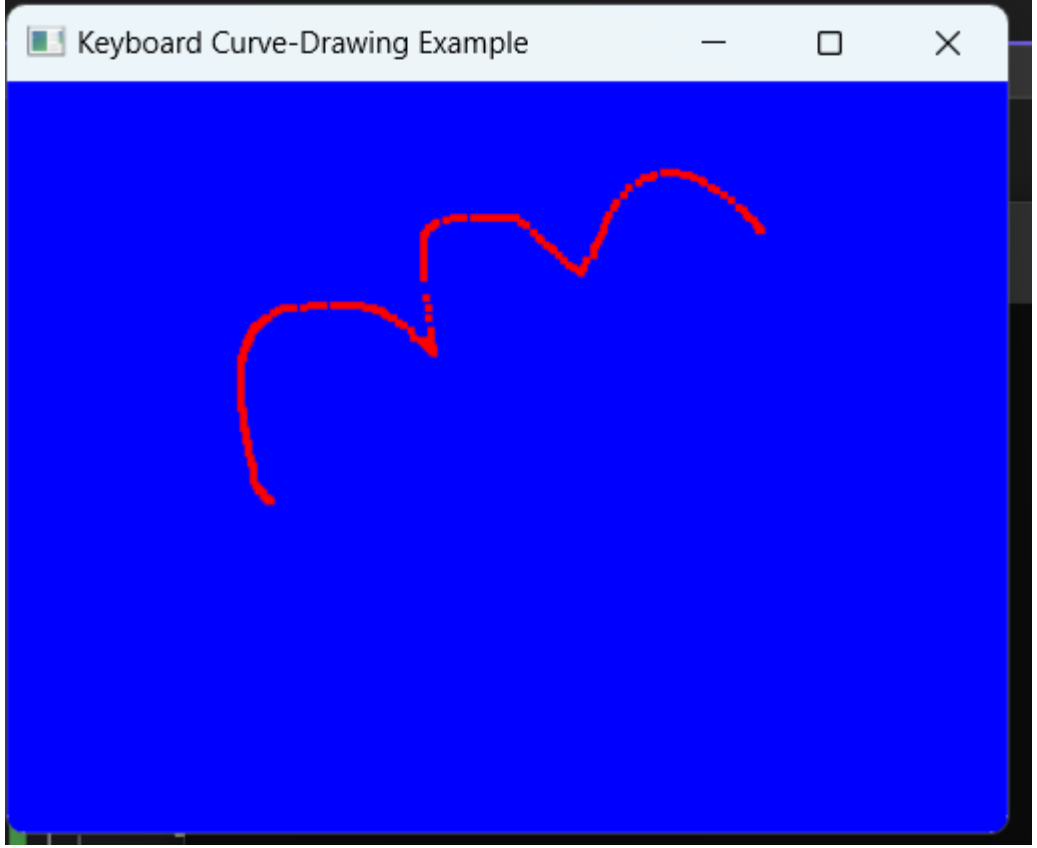
Example : In the following code, we present a simple curve-drawing procedure using keyboard input.

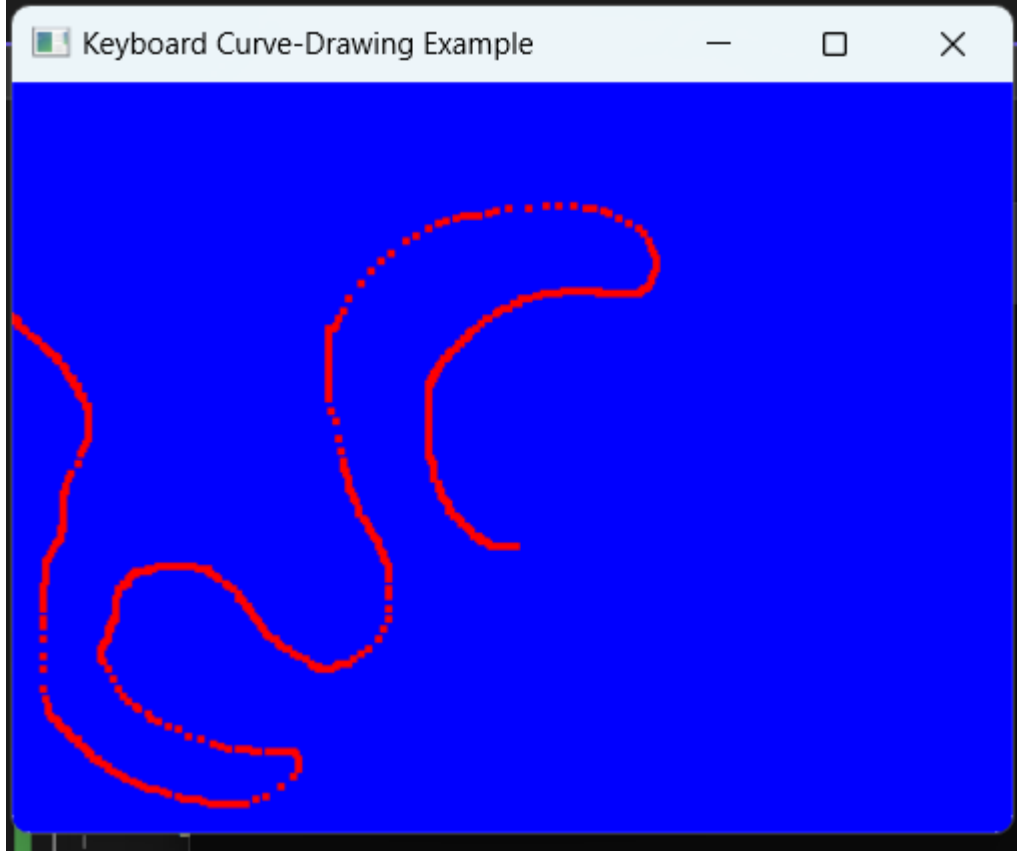
- A freehand curve is generated by moving the mouse within the display window while holding down the "c" key.
- This displays a sequence of red dots at each recorded mouse position.
- By slowly moving the mouse, we can obtain a solid curved line. Mouse buttons have no effect in this example.

```
void plotPoint(GLint x, GLint y)
{
    glBegin(GL_POINTS);
    glVertex2i(x, y);
    glEnd();
}
```

```
void curveDrawing(GLubyte curvePlotKey, GLint xMouse, GLint yMouse)
{
    GLint x = xMouse;
    GLint y = winHeight - yMouse;
    switch (curvePlotKey)
    {
    case 'c':
        plotPoint(x, y);
        break;
    default:
        break;
    }
    glFlush();
}
```

```
void main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(winWidth, winHeight);
    glutCreateWindow("Keyboard Curve-Drawing Example");
    init();
    glutDisplayFunc(displayFcn);
    glutReshapeFunc(winReshapeFcn);
    glutKeyboardFunc(curveDrawing);
    glutMainLoop();
}
```


Keyboard Curve-Drawing Example


Keyboard Curve-Drawing Example

OPENGL INTERACTIVE INPUT-DEVICE FUNCTIONS

GLUT Keyboard Functions

For function keys, arrow keys, and other special-purpose keys, we can use the command

```
glutSpecialFunc (specialKeyFcn);
```

The specified procedure has the same three arguments:

```
void specialKeyFcn (GLint specialKey, GLint xMouse,
                                      GLint yMouse)
```

OPENGL INTERACTIVE INPUT-DEVICE FUNCTIONS

GLUT Keyboard Functions

- Now parameter specialKey is assigned an integer-valued GLUT symbolic constant.
- To select a function key, we use one of the constants GLUT_KEY_F1 through GLUT_KEY_F12.
- For the arrow keys, we use constants such as GLUT_KEY_UP and GLUT_KEY_RIGHT.
- Other keys can be designated using GLUT_KEY_PAGE_DOWN, GLUT_KEY_HOME, and similar constants for the page up, end, and insert keys.
- The backspace, delete, and escape keys can be designated with the glutKeyboardFunc routine using their ASCII codes, which are 8, 127, and 27, respectively.

OPENGL INTERACTIVE INPUT-DEVICE FUNCTIONS

GLUT Tablet Functions

Usually, tablet activation occurs only when the mouse cursor is in the display window. A button event for tablet input is then recorded with

```
glutTabletButtonFunc (tabletFcn);
```

and the arguments for the invoked function are similar to those for a mouse:

```
void tabletFcn (GLint tabletButton, GLint action,
                GLint xTablet, GLint yTablet)
```

OPENGL INTERACTIVE INPUT-DEVICE FUNCTIONS

GLUT Tablet Functions

- **tabletButton :** Usually we will give values like 1,2,3 and so on.
- **action :** It can take values GLUT_UP and GLUT_DOWN
- **xTablet and yTablet :** These are tablet coordinates.

✓We can determine the number of available tablet buttons with the command

glutDeviceGet (GLUT_NUM_TABLET_BUTTONS);

OPENGL INTERACTIVE INPUT-DEVICE FUNCTIONS

GLUT Spaceball Functions

- We use the following function to specify an operation when a spaceball button is activated for a selected display window:

```
glutSpaceballButtonFunc (spaceballFcn);
```

- The callback function has two parameters:

```
void spaceballFcn (GLint spaceballButton, GLint action)
```

- spaceball buttons are identified with the same integer values as a tablet
- action is assigned either the value GLUT_UP or the value GLUT_DOWN.
- We can determine the number of available spaceball buttons with a call to glutDeviceGet using the argument GLUT NUM SPACEBALL BUTTONS.

OPENGL MENU FUNCTIONS

- In addition to the input-device routines, GLUT contains various functions for adding simple pop-up menus to programs.
- With these functions, we can set up and access a variety of menus and associated submenus.
- The GLUT menu commands are placed in procedure main along with the other GLUT functions.

Creating a GLUT Menu

- A pop-up menu is created with the statement

```
glutCreateMenu (menuFcn);
```

- where parameter menuFcn is the name of a procedure that is to be invoked when a menu entry is selected.

OPENGL MENU FUNCTIONS

- This procedure has one argument, which is the integer value corresponding to the position of a selected option.

```
void menuFcn (GLint menuItemNumber);
```

- To specify the options that are to be listed in the menu, we can use the function

```
glutAddMenuEntry (charString, menuItemNumber);
```

charString : specifies text that is to be displayed in the menu

menuItemNumber : gives the location for that entry in the menu.

For Example :

```
glutCreateMenu (menuFcn);
glutAddMenuEntry ("First Menu Item", 1);
glutAddMenuEntry ("Second Menu Item", 2);
glutAttachMenu (button);
```

We must specify a mouse button that is to be used to select a menu option.

OPENGL MENU FUNCTIONS

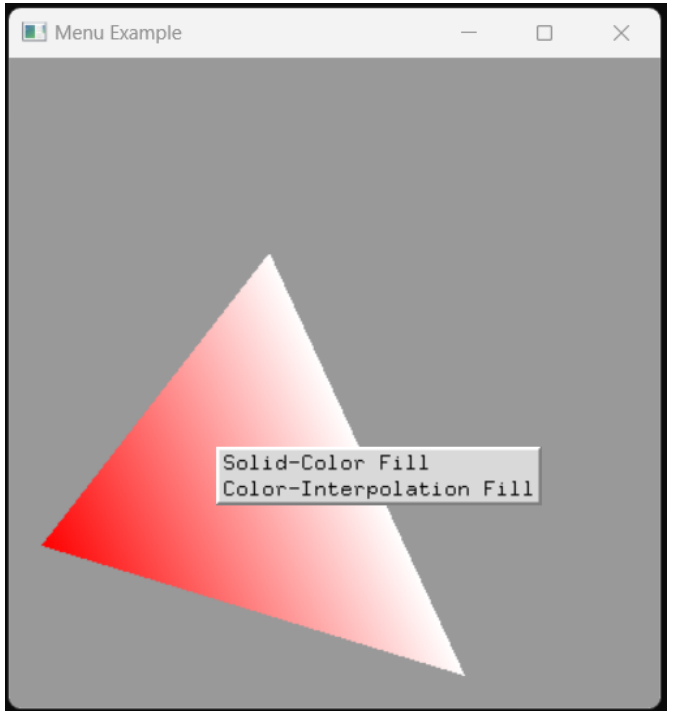
Example Program

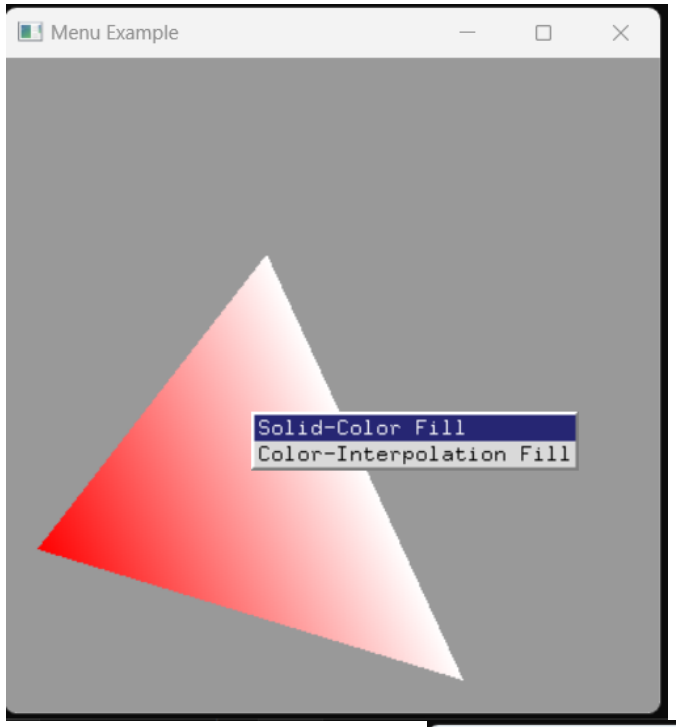
- The following program provides two options for displaying the interior fill of a triangle.
- Initially, the triangle is defined with two white vertices, one red vertex, and a fill color determined by an interpolation of the vertex colors.
- We use the glShadeModel function to select a polygon fill, that is either a solid color or interpolation(Gouraud rendering) of vertex colors.

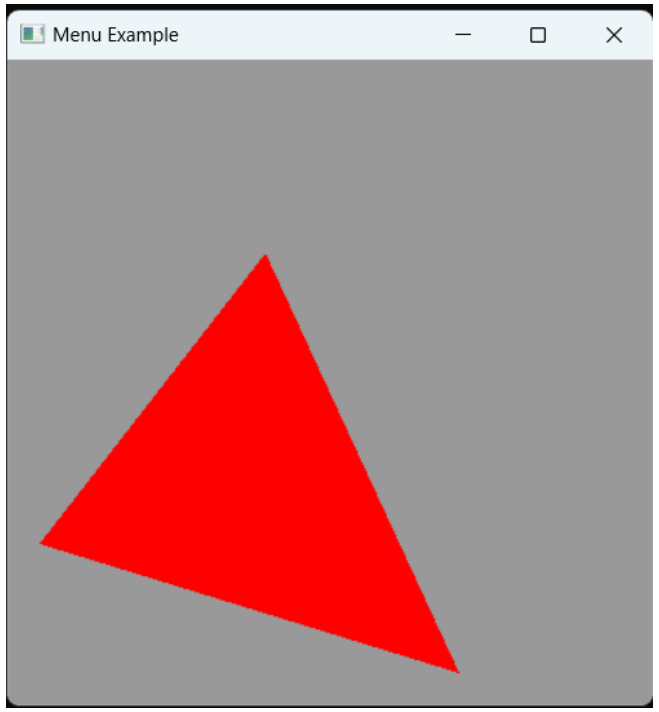
```
#include <GL/glut.h>
GLsizei winWidth = 400, winHeight = 400; // Initial display-window size.
GLfloat red = 1.0, green = 1.0, blue = 1.0; // Initial triangle color: white.
GLenum fillMode = GL_SMOOTH; // Initial polygon fill: color interpolation.
void init(void)
{
    glClearColor(0.6, 0.6, 0.6, 1.0); // Set display-window color to gray.
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0, 300.0, 0.0, 300.0);
}
void fillOption(GLint selectedOption)
{
    switch (selectedOption) {
    case 1: fillMode = GL_FLAT; break; // Flat surface rendering.
    case 2: fillMode = GL_SMOOTH; break; // Gouraud rendering.
    }
    glutPostRedisplay();
}
```

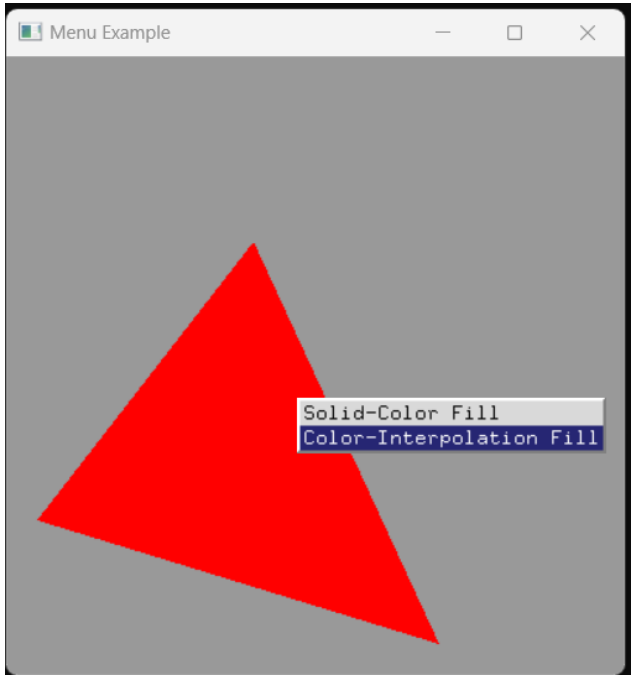
```
void displayTriangle(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    glShadeModel(fillMode); // Set fill method for triangle.
    glColor3f(red, green, blue); // Set color for first two vertices.
    glBegin(GL_TRIANGLES);
    glVertex2i(280, 20);
    glVertex2i(160, 280);
    glColor3f(red, 0.0, 0.0); // Set color of last vertex to red.
    glVertex2i(20, 100);
    glEnd();
    glFlush();
}
void reshapeFcn(GLint newWidth, GLint newHeight)
{
    glViewport(0, 0, newWidth, newHeight);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, GLfloat(newWidth), 0.0, GLfloat(newHeight));
    displayTriangle();
    glFlush();
}
```

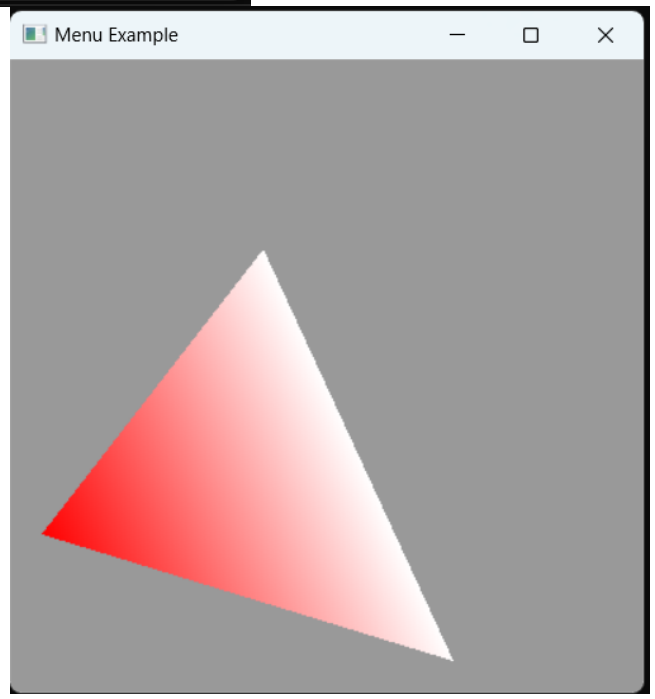
```
void main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowPosition(200, 200);
    glutInitWindowSize(winWidth, winHeight);
    glutCreateWindow("Menu Example");
    init();
    glutDisplayFunc(displayTriangle);
    glutCreateMenu(fillOption); // Create pop-up menu.
    glutAddMenuEntry("Solid-Color Fill", 1);
    glutAddMenuEntry("Color-Interpolation Fill", 2);
    /* Select a menu option using the right mouse button. */
    glutAttachMenu(GLUT_RIGHT_BUTTON);
    glutReshapeFunc(reshapeFcn);
    glutMainLoop();
}
```


Menu Example
Solid-Color Fill
Color-Interpolation Fill


Menu Example
Solid-Color Fill
Color-Interpolation Fill


Menu Example


Menu Example
Solid-Color Fill
Color-Interpolation Fill


Menu Example

OPENGL MENU FUNCTIONS

Creating and Managing Multiple GLUT Menus

- When a menu is created, it is associated with the current display window.
- We can create multiple menus for a single display window, and we can create different menus for different windows.
- As each menu is created, it is assigned an integer identifier, starting with the value 1 for the first menu created.
- The integer identifier for a menu is returned by the glutCreateMenu routine, and we can record this value with a statement such as:

```
menuID = glutCreateMenu (menuFcn);
```

OPENGL MENU FUNCTIONS

Creating and Managing Multiple GLUT Menus

- A newly created menu becomes the current menu for the current display window.
- To activate a menu for the current display window, we use the statement

glutSetMenu (menuID);

- We eliminate a menu with the command

glutDestroyMenu (menuID);

OPENGL MENU FUNCTIONS

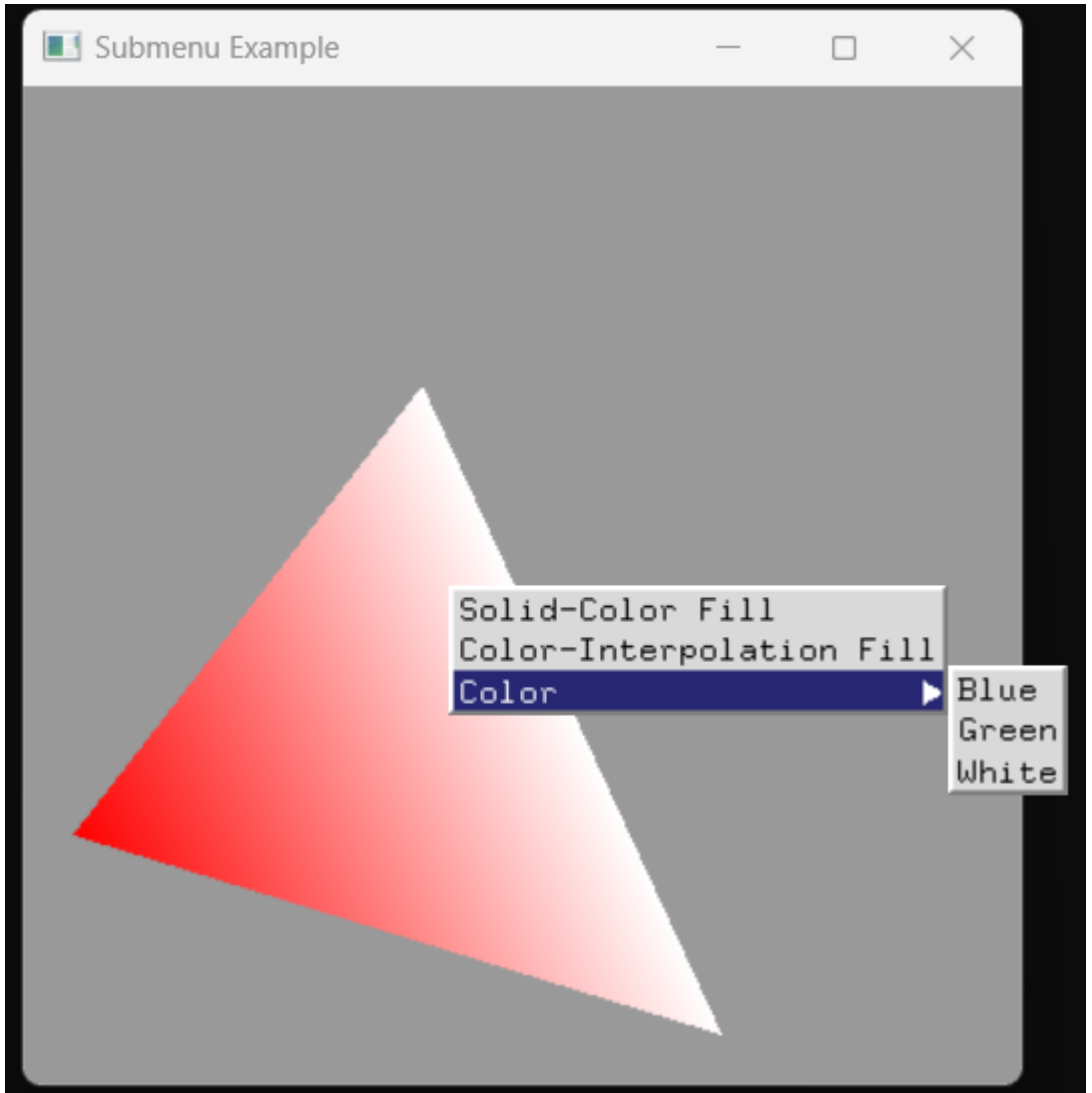
Creating GLUT submenus

- A submenu can be associated with a menu by first creating the submenu using glutCreateMenu, along with a list of suboptions, and then listing the submenu as an additional option in the main menu.

```
submenuID = glutCreateMenu (submenuFcn);
   glutAddMenuEntry ("First Submenu Item", 1);
   .
   .
   .
glutCreateMenu (menuFcn);
   glutAddMenuEntry ("First Menu Item", 1);
   .
   .
   .
   glutAddSubMenu ("Submenu Option", submenuID);
```

OPENGL MENU FUNCTIONS

Creating GLUT submenus



OPENGL MENU FUNCTIONS

Modifying GLUT Menus

- If we want to change the mouse button that is used to select a menu option, we first cancel the current button attachment and then attach the new button. A button attachment is cancelled for the current menu with

```
glutDetachMenu (mouseButton);
```

- where parameter mouseButton is assigned the GLUT constant identifying the button (left, middle, or right)
- Once we have detached the menu from the button, we can use `glutAttachMenu(mouseButton);` to attach it to a different button

DESIGNING A GRAPHICAL USER INTERFACE

- A common feature of modern applications software is a graphical user interface (GUI)
- GUI composed of display windows, icons, menus, and other features to aid a user in applying the software to a particular problem.

1)User Dialogue

✓The user dialogue in a GUI focusing on the communication between the user and the system.

✓It presents options and operations in a way that's easy to understand and use.

- **User's Model**:
- This is a simple description of what the software does and how users can use it.
- For example, in architectural design software, the user's model explains how to create and manipulate parts of a building like walls, doors, and windows.

DESIGNING A GRAPHICAL USER INTERFACE

Language of the Application

- The dialogue uses terms and concepts that are familiar to the user.
- For instance, architectural software will use terms like "wall," "window," and "door" instead of technical computer terms.

Operations:

- These are the actions users can take, such as adding, moving, or deleting objects.
- For example, in a furniture layout program, users can drag and drop furniture pieces into a floor plan.

Consistency and Feedback:

- The user dialogue should be consistent, meaning similar actions should work in similar ways throughout the application.
- The software should give feedback, like showing a message when an action is completed or if there's an error.

DESIGNING A GRAPHICAL USER INTERFACE

Example:

- Imagine using a kitchen design tool:
- **Model**: The tool helps you design a kitchen by placing cabinets, appliances, and countertops.
- **Language**: It uses words like "Cabinet," "Sink," and "Stove."
- **Operations**: You can click to add a cabinet, drag it to move it, or click a delete button to remove it.
- **Consistency and Feedback**: When you place a cabinet, it snaps into place, and you see a confirmation message. If you try to place it in an invalid spot, you get an error message.

DESIGNING A GRAPHICAL USER INTERFACE

2)Windows and Icons

- Icons representing objects such walls, doors, windows, and circuit elements are often referred to as application icons.
- The icons representing actions, such as rotate, magnify, scale, clip, or paste, are called control icons, or command icons.

DESIGNING A GRAPHICAL USER INTERFACE

3)Accomodating Multiple Skill Levels

- Graphical User Interfaces (GUIs) are designed to accommodate users with different levels of experience by offering various methods for performing actions.
- This ensures that both beginners and experienced users can interact with the software efficiently.

Beginners:

- **Simplified Interface:** A smaller, easy-to-understand set of operations helps beginners focus on using the application without getting overwhelmed by too many options.
- **Detailed Prompts:** Step-by-step instructions and prompts guide users through tasks.
- **Point-and-Click Operations:** Simple interactions, such as clicking buttons, are often the easiest for new users.

DESIGNING A GRAPHICAL USER INTERFACE

3)Accomodating Multiple Skill Levels

Experienced Users:

- **Speed and Efficiency:** Experienced users prefer fewer prompts and faster ways to perform tasks.
- **Keyboard Shortcuts:** Function keys or combinations of keys allow for quick action execution, which experienced users can remember and use frequently.
- **Advanced Options:** More complex features and multiple mouse-button clicks enable quicker navigation and task completion.

DESIGNING A GRAPHICAL USER INTERFACE

3)Accomodating Multiple Skill Levels

Adapting to User Skill Levels:

- **User-Selectable Settings:** Users can choose their preferred level of complexity through application settings.
- **Automatic Adjustments:** The application can suggest changes as it detects the user becoming more experienced.
- **Help Facilities:**
  - **Multiple Levels:** Beginners receive detailed guidance, while experienced users can opt for minimal or no prompts.
  - **Tutorials:** Built-in tutorials help users learn the system's capabilities and usage, providing an introduction for beginners and deeper insights for advanced users.

DESIGNING A GRAPHICAL USER INTERFACE

4)Consistency

- An important design consideration in an interface is consistency.
- An icon shape should always have a single meaning, rather than serving to represent different actions or objects depending on the context.
- Some other examples of consistency are always placing menus in the same relative positions so that a user does not have to hunt for a particular option.
- always using the same combination of keyboard keys for an action, and
- always using the same color encoding so that a color does not have different meanings in different situations.

DESIGNING A GRAPHICAL USER INTERFACE

5) Minimizing Memorization

- Operations in an interface should also be structured so that they are easy to understand and to remember.
- Obscure, complicated, inconsistent, and abbreviated command formats lead to confusion and reduction in the effective application of the software.
- One key or button used for all delete operations, for example, is easier to remember than a number of different keys for different kinds of delete procedure
- Icons and window systems can also be organized to minimize memorization.
- Different kinds of information can be separated into different windows so that a user can identify and select items easily.
- Icons should be designed as easily recognizable shapes that are related to application objects and actions

DESIGNING A GRAPHICAL USER INTERFACE

5) Backup and Error Handling

- A mechanism for undoing a sequence of operations is another common feature of an interface, which allows a user to explore the capabilities of a system, knowing that the effects of a mistake can be corrected.
- Typically, systems can now undo several operations, thus allowing a user to reset the system to some specified action.
- For those actions that cannot be reversed, such as closing an application without saving changes, the system asks for a verification of the requested operation.

DESIGNING A GRAPHICAL USER INTERFACE

6) Feed Back

- Responding to user actions is another important feature of an interface, particularly for an inexperienced user.
- As each action is entered, some response should be given. Otherwise, a user might begin to wonder what the system is doing and whether the input should be reentered
- Feedback can be given in many forms, such as highlighting an object, displaying an icon or message, and displaying a selected menu option in a different color.
- A cross, a frowning face, or a thumbs-down symbol is often used to indicate an error, and some kind of time symbol or a blinking "at work" sign is used to indicate that an action is being processed.
- This type of feedback can be very effective with a more experienced user, but the beginner may need more detailed feedback that not only clearly indicates what the system is doing but also what the user should input next.

COMPUTER ANIMATION

- Computer animation generally refers to any time sequence of visual changes in a picture.

- In addition to changing object positions using translations or rotations, a computer-generated animation could display time variations in object size, color, transparency, or surface texture.

COMPUTER ANIMATION

- Constructing an animation sequence can be a complicated task, particularly when it involves a story line and multiple objects, each of which can move in a different way.
- A basic approach is to design such animation sequences using the following development stages:

- Storyboard layout
- Object definitions
- Key-frame specifications
- Generation of in-between frames

COMPUTER ANIMATION

Storyboard Layout

- ✓The storyboard is an outline of the action.
- ✓It defines the motion sequence as a set of basic events that are to take place.
- ✓Depending on the type of animation to be produced, the storyboard could consist of a set of rough sketches, along with a brief description of the movements, or it could just be a list of the basic ideas for the action.
- ✓Originally, the set of motion sketches was attached to a large board that was used to present an overall view of the animation project. Hence, the name "storyboard."

COMPUTER ANIMATION

Object Definition

- An object definition is given for each participant in the action.
- Objects can be defined in terms of basic shapes, such as polygons or spline surfaces.
- In addition, a description is often given for the movements that are to be performed by each character or object in the story.

COMPUTER ANIMATION

Key Frame

- A key frame is a detailed drawing of the scene at a certain time in the animation sequence.
- Within each key frame, each object (or character) is positioned according to the time for that frame.
- More key frames are specified for complicated motions than for simple, slowly varying motions.
- Development of the key frames is generally the responsibility of the senior animators, and often a separate animator is assigned to each character in the animation.

COMPUTER ANIMATION

Generation of in between frames

- In-between frames are the intermediate frames between the key frames.
- The total number of frames, and hence the total number of in-betweens, needed for an animation is determined by the display media that is to be used.
- Typically, time intervals for the motion are set up so that there are from three to five in-betweens for each pair of key frames.
- Depending on the speed specified for the motion, some key frames could be duplicated.

TRADITIONAL ANIMATION TECHNIQUES

(or Techniques for showing motion in animation)

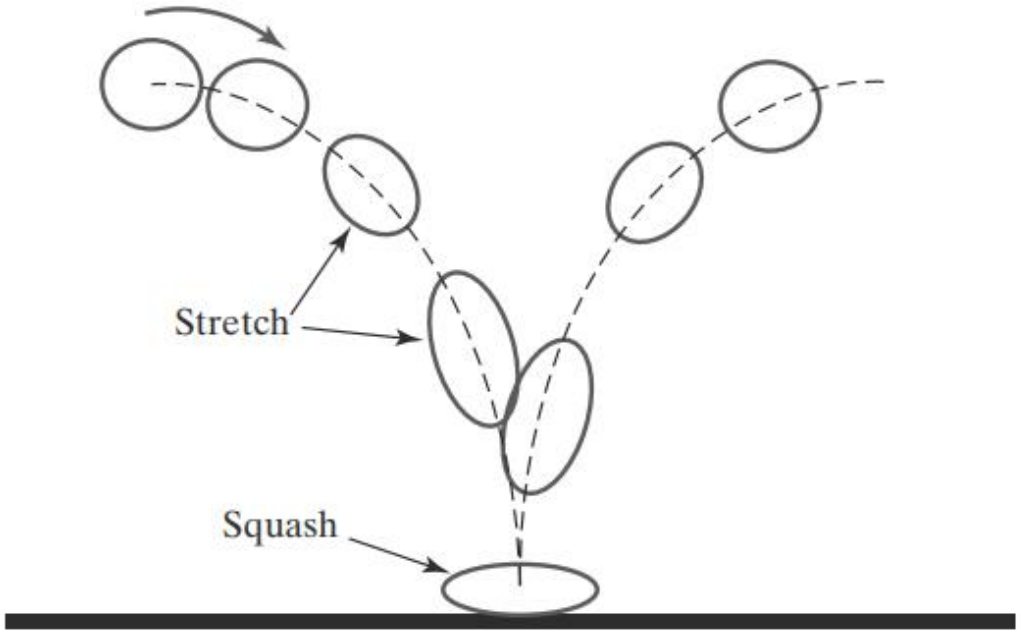
Animators use several tricks to make movements look more natural and interesting in films.

1. Squash and Stretch

This is used to show acceleration, especially for flexible objects. Imagine a bouncing ball:

- **As it speeds up, it gets longer (stretches).**
- **When it hits the ground, it flattens (squashes).**
- **Then it stretches again as it bounces back up.**

This makes the motion look more lively and dynamic.

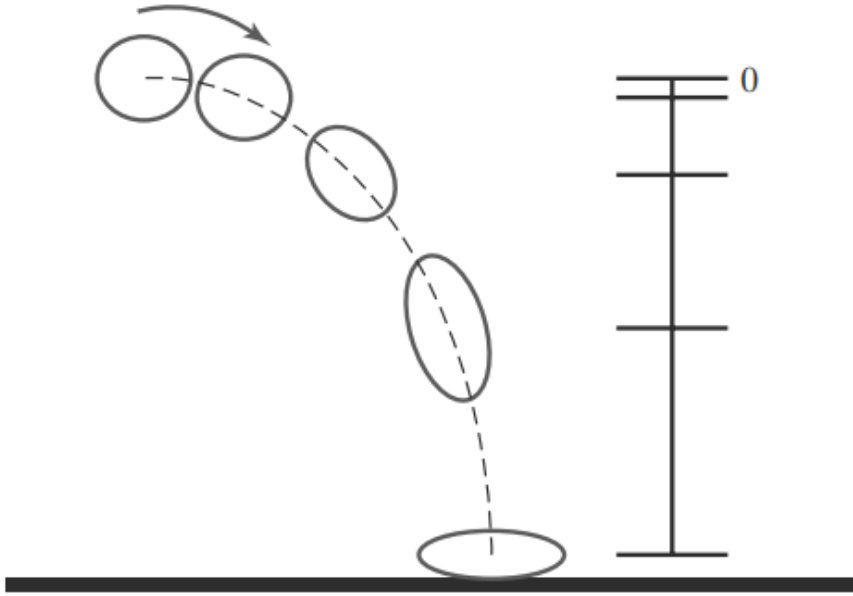


TRADITIONAL ANIMATION TECHNIQUES

2. Timing

- This is about how close or far apart the frames (individual pictures) are.
- Slow movements: Frames are closer together, so the changes between each frame are small.
- Fast movements: Frames are farther apart, so the changes between each frame are bigger.

This helps show different speeds of movement clearly.



TRADITIONAL ANIMATION TECHNIQUES

3. **Anticipation and Follow-Through**

These are actions that happen before and after the main motion to make it more believable.

Anticipation: The character does something before the main action to prepare for it. For example, a character might lean back before jumping forward.

Follow-Through: After the main action, the character continues moving a bit. For example, after throwing a ball, the character's arm keeps swinging.

TRADITIONAL ANIMATION TECHNIQUES

3. **Staging**

This focuses attention on important parts of the scene. It could be:

- Making sure the audience looks at where the main action or important object is.
- For example, a character might hide something in their hand to draw attention to it.

GENERAL COMPUTER-ANIMATION FUNCTIONS

Animation software helps animators create and manage animations. Here are four key functions that many animation programs provide:

Managing Object Motions:

- These tools let you control how objects move.
- You can specify movements in 2D or 3D, like rotating, scaling, or moving objects from one place to another.

Generating Views of Objects:

- This involves creating different views or angles of the objects.
- The software helps in rendering the object's surfaces, making them look realistic and identifying which parts of the objects are visible.

GENERAL COMPUTER-ANIMATION FUNCTIONS

Producing Camera Motions:

- Just like in movies, you can control virtual cameras in animation.
- Standard camera movements include zooming in/out, panning (moving the camera left or right), and tilting (moving the camera up or down).

Generating In-Between Frames:

- In animation, key frames are the main frames that define important positions of an object.
- The software can automatically create the frames between these key frames, called in-between frames, to make the motion smooth.

GENERAL COMPUTER-ANIMATION FUNCTIONS

Types of Animation Software

General Animation Design Software: These are comprehensive tools that provide all the functions mentioned above. They handle the overall design and details of individual objects.

For example, Wavefront is a package that offers a wide range of animation functions.

Specialized Animation Software: These tools focus on specific aspects of animation. For instance, some software might specialize in creating in-between frames or animating characters.

CHARACTER ANIMATION

- Animation of simple objects is relatively straightforward.
- When we consider the animation of more complex figures such as humans or animals, however, it becomes much more difficult to create realistic animation.
- Consider the animation of walking or running human (or humanoid) characters.
- Based upon observations in their own lives of walking or running people, viewers will expect to see animated characters move in particular ways.
- If an animated character's movement doesn't match this expectation, the believability of the character may suffer.
- Thus, much of the work involved in character animation is focused on creating believable movements.

CHARACTER ANIMATION

Articulated Figure Animation

- A basic technique for animating people, animals, insects is to model them as articulated figures, which are hierarchical structures composed of a set of rigid links that are connected at rotary joints.
- We model animatation objects as moving stick figures, or simplified skeletons, that can later be wrapped with surfaces representing skin, hair, fur, feathers, clothes, or other outer coverings.
- The connecting points, or hinges, for an articulated figure are placed at the shoulders, hips, knees, and other skeletal joints, which travel along specified motion paths as the body moves.
- For example, when a motion is specified for an object, the shoulder automatically moves in a certain way and, as the shoulder moves, the arms move.

CHARACTER ANIMATION

Articulated Figure Animation

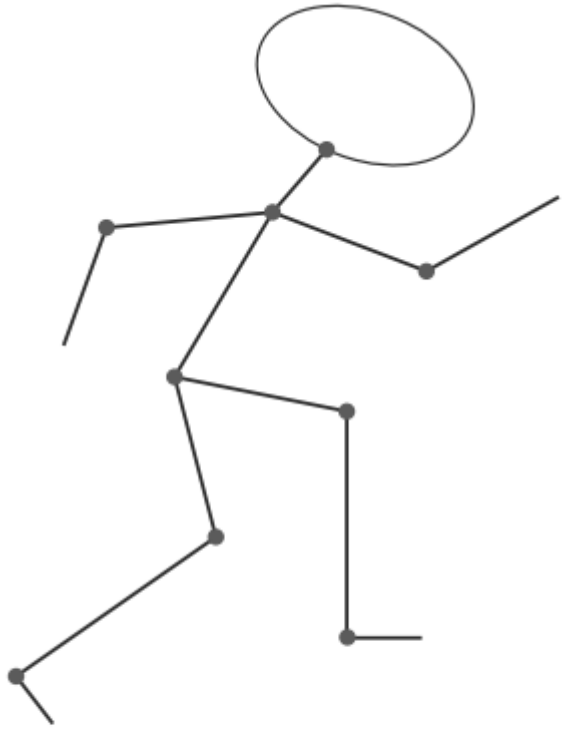


FIGURE 17
A simple articulated figure with nine joints and twelve connecting links, not counting the oval head.

CHARACTER ANIMATION

Articulated Figure Animation

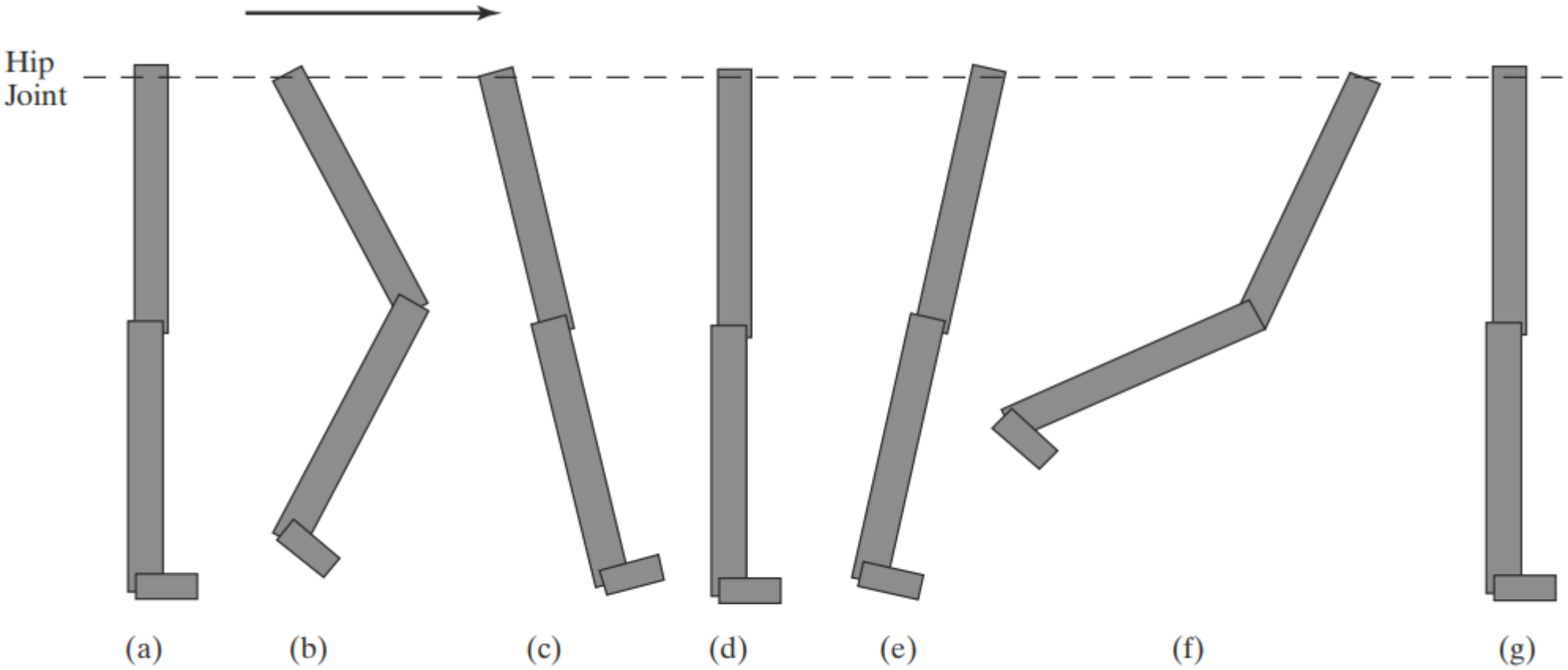


FIGURE 18
Possible motions for a set of connected links representing a walking leg.

CHARACTER ANIMATION

Motion Capture

Motion capture is a technique where the movements of a live actor are recorded digitally to animate a character. The animated character will mimic the actor's movements exactly.

- When the movement of the character is as in a scripted scene. The animated character will perform the same series of movements as the live actor.
- The classic motion capture technique involves placing a set of markers at strategic positions on the actor's body, such as the arms, legs, hands, feet, and joints.
- It is possible to place the markers directly on the actor, but more commonly they are affixed to a special skintight body suit worn by the actor.

CHARACTER ANIMATION

Motion Capture

- The actor is then filmed performing the scene.
- Image processing techniques are then used to identify the positions of the markers in each frame of the film, and their positions are translated to coordinates.
- These coordinates are used to determine the positioning of the body of the animated character

PERIODIC MOTIONS

- Periodic motions are repeated motion patterns, like a wheel rotating continuously.

Sampling Rate and Frame Rate:

Sampling Rate:

- This is how often the position of a moving object is recorded.
- In animations, you can control this rate to make the motion look smooth.

Frame Rate:

- This is the number of frames displayed per second.
- Standard film frame rate is 24 frames per second.

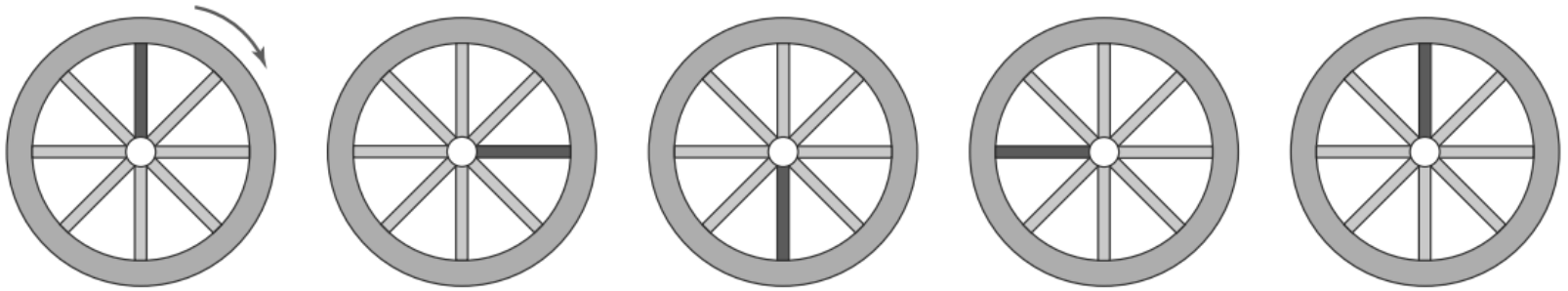
PERIODIC MOTIONS

Undersampling Example:

- When the sampling rate is too low, the motion can appear incorrect.
- A classic example is the wagon wheel in old Western movies:
  - The wheel spins at 18 revolutions per second.
  - At 24 frames per second, the wheel appears to turn in the opposite direction because it completes 3/4 of a turn every 1/24 second.
  - This effect is known as the "wagon wheel effect."

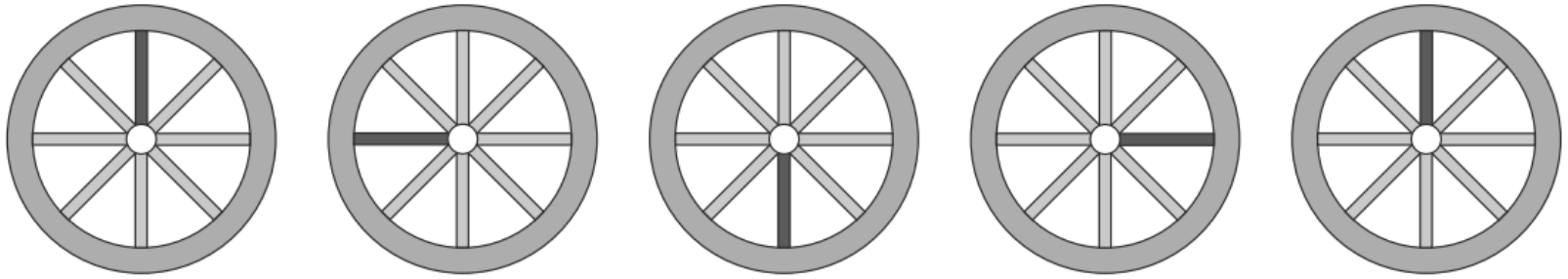
PERIODIC MOTIONS

Undersampling Example:



(a) 0 sec. (b) 1/72 sec. (c) 1/36 sec. (d) 1/24 sec. (e) 1/18 sec.

FIGURE 19
Five positions for a red spoke during one cycle of a wheel motion that is turning at the rate of 18 revolutions per second.



Frame 0 0 sec. Frame 1 1/24 sec. Frame 2 2/24 sec. Frame 3 3/24 sec. Frame 4 4/24 sec.

FIGURE 20
The first five film frames of the rotating wheel in Figure 19 produced at the rate of 24 frames per second.

OPENGL ANIMATION PROCEDURES

Most of the OPENGL animation procedures are included in GLUT.

1. **glutInitDisplayMode (GLUT_DOUBLE);**

- Double-buffering operations, if available, are activated using the above GLUT command:
- This provides two buffers, called the front buffer and the back buffer, that we can use alternately to refresh the screen display.
- While one buffer is acting as the refresh buffer for the current display window, the next frame of an animation can be constructed in the other buffer.

2. **glutSwapBuffers ( );**

roles of the two buffers can be interchanged using the above function.

OPENGL ANIMATION PROCEDURES

3. **glGetBooleanv (GL_DOUBLEBUFFER, status);**

- To determine whether double-buffer operations are available on a system, we can issue the above query:
- A value of GL TRUE is returned to array parameter status if both front and back buffers are available on a system. Otherwise, the returned value is GL FALSE.

OPENGL ANIMATION PROCEDURES

4. **glutIdleFunc (animationFcn);**

- For a continuous animation, we can also use the above function.
- **animationFcn** can be assigned the name of a procedure that is to perform the operations for incrementing the animation parameters.
- This procedure is continuously executed whenever there are no display-window events that must be processed.
- To disable the glutIdleFunc, we set its argument to the value NULL or the value 0.