COMPUTER GRAPHICS 334 CS

1434-35
Second Semester
UNIT 1 : INTRODUCTION

Computer Graphics is concerned with producing images and animations using computer hardware, software and applications.

A computer graphic is pictorial representation of information using a computer program.

Computer graphics generally means creation, storage and manipulation of models and images.

Computer Graphics have become a powerful tool for the rapid and economical production of pictures.

Today, computer graphics are used routinely in such diverse fields as science, art, engineering, business, industry, medicine, government, entertainment, advertising, education, training, and home applications.

**Pixel**

In Computer Graphics, pictures as graphical subjects are represented as a collection of discrete picture element called pixels.

Pixel is smallest addressable screen element.

![Pixel Diagram](image)

Pixel P is represented as (4, 3)
Basic Graphics System

Input devices  Image formed in FB  Output device

Applications of Computer Graphics

- Graphs and charts
- Computer-aided design
- Virtual-Reality Environments
- Data Visualization
- Education and Training
- Computer Art
- Entertainment
- Image processing
- Graphical User Interfaces

Graphs and charts

An early application for computer graphics is the display of simple data graphs, usually plotted on a character printer.

Data plotting is still one of the most common graphics applications, but now it is easy to generate graphs for highly complex data relationships.

Graphs and charts are commonly used to summarize financial, statistical, mathematical, scientific, engineering, and economic data for research reports, managerial summaries, consumer information bulletins, and other types of publications.
Computer –Aided Design

A major use of computer graphics is in design processes – particularly for engineering and architectural systems, although most products are now computer designed.

Generally referred to as CAD, computer-aided design or CADD computer aided drafting and design, these methods are now routinely used in the design of buildings, automobiles, aircraft, watercraft, spacecraft, computers, textiles, home appliances, and a multitude of other products.

For some design applications, objects are first displayed in a wire-frame outline that shows the overall shape and internal features of the objects.

Wire-frame displays also allow designers to quickly see the effects of interactive adjustments to design shapes without waiting for the object surfaces to be fully generated.

Virtual-reality environments

A more recent application of computer graphics is in the creation of virtual-reality environments in which a user can interact with the objects in a three-dimensional scene.

Specialized hardware devise provide three-dimensional viewing effects and allow the user to “pick up” objects in a scene
Data Visualizations

Producing graphical representations for scientific, engineering, and medical data sets and processes is another fairly new application of computer graphics, which is generally referred to as scientific visualization.

The term business visualization is used in connection with data sets related to commerce, industry, and other nonscientific areas.

Education and Training

Computer-generated models of physical, financial, political, social, economic, and other systems are often used as educational aids.

Models of physical processes, physiological functions, population trends, or equipment, such as the color-coded diagram, can help trainees to understand the operations of a system.

For more training applications, special hardware systems are designed. Examples of such specialized systems are the simulators for practice sessions or training of ship captains, aircraft pilots, heavy-equipment operators, and air traffic-control personnel.

Computer Art

Both fine art and commercial art make use of computer-graphics methods.

Artists now have available a variety of computer methods and tools, including specialized hardware, commercial software packages (such as Lumena), symbolic mathematics programs (such as Mathematica), CAD packages, desktop publishing software, and animation systems that provide facilities for designing object shapes and specifying object motions.
**Entertainment**

Television productions, motion pictures, and music videos routinely use computer-graphics methods.

Sometimes graphics images are combined with live actors and scenes, and sometimes the films are completely generated using computer-rendering and animation techniques.

**Image processing**

Modification or interpretation of existing images or pictures such as photographs, TV scans is called image processing.

Example Improve picture quality, analyze images, or recognize visual patterns for robotics application. Also used extensively in medical applications

**Graphical user Interfaces**

Computer graphics is an integral part of everyday computing. Computers use graphics in order to present output to users.

It is common now for applications software to provide a graphical user interface (GUI). A major component of a graphical interface is a window manager that allows a user to display multiple, rectangular screen areas, called display windows.
UNIT 2 : Overview of a Graphics systems

VIDEO DISPLAY DEVICES

Cathode Ray Tube – CRT

Basic Operation

A beam of electrons (i.e. cathode rays) is emitted by the electron gun, it passes through focusing and deflection systems that direct the beam towards the specified position on the phosphor screen. The phosphor then emits a small spot of light at every position contacted by the electron beam. Since light emitted by the phosphor fades very quickly some method is needed for maintaining the screen picture. One of the simplest way to maintain pictures on the screen is to redraw the image rapidly. This type of display is called Refresh CRT.

Components of CRT:

1. Heater Element and Cathode

Heat is supplied to the cathode by passing current through heater element. Cathode is cylindrical metallic structure which is rich in electrons. On heating the electrons are released from cathode surface.
2. **Control Grid**

It is the next element which follows cathode. It almost covers cathode, leaving small opening for electrons to come out. Intensity of the electron beam is controlled by setting voltage levels on the control grid. A high negative voltage applied to the control grid will shut off the beam by repelling electrons and stopping them from passing through the small hole at the end of control grid structure. A smaller negative voltage on the control grid will simply decrease the number of electrons passing through. Thus we can control the brightness of a display by varying the voltage on the control grid.

3. **Accelerating Anode**

They are positively charged anodes which accelerate the electrons towards phosphor screen.

4. **Focusing and deflection coils**

They are together needed to force the electron beam to converge into a small spot as it strikes the screen otherwise the electrons would repel each other and the beam would spread out as it approaches the screen. Electrostatic focusing is commonly used in television and computer graphics monitor.

5. **Phosphor Coating**

When the accelerating electron beam collides with the phosphor screen, a part of kinetic energy is converted into light and heat. When the electrons in the beam collide with the phosphor coating they are stopped and their kinetic energy is absorbed by the phosphor.

There are two techniques for producing images on the CRT screen.

1. Raster Scan Displays.
2. Random Scan Displays.
1. Raster Scan Displays

The most common type of graphics monitor employing a CRT is the raster-scan display, based on television technology.

In a raster-scan system, the electron beam is swept across the screen, one row at a time, from the top to bottom. Each row is referred to as a scan line.

As the electron beam moves across a scan line, the beam intensity is turned on and off to create a pattern of illuminated spots.

Picture definition is stored in a memory area called the refresh buffer or frame buffer, where the term frame refers to the total screen area.

These stored color values are retrieved from the refresh buffer and used to control the intensity of the electron beam as it moves from spot to spot across the screen. In this way, picture is painted on the screen one line at a time as shown below.

The resolution of a raster-system is defined as the number of pixel positions that can be plotted.

Aspect ratio of a raster-system is the number of pixel columns divided by the number of scan lines that can be displayed by the system or the aspect ratio of an image is its width divided by its height.
Thus, an aspect ratio of 4:3, for example means that a horizontal line plotted with four points has the same length as vertical line plotted with three points.

The range of colors or shades of gray that can be displayed on a raster system depends on both the types of phosphor used in the CRT and the number of bits per pixel available in the frame buffer.

For a simple black and white system, each screen point is either on or off, so only one bit per pixel is needed to control the intensity of the screen positions.

A bit value of 1, for example, indicates that the electron beam is to be turned on at that position, and a value of 0 turns the beam off.

**Interlacing**

On some raster-scan systems, each frame is displayed in two passes using an interlaced refresh procedure.

First, all points on the even-numbered (solid) scan lines are displayed; and then all points along the odd-numbered (dashed) lines are displayed.

<table>
<thead>
<tr>
<th>Interlaced Scan</th>
<th>Progressive Scan</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

**2. Random scan displays**

When operated as a random-scan display unit, a CRT has the electron beam directed only to those parts of the screen where a picture is to be displayed.

Pictures are generated as line drawings, with the electron beam tracing out the component lines one after the other.
Picture definitions are stored as set of line-drawing commands in an area of memory referred to as the display list.

To display a specified picture, the system cycles through the set of commands in the display list, drawing each component line in turn.

After all line-drawing commands have been processed, the system cycles back to the first line command in the list.

Random-scan systems were designed for line-drawing applications, such as architectural and engineering layouts, and they cannot display realistic shaded scenes.

Since picture definition is stored as a set of line-drawing instructions rather than as set of intensity values for all screen points, random displays generally have higher resolutions than raster systems.

Random displays produce smooth line drawings because the CRT beam directly follows the line path.

Raster displays produce jagged lines that are plotted as discrete point sets.

**Display File**

The commands present in the display file contain two fields, an operation code (opcode) and operand. Opcode identifies the commands such as line draw, move cursors, etc and the operands provide the co-ordinate of a point to process the commands.

One way to store opcode and operands of series of commands is to use to separate arrays, one for opcode, one for x-coordinate and one for y-co-ordinate of the
operand. It is also necessary to assign meaning to the possible opcodes before we can proceed to interpret them.

<table>
<thead>
<tr>
<th>e.g.</th>
<th>COMMAND</th>
<th>OPCODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOVE</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>LINE</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

**Differences between Raster and Random Scan Displays**

<table>
<thead>
<tr>
<th>RASTER SCAN DISPLAY</th>
<th>RANDOM SCAN DISPLAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>It draws the image by scanning one row at a time.</td>
<td>It draws the image by directing the electron beam directly to the part of the screen where the image is to be drawn.</td>
</tr>
<tr>
<td>They generally have resolution limited to</td>
<td>They have higher resolution than raster</td>
</tr>
<tr>
<td>pixel size.</td>
<td>scan system</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Lines are jagged and curves are less smoother</td>
<td>Line plots are straight and curves are smooth.</td>
</tr>
<tr>
<td>They are more suited to geometric area drawing applications e.g. monitors, television.</td>
<td>They are more suited to line drawing application e.g. CRO's, pen plotter.</td>
</tr>
</tbody>
</table>

### 3. Color CRT Monitors

A CRT monitor displays color pictures by using a combination of phosphors that emit different-colored light.

The emitted light from the different phosphors merges to form a single perceived color, which depends on the particular set of phosphors that have been excited.

**Beam-penetration method**

This technique is used in Random Scan Monitors. In this technique, the inside of CRT is coated with two layers of phosphor, usually red & green.

The displayed color depends on how far the electron beam penetrates into the phosphor layers. The outer layer is of red phosphor and inner layer is of green phosphor. A beam of slow electrons excites only the outer red layer.

A beam of fast electrons penetrates the outer red layer and excites the inner green layer. At intermediate beam speeds, combination of red and green light is emitted and two additional colors orange and yellow are displayed.

The beam acceleration voltage controls the speed of the electrons and hence the screen color at any point on the screen.

**Shadow-mask method**

The shadow mask technique produces a much wider range of colors than the beam penetration technique. Hence this technique is commonly used in raster scan displays including color T.V.

In shadow mask technique, the CRT screen has three phosphor color dots at each pixel position. One phosphor dot emits red light, another emits a green light
and the third one emits a blue light. The CRT has three electron guns one for each dot, a shadow mask grid just behind the phosphor coated screen.

The shadow mask grid consists of series of holes aligned with the phosphor dot patterns. As shown in figure, the three electron beams are deflected and focused as a group onto the shadow mask and when they pass through a hole onto a shadow mask they excite a dot triangle.

A dot triangle consists of 3 small phosphor dots of red, green and blue color. These phosphor dots are arranged so that each electron beam can activate only its corresponding color dot when it passes through the shadow mask.

A dot triangle when activated appears as a small dot on the screen which has color combination of three small dots in the dot triangle. By varying the intensity of the three electron beams we can obtain different colors in the shadow mask CRT.

![Diagram of CRT with electron guns, shadow mask, and phosphor triangle.]

### 4. Flat Panel Displays

The term flat-panel display refers to a class of video devices that have reduced volume, weight, and power requirements compared to a CRT.

A significant feature of flat-panel display is that they are thinner than CRTs, and we can hang them on walls or wear them on our wrists.

We can even write on some flat-panel displays, they are also available as pocket notepads. There are two categories of flat-panel displays:
1. Emissive displays
2. Non emissive displays

1. Emissive displays:
   They convert electrical energy into light. Plasma panels, thin-film Electro luminescent and light emitting diodes are examples of emissive displays.

2. Non emissive displays:
   They use optical effects to convert sunlight or light from some other source into graphics patterns.

**Plasma Panels:**

It has narrow plasma tubes that are lined up together horizontally to make a display. The tubes, which operate in the same manner as standard plasma displays, are filled with xenon and neon gas.

Their inside walls are partly coated with either red, green, or blue phosphor, which together produce the full color spectrum.

The tubes are packed together vertically and are sandwiched between two thin and lightweight glass or plastic retaining plates.
The display electrodes in the tubular display run across its front, perpendicular to the tubes, while the address electrodes are on the back, parallel to the tubes.

When current runs through any pair of intersecting display and control electrodes, an electric charge prompts gas in the tube to discharge and emit ultraviolet light at the intersection point, which in turn causes the phosphor coating to emit visible light.

The combination of three tubes at any corresponding intersection point defines a pixel, and by varying the pulses of applied voltage in the underlying control electrodes, the intensity of each pixel's color can be regulated to produce myriad combinations of red, blue, and green.

**Thin-film Electro Luminescent Displays:**

They are similar in construction to plasma panels. The difference is that the region between the glass plate is filled with a phosphor, such as zinc sulfide doped with manganese, instead of gas.

When sufficiently high voltage is applied to a pair of crossing electrodes, the phosphor becomes a conductor in the area of the intersection of the two electrodes.

Electrical energy is absorbed by the manganese atoms, which then release the energy as a spot of light similar to the glowing plasma effect in a plasma panel.
**Liquid Crystal Display**

They are commonly used in small systems, such as laptop computers and calculators. They are non emissive devices.

They produce a picture by passing polarized light from the surroundings or from an internal light source through a liquid-crystal material that can be aligned to either block or transmit the light.

A simple black - or - white LCD display works by either allowing daylight to be reflected back out at the viewer or preventing it from doing so - in which case the viewer sees a black area. The liquid crystal is the part of the system that either prevents light from passing through it or not. The crystal is placed between two polarizing filters that are at right angles to each other and together block light. When there is no electric current applied to the crystal, it twists light by 90 oo, which allows the light to pass through the second polarizer and be reflected back. But when the voltage is applied, the crystal molecules align themselves, and light cannot pass through the polarizer: the segment turns black. Selective application of voltage to electrode segments creates the digits we see.
UNIT 3:  Input Devices

Graphics workstations can make use of various devices for data input. Most systems have a keyboard and one or more additional devices specifically designed for interactive input. These include a mouse trackball, space ball, and joystick. Some other input devices used in particular applications are digitizers, dials, button boxes, data gloves, touch panels, image scanners and voice systems.

**Physical Devices**

- mouse
- trackball
- light pen
- data tablet
- joy stick
- space ball

**Keyboards, Button Boxes, and Dials**

An alphanumeric keyboard on a graphics system is used primarily as a device for entering text strings, issuing certain commands, and selecting menu options.

The keyboard is an efficient device for inputting such non-graphic data as picture labels associated with a graphics display.

When we press a key on the keyboard, the keyboard controller places a code corresponding to the key pressed, in a part of its memory called keyboard buffer. This code is called the scan code. The keyboard controller informs the CPU of the computer about the key pressed with the help of interrupt signals. The CPU then reads the scan code from the Keyboard Buffer.

For specialized tasks, input to a graphics application may come from a set of buttons, dials.
Button and switches are often used to input predefined functions, and dials are common devices for entering scalar values.

**Trackballs and Spaceballs**

A trackball is a ball device that can be rotated with the fingers or palm of the hand to produce screen-cursor movement.

Potentiometers, connected to the ball, measure the amount and direction of rotation. Laptop keyboards are often equipped with a trackball to eliminate the extra space required by a mouse.

An extension of the two dimensional trackball concept is the spaceball, which provides six degrees of freedom.

Unlike the trackball, a spaceball does not actually move. Strain gauges measure the amount of pressure applied to the spaceball to provide input for spatial positioning and orientation as the ball is pushed or pulled in various directions.

Spaceball are used for 3-D positioning and selection operations in virtual-reality systems, modeling, animation, CAD and other applications.

**Joysticks**

![Joystick Diagram]

Another positioning device is the joystick, which consists of a small vertical lever mounted on a base. Most joystick, select screen positions with actual stick movement others respond to pressure on the stick.

A joystick has a small vertical lever mounted on the base and used to steer the screen cursor around. It consists of two potentiometers attached to a single lever. Moving the liver changes the settings on the potentiometer. The left or right movement is indicated by one potentiometer & the forward or backward movement is indicated by other potentiometer. Thus with a joystick box x & y co-ordinate positions can be simultaneously altered by the motion of a single lever.

Some joysticks may return to this zero (centre) positions when released. Joysticks are inexpressive and quiet commonly used where only rough positioning is needed.
**Data Gloves**

Data gloves can be used to grasp a virtual object. The glove is constructed with a series of sensors that detect hand and finger motions.

Electromagnetic coupling between transmitting antennas and receiving antennas are used to provide information about the position and orientation of the hand.

![Data Gloves Image](http://example.com/data_gloves.jpg)

**Digitizers**

A common device for drawing, painting, or interactively selecting positions is a digitizer. These devices can be designed to input coordinate values in either a two dimensional or a three dimensional space.

In engineering or architectural applications, a digitizer is often used to scan a drawing or object and to input a set of discrete coordinate positions.

**Image Scanners**

![Image Scanners Diagram](http://example.com/image_scanners.png)

Drawings, graphs, photographs, or text can be stored for computer processing with an image scanner by passing an optical scanning mechanism over the information to be stored.

As shown in the figure the photograph is mounted on a rotating drum. A fine light beam is directed at the photo and the amount of light reflected is measured by a photocell. As the drum rotates, the light source slowly moves from one end to the other, thus doing a raster scan of the inter photograph.
We can also apply various image-processing methods to modify the array representation of the picture.

**Touch Panels**

Touch panels allow displayed object or screen positions to be selected with the touch of a finger.

A typical application of touch panels is for the selection of processing options that are represented as a menu of graphical icons.

Some monitors, such as plasma panels are designed with touch screens.

![Plasma panels with touch screen](image1.png) ![Touch screen overlay](image2.png)

**Light Pen**

Pencil shaped devices are used to select screen positions by detecting the light coming from points on the CRT screen.

They are sensitive to the short burst of light emitted from the phosphor coating at the instant the electron beam strikes a particular point.

![Light pen](image3.png)

**Voice System**

Speech recognizers are used with some graphics workstations as input devices for voice commands.

The voice system input can be used to initiate graphics operations or to enter data. These systems operate by matching an input against a predefined dictionary of words and phrases.
UNIT 4: Graphics Output Primitives

Line Drawing Algorithms

A straight line segment in a scene is defined by the coordinate positions for the endpoints of the segment.

To display the line on a raster monitor, the graphics system must first project the endpoints to integer screen coordinates and determine the nearest pixel positions along the line path between the two endpoints.

Rasterization

As a cathode ray tube (CRT) raster display is considered a matrix of discrete finite area cells (pixels), each of which can be made bright, it is not possible to directly draw a straight line from one point to another.

The process of determining which pixels provide the best approximation to the desired line is properly known as rasterization.
**Digital Differential Analyzer Algorithm**

One technique for obtaining a rasterized straight line is to solve the differential equation for a straight line, i.e.

If we have to draw a line from the point \((x_1,y_1)\) to \((x_2,y_2)\) then let us assume

\[
\text{Length} = \text{abs}(x_2 - x_1)
\]

or

\[
\text{Length} = \text{abs}(y_2 - y_1)
\]

dx as small increment along x

dy as small increment along y

\[
\frac{dy}{dx} = \frac{y_2 - y_1}{x_2 - x_1}
\]

There fore we get \(x_{i+1}\) from \(x_i\) as

\[
x_{i+1} = x_i + dx
\]

and we get \(y_{i+1}\) from \(x_i\) as

\[
y_{i+1} = y_i + dy
\]
The actual DDA algorithm is as follows

```plaintext
if abs(x2-x1) >= abs(y2-y1) then
    Length = abs(x2-x1)
else
    Length = abs(y2-y1)
end if

dx = (x2-x1)/Length
dy = (y2-y1)/Length

x = x1  //the value of x
y = y1  //the value of y

begin
    i=1
    while ( i <= Length)
        glVertex2i(x,y)
        x=x+dx
        y=y+dy
        i=i+1
    end while
end
```

**Example:**

Consider the line from (0,0) to (5,5). Use the simple DDA to rasterize this line. Evaluating the steps in the algorithm yields initial calculation as

x1 = 0 ; y1 = 0 ; x2 = 5 ; y2 = 5 ; Length = 5 ; dy = 1 ; dx = 1 ; x= 0 ; y= 0

Incrementing through the main loop yields

<table>
<thead>
<tr>
<th>i</th>
<th>glVertex</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(0,0)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>(1,1)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>(2,2)</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>(3,3)</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>(4,4)</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>
**Bresenham’s Line Algorithm**

The algorithm seeks to select the optimum raster locations that represent a straight line. To accomplish this, the algorithm always increments by one unit either x or Y, depending on the slope of the line.

The increment in the other variable, either zero or one, is determined by examining the distance between the actual line and the nearest grid locations. This distance is called the error.

For example, in the above diagram after rastering the pixel at (0,0) we have to choose whether we have to raster pixel (1,0) or (1,1).

If slope of the required line through (0,0) is greater than ½, then raster point at (1,1) and if it is less than ½, then raster (1,0).

That is,

\[
\text{If } \frac{1}{2} \leq \frac{\Delta y}{\Delta x} \leq 1 \text{ then } \quad (\text{error} \geq 0)
\]

\[
\text{plot}(1,1)
\]

\[
\text{else if } 0 \leq \frac{\Delta y}{\Delta x} < \frac{1}{2} \text{ then } \quad (\text{error} < 0)
\]

\[
\text{plot}(1,0)
\]

end if
If we have to draw a line from \((x_1,y_1)\) to \((x_2,y_2)\) then Bresenham algorithm is

**Algorithm:**

1. Input two endpoints \(x_1, y_1, x_2, y_2\)
2. Calculate \(dx = x_2 - x_1, dy = y_2 - y_1\)
3. Obtain starting value for the decision variable
   \[ p_0 = 2*dy - dx \]
4. At each \(x_k\) along the line, starting at \(k=0\) perform the following test.
5. If \((p_k < 0)\), the next point to plot is
   
   
   \[
   \begin{cases}
   (x_k + 1, y_k) \\
   p_{k+1} = p_k + 2*dy
   \end{cases}
   \]
6. Else
   
   \[
   \begin{cases}
   (x_k + 1, y_{k+1}) \\
   p_{k+1} = p_k + 2*dy - 2*dx
   \end{cases}
   \]
7. End if

6. Perform step 4 \(dx-1\) times.
**Example:**
To draw line from (20,10) to (30,18);

\[ x_1 = 20; \quad y_1 = 10; \]
\[ x_2 = 30; \quad y_2 = 18; \]
\[ dx = \text{fabs}(x_2 - x_1) = 30-20 = 10; \]
\[ dy = \text{fabs}(y_2 - y_1) = 18-10 = 8; \]
\[ p_0 = 2 \times dy - dx = 2 \times 8 - 10 = 6; \]

<table>
<thead>
<tr>
<th>k</th>
<th>p</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>6</td>
<td>21</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>22</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>-2</td>
<td>23</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>24</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>25</td>
<td>14</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>26</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>27</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>-2</td>
<td>28</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>14</td>
<td>29</td>
<td>17</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>30</td>
<td>18</td>
</tr>
</tbody>
</table>
**Circle Generating Algorithm**

*Properties of Circles*

A Circle is defined as the set of points that are all at a given distance \( r \) from a center position \((x_c, y_c)\). For any circle point \((x, y)\), this distance relationship is expressed as

\[
(x - x_c)^2 + (y - y_c)^2 = r^2
\]

Another way to calculate points along the circular boundary is to use polar coordinates \( r \) and \( \theta \).

Expressing the circle equation in parametric polar form yields the pair of equations

\[
\begin{align*}
x &= x_c + r \cos \theta \\
y &= y_c + r \sin \theta
\end{align*}
\]

**Midpoint Circle Algorithm**

For a given radius \( r \) and screen center position \((x_c, y_c)\), we have to calculate pixel positions around a circular path centered at the coordinate origin \((0,0)\).

The equation for the circle is

\[
F(x, y) = x^2 + y^2 - R^2
\]

Any point \((x, y)\) on the boundary of the circle with radius \( R \) satisfies the equation \( F(x, y) = 0 \).

If the value of \( F(x, y) \) is positive then that point lies outside the circle and if it is negative then that point lies inside the circle.

Therefore if the value of \( F(M) \) is

- Positive: \( M \) is outside the circle and the next pixel is SE.
- Negative: \( M \) is inside the circle and the next pixel is E.
We decide the next pixel to be selected depending upon the value of the decision variable \( d \) as

\[
d_{\text{old}} = F(x_{p}+1, y_{p}-1/2) = (x_{p}+1)^2 + (y_{p}-1/2)^2 - R^2
\]

**Case 1:** If \( d_{\text{old}} \) is <0 then

- the pixel E is chosen and
- the next midpoint will be \((x_{p}+2, y_{p}-1/2)\).
- the value of the next decision variable will be:

\[
d_{\text{new}} = F(x_{p}+2, y_{p}-1/2) = (x_{p}+2)^2 + (y_{p}-1/2)^2 - R^2
\]

- the difference between \( d_{\text{old}} \) and \( d_{\text{new}} \) will be

\[
d_{\text{new}} - d_{\text{old}} = (x_{p}+2)^2 + (y_{p}-1/2)^2 - R^2 - [(x_{p}+1)^2 + (y_{p}-1/2)^2 - R^2]
\]

\[
= 2x_p + 3
\]

**Proof:**

\[
d_{\text{old}} = F(X_p + 1, Y_p - 1/2) = (X_p + 1)^2 + (Y_p - 1/2)^2 - R^2
\]

\[
d_{\text{new}} = F(X_p + 2, Y_p - 1/2) = (X_p + 2)^2 + (Y_p - 1/2)^2 - R^2
\]

\[
d_{\text{new}} - d_{\text{old}} = 2x_p + 3
\]

Hence, we prove that \( d_{\text{new}} - d_{\text{old}} = 2x_p + 3 \)
**Case 2**: If \( d_{\text{old}} \) is \( \geq 0 \) then

- the pixel SE is chosen and
- the next midpoint will be \((x_p+2, y_p-3/2)\).
- the value of the next decision variable will be:
  \[
  d_{\text{new}} = F(x_p+2, y_p-3/2)
  = (x_p+2)^2 + (y_p-3/2)^2 - R^2
  \]
- the difference between \( d_{\text{old}} \) and \( d_{\text{new}} \) will be
  \[
  d_{\text{new}} - d_{\text{old}} = (2x_p - 2y_p + 5)
  \]
- therefore \( \Delta SE = 2x_p - 2y_p + 5 \)

\[
\begin{align*}
  d_{\text{old}} &= F(X_p + 1, Y_p - 1/2) \\
  &= (X_p + 1)^2 + (Y_p - 1/2)^2 - R^2 \\
  &= (X_p^2 + 2*1*X_p + 1^2) + (Y_p^2 - 2*(1/2)*Y_p + (1/2)^2) - R^2 \\
  &= (X_p^2 + 2*X_p + 1) + (Y_p^2 - Y_p + 1/4) - R^2 \\
  \\
  d_{\text{new}} &= F(X_p + 2, Y_p - 3/2) \\
  &= (X_p + 2)^2 + (Y_p - 3/2)^2 - R^2 \\
  &= (X_p^2 + 2*2*X_p + 2^2) + (Y_p^2 - 2*(3/2)*Y_p + (3/2)^2) - R^2 \\
  &= (X_p^2 + 4*X_p + 4) + (Y_p^2 - 3Y_p + 9/4) - R^2 \\
  \\
  d_{\text{new}} - d_{\text{old}} &= (X_p^2 + 4*X_p + 4) + (Y_p^2 - 3Y_p + 9/4) - R^2 - [(X_p^2 + 2*X_p + 1) + (Y_p^2 - Y_p + 1/4) - R^2] \\
  &= X_p^2 + 4*X_p + 4 + Y_p^2 - 3Y_p + 9/4 - R^2 - X_p^2 - 2X_p - Y_p^2 + Y_p - 1/4 + R^2 = 2X_p - 2Y_p + 5 \\
  \text{Hence, we prove that} \quad d_{\text{new}} - d_{\text{old}} = 2X_p - 2Y_p + 5
\end{align*}
\]

The initial decision variable is based on the initial pixel location \((0, R)\) and the first midpoint \((1, R-1/2)\).

Therefore,

\[
\begin{align*}
  d_0 &= F(1, R-1/2) \\
  &= (1)^2 + (R - 1/2)^2 - R^2 \\
  &= 1 + (R^2 - R + 1/4) - R^2 \\
  &= (5/4) - R
\end{align*}
\]
The actual algorithm is as follows

//initialize variables
x = 0
y = radius
d = ( 5.0 / 4.0 ) – radius

glfwVertex2f ( x, y );

While ( y > x )
{
    if ( d < 0 )
        {
            d = d + ( 2 * x + 3 )
            x= x+ 1
        }
    else
        {
            d = d + (2 * x - 2 * y + 5 )
            x= x+ 1
            y= y-1
        }
    glfwVertex2f ( x, y );
}
UNIT 5: Geometric Transformation.

A transformation $T$ is the process of moving a point in space from one position to another.

The functions that are available in all graphics packages are those for translation, rotation and scaling.

Basic Two-Dimensional Geometric Transformations.

Two Dimensional Translation.

Translation on a single coordinate point is done by adding offsets to its coordinates so as to generate a new coordinate position.

To translate a two-dimensional position, we add translation distances $tx$ and $ty$ to the original coordinates $(x,y)$ to obtain the new coordinate position $(x',y')$.

$$x' = x + tx$$
$$y' = y + ty$$

The translation distance pair $(tx,ty)$ is called a translation vector or shift vector.

We can express the translation as a single matrix equation by using the following column vectors to represent the coordinate positions and the translation vector.

$$P = \begin{bmatrix} x \\ y \end{bmatrix}, \quad P' = \begin{bmatrix} x' \\ y' \end{bmatrix}, \quad T = \begin{bmatrix} tx \\ ty \end{bmatrix}$$

This allows us to write the two dimensional translation equations in the matrix form as

$$P' = P + T.$$

Rigid-body translation

A Translation is said to be a rigid body translation if it moves the objects without deformation.

That is, every point on the object is translated by the same amount. A straight line segment is translated by applying the transformation equation to each of the two endpoints and redrawing the line between the new endpoints.

A polygon is translated similarly, by adding a translation vector to the coordinate position of each vertex and then regenerate the polygon using the new set of vertex coordinates.
Program for translation

#include <glut.h>

int main (int argc, char** argv)
{
    void PatternSegment(void);
    void init (void);
    glutInit (&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (400, 300);
    glutInitWindowPosition (50, 100);
    glutCreateWindow ("Muhammad");
    init ();
    glutDisplayFunc (PatternSegment);
    glutMainLoop();
    return 0;
}

void PatternSegment (void)
{
    void Tri();
    glClear (GL_COLOR_BUFFER_BIT);
    glPushMatrix();
    glColor3f (1.0, 0.0, 0.0);
    Tri();
    glTranslatef(10.0,10.0,0.0);
Two Dimensional Scaling.

Scaling transformation is applied to alter the size of the object.

A simple two dimensional scaling operation is performed by multiplying object positions \((x, y)\) by scaling factors \(s_x\) and \(s_y\), to produce the transformed coordinates \((x', y')\).

\[
x' = x \cdot s_x \\
y' = y \cdot s_y
\]

Scaling factor \(s_x\) scales an object in the \(x\) direction, while \(s_y\) scales in the \(y\) direction.

The basic two dimensional scaling equation can be written in the matrix form as

\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}
\]

\[
P' = S \cdot P
\]
Rules for Scaling

1. Any positive value can be assigned to the scaling factors sx and sy.
2. Values less than 1 will reduce the size of the objects.
3. Values greater than 1 produce enlargements.
4. Specifying a value of 1 for both sx and sy leaves the size of object unchanged.

Uniform Scaling

When sx and sy are assigned the same value, a uniform scaling is produced which maintains relative object proportions.

Differential Scaling

Unequal values for sx and sy results in a differential scaling that is often used in design applications.

Polygon Scaling

Polygons are scaled by applying transformations to each vertex, then regenerating the polygon using the transformed vertices.

Program for scaling

#include <glut.h>

int main(int argc, char** argv)
{
    void PatternSegment(void);
    void init (void);
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (400, 300);
    glutInitWindowPosition (50, 100);
    glutCreateWindow ("Muhammad");
    init ();
}
void PatternSegment(void)
{
    void Tri();
    glClear (GL_COLOR_BUFFER_BIT);
   PushMatrix();
    glColor3f (1.0, 0.0, 0.0);
    Tri();
    glScalef(10.0,10.0,0.0);
    glColor3f (0.0, 1.0, 0.0);
    Tri();
   PopMatrix();
    glutSwapBuffers();
    glFlush ();
}

void init (void)
{
    glClearColor (1.0, 1.0, 1.0, 0.0);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0, 200.0, 0.0, 150.0);
}

void Tri()
{
    glBegin(GL_TRIANGLES);
        glVertex2i(10, 10);
        glVertex2i(60, 10);
        glVertex2i(30, 60);
    glEnd();
}
Two Dimensional Rotation.

To generate a rotation transformation of an object by specifying a rotation axis and a rotation angle.

All points of the object are then transformed to new positions by rotating the points through the specified angle about the rotation axis.

A two dimensional rotation of an object is obtained by repositioning the object along a circular path in the xy plane.

If \( r \) is the constant distance of the point from the origin, angle \( \Phi \) is the original angular position of the point form the horizontal and \( \varphi \) is the rotation angle as shown in the figure then

![Diagram of rotation](image)

The original coordinates can be expressed as

\[
\begin{align*}
x &= r \cos \Phi \\
y &= r \sin \Phi
\end{align*}
\]

The transformed coordinates can be expressed as

\[
\begin{align*}
x' &= r \cos (\Phi + \varphi) \\
y' &= r \sin (\Phi + \varphi)
\end{align*}
\]

\[
\begin{align*}
x' &= r \cos \Phi \cdot \cos \varphi - r \sin \Phi \cdot \sin \varphi \\
y' &= r \sin \Phi \cdot \sin \varphi + r \sin \Phi \cdot \cos \varphi
\end{align*}
\]

Substituting the value of A in B we get

\[
\begin{align*}
x' &= x \cos \varphi - y \sin \varphi \\
y' &= x \sin \varphi + y \cos \varphi
\end{align*}
\]
Program for Rotation

#include <glut.h>

int main(int argc, char** argv)
{
    void PatternSegment(void);
    void init (void);
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (400, 300);
    glutInitWindowPosition (50, 100);
    glutCreateWindow ("Muhammad");
    init ();
    glutDisplayFunc(PatternSegment);
    glutMainLoop();
    return 0;
}

void PatternSegment(void)
{
    void Tri();
    glClear (GL_COLOR_BUFFER_BIT);
    glPushMatrix();
    glColor3f (1.0, 0.0, 0.0);
Tri();	glRotatef(15.0,0,0,1);	glColor3f (0.0, 1.0, 0.0);
Tri();
glPopMatrix();
glutSwapBuffers();
glFlush ()
}

void init (void)
{
   glClearColor (1.0, 1.0, 1.0, 0.0);
   glMatrixMode(GL_PROJECTION);
   gluOrtho2D(0.0, 200.0, 0.0, 150.0);
}

void Tri()
{
   glBegin(GL_TRIANGLES);
      glVertex2i(10, 10);
      glVertex2i(60, 10);
      glVertex2i(30, 60);
   glEnd();
}
Matrix Transformation

Translation \( T = \begin{bmatrix} 1 & 0 & Tx \\ 0 & 1 & Ty \\ 0 & 0 & 1 \end{bmatrix} \)

Scaling \( S = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \)

Rotation \( R = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \)

Rules for Transformation

Translation

Translation by \((tx,ty)\) then we have

\[
\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & Tx \\ 0 & 1 & Ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}
\]

Scaling

Scaling by \((Sx,Sy)\) then we have

\[
\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}
\]

Rotation

Rotation by \(\theta\) then we have

\[
\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}
\]
Simple Transformation

1. Translate point (1,3) by (5,-1)
   \( x = 1, \ y = 3, \ Tx = 5, \ Ty = -1 \)

\[
\begin{bmatrix}
    x' \\
    y' \\
    1
\end{bmatrix} = \begin{bmatrix} 1 & 0 & Tx \\
                             0 & 1 & Ty \\
                             0 & 0 & 1 \end{bmatrix} \times 
\begin{bmatrix}
    x \\
    y \\
    1
\end{bmatrix}
\]

\[
\begin{bmatrix}
    x' \\
    y' \\
    1
\end{bmatrix} = \begin{bmatrix} 1 & 0 & 5 \\
                             0 & 1 & -1 \\
                             0 & 0 & 1 \end{bmatrix} \times 
\begin{bmatrix}
    1 \\
    3 \\
    1
\end{bmatrix}
\]

\[
= \begin{bmatrix}
    1 \times 1 + 0 \times 3 + 5 \times 1 \\
    0 \times 1 + 1 \times 3 + (-1) \times 1 \\
    0 \times 1 + 0 \times 3 + 1 \times 1
\end{bmatrix}
\]

\[
= \begin{bmatrix} 6 \\
                   2 \\
                   1 \end{bmatrix}
\]

2. a. Scale coordinates of (1,1) by (2,3).
   \( x = 1, \ y = 1, \ Sx = 2, \ Sy = 3 \)

\[
\begin{bmatrix}
    x' \\
    y' \\
    1
\end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\
                             0 & 3 & 0 \\
                             0 & 0 & 1 \end{bmatrix} \times 
\begin{bmatrix}
    1 \\
    1 \\
    1
\end{bmatrix}
\]

\[
= \begin{bmatrix} 2 \\
                   3 \\
                   1 \end{bmatrix}
\]

b. Scale a line between (2,1) and (4,1) to twice its length.

For first point
\( Sx = 2, \ x = 2, \ y = 1 \)

\[
\begin{bmatrix}
    x' \\
    y' \\
    1
\end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\
                             0 & 1 & 0 \\
                             0 & 0 & 1 \end{bmatrix} \times 
\begin{bmatrix}
    2 \\
    1 \\
    1
\end{bmatrix}
\]

\[
= \begin{bmatrix} 4 \\
                   1 \\
                   1 \end{bmatrix}
\]

For second point
\( Sx = 2, \ x = 4, \ y = 1 \)
c. Scale a line between (0,1) and (2,1) to twice its length, the left hand endpoint does not move.

For first point
\[ S_x = 2 \] , \( x = 0 \) , \( y = 1 \)
\[
\begin{bmatrix}
  x' \\
y' \\
1
\end{bmatrix} =
\begin{bmatrix}
  2 & 0 & 0 \\
  0 & 1 & 0 \\
  0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
  0 \\
  1 \\
1
\end{bmatrix} =
\begin{bmatrix}
  0 \\
  1 \\
1
\end{bmatrix}
\]

For second point
\[ S_x = 2 \] , \( x = 2 \) , \( y = 1 \)
\[
\begin{bmatrix}
  x' \\
y' \\
1
\end{bmatrix} =
\begin{bmatrix}
  2 & 0 & 0 \\
  0 & 1 & 0 \\
  0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
  2 \\
  1 \\
1
\end{bmatrix} =
\begin{bmatrix}
  4 \\
  1 \\
1
\end{bmatrix}
\]
3. Rotation of coordinates (1,0) by 45 about the origin
\[ x = 1, \ y = 0, \ \varphi = 45 \]

\[
\begin{bmatrix}
x' \\
y' \\
1
\end{bmatrix} = \begin{bmatrix}
\cos 45 & -\sin 45 & 0 \\
\sin 45 & \cos 45 & 0 \\
0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
1 \\
0 \\
1
\end{bmatrix}
\]

From the table we see the value of angles and substitute in the matrix as shown below

<table>
<thead>
<tr>
<th>( \theta )</th>
<th>0°</th>
<th>30°</th>
<th>45°</th>
<th>60°</th>
<th>90°</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sin( \theta )</td>
<td>0</td>
<td>1/2</td>
<td>1</td>
<td>( \sqrt{3} )</td>
<td>1</td>
</tr>
<tr>
<td>Cos( \theta )</td>
<td>1</td>
<td>( \sqrt{3} )</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Tan( \theta )</td>
<td>0</td>
<td>1</td>
<td>( \sqrt{3} )</td>
<td>( \sqrt{3} )</td>
<td>*</td>
</tr>
</tbody>
</table>

\[
\begin{bmatrix}
x' \\
y' \\
1
\end{bmatrix} = \begin{bmatrix}
1/\sqrt{2} & -1/\sqrt{2} & 0 \\
1/\sqrt{2} & 1/\sqrt{2} & 0 \\
0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
1 \\
0 \\
1
\end{bmatrix} = \begin{bmatrix}
1/\sqrt{2} \\
1/\sqrt{2} \\
1
\end{bmatrix}
\]

**Composite Transformation**

1. Find the transformation matrix for scale by 2 with fixed point (2,1) and do the following composite transformation for line between (2,1) and (4,1).

First find the transformation matrix for scale by 2 with fixed point (2,1).

1. Translate origin to the point (2,1).
2. Scale by 2.
3. Translate the point (2,1) to the origin.
Translate origin to the point (2,1) the matrix is

\[
\begin{bmatrix}
1 & 0 & 2 \\
0 & 1 & 1 \\
0 & 0 & 1 \\
\end{bmatrix}
\]

Scale by 2 the matrix is

\[
\begin{bmatrix}
2 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\end{bmatrix}
\]

Translate the point (2,1) to the origin the matrix is

\[
\begin{bmatrix}
1 & 0 & -2 \\
0 & 1 & -1 \\
0 & 0 & 1 \\
\end{bmatrix}
\]

Multiply all the three we get the transformation matrix as

\[
\begin{bmatrix}
1 & 0 & 2 \\
0 & 1 & 1 \\
0 & 0 & 1 \\
\end{bmatrix} \times \begin{bmatrix}
2 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\end{bmatrix} \times \begin{bmatrix}
1 & 0 & -2 \\
0 & 1 & -1 \\
0 & 0 & 1 \\
\end{bmatrix} = \begin{bmatrix}
2 & 0 & -2 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\end{bmatrix}
\]

Now we use the transformation matrix to perform transformation of line between (2,1) and (4,1) as shown below

\[
\begin{bmatrix}
2 & 0 & -2 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\end{bmatrix} \times \begin{bmatrix}
2 \\
1 \\
1 \\
\end{bmatrix} = \begin{bmatrix}
2 \\
1 \\
1 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
2 & 0 & -2 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\end{bmatrix} \times \begin{bmatrix}
4 \\
1 \\
1 \\
\end{bmatrix} = \begin{bmatrix}
6 \\
1 \\
1 \\
\end{bmatrix}
\]
UNIT 6: Clipping

Clipping is the process of extracting identifying element of a scene or picture inside or outside a special region, called the clipping region.

Clipping is useful for copying, moving or deleting a portion of a scene or picture.

Example:

The classical ‘cut and paste’ operation in a windowing system.

Clipping Window

A section of a two dimensional scene that is selected for display is called a clipping window, because all parts of the scene outside the selected section are clipped off.

Viewport

Objects inside the clipping window are mapped to the viewport and it is the viewport that is then positioned within the display window. The clipping window selects what we want to see. The viewport indicates where it is to be viewed on the output device.
Window to Viewport transformation

The mapping of a two-dimensional, world-coordinate scene description to device coordinates is called a two-dimensional viewing transformation.

Sometime this transformation is simply referred to as the window-to-viewport transformation or the windowing transformation. But in general viewing involves more than just the transformation from clipping-window coordinates to viewport coordinates.

Normalization and Viewport Transformations

Many applications combine normalization and window-to-viewport transformations. The coordinates of the viewport are given in the range [0,1] so that the viewport is positioned within a unit square.

After clipping, the unit square viewport is mapped to the output device. In other systems, the normalization and clipping is performed before viewport transformation. The viewport boundaries are given in screen coordinates relative to the display-window position.

Mapping the Clipping Window into a Normalized Viewport

Object descriptions are transferred to this normalized space using a transformation that maintains the same relative placement of a point in the viewport as it had in the clipping window.
If a coordinate position is at the centre of the clipping window, for instance, it would be mapped to the centre of the viewport.

Position(Xw,Yw) in the clipping window is mapped into position (Xv,Yv) in the associated viewport.

We can obtain the transformation from the world coordinates to viewport coordinates with the sequence as shown below

**Step 1:** Scale the clipping window to the size of the viewport using a fixed point position of \((x_{wmin}, y_{wmin})\).

**Step 2:** Translate \((x_{wmi}, y_{wmin})\) to \((x_{vmi}, y_{vmin})\).

A point \((xw,yw)\) in a world coordinate clipping window is mapped to viewport coordinates \((xv,yv)\), within a unit square, so that the relative positions of the two points in their respective rectangles are the same.
To transform the world coordinate point into the same relative position within the viewport we require that

\[
\frac{x_v-x_v_{min}}{x_v_{max}-x_v_{min}} = \frac{x_w-x_w_{min}}{x_w_{max}-x_w_{min}}; \quad \frac{y_v-y_v_{min}}{y_v_{max}-y_v_{min}} = \frac{y_w-y_w_{min}}{y_w_{max}-y_w_{min}}
\]

\[
\frac{x-x_{min}}{x_{max}-x_{min}}; \quad \frac{y-y_{min}}{y_{max}-y_{min}}
\]

Clipping Algorithms

Generally, any procedure that eliminated those portions of a picture that are either inside or outside of a specified region of space is referred to as a clipping algorithm or simply clipping.

Usually a clipping region is a rectangle in standard position, although we could use any shape for a clipping application.

The following are few two dimensional algorithms

1. Point Clipping
2. Line Clipping (Straight line segment)
3. Fill Area Clipping (Polygons)
4. Curve Clipping
5. Text Clipping

Two Dimensional Point Clipping

For a Clipping rectangle in standard position, we save a two-dimensional point \( P = (x,y) \) for display if the following inequalities are satisfied

\[
x_{min} \leq x \leq x_{max}
\]

\[
y_{min} \leq y \leq y_{max}
\]
If any one of these four inequalities is not satisfied, the point is clipped (not saved for display)

Although point clipping is applied less often than line or polygon clipping, it is useful in various situations, particularly when pictures are modeled with particle systems.

**Two dimensional Line Clipping**

A line clipping algorithm processes each line in a scene through a series of tests and intersection calculations to determine whether the entire line or any part of it is to be saved.

1. When both endpoints of a line segment are inside all four clipping boundaries, such as line from P3 to P4 in the above figure, the line completely inside the clipping window and we save it.

2. When both endpoints of line segment are outside any one of the four boundaries such as line P1 and P2, that line is completely outside the window and it is eliminated from the scene description.

3. But if both these test fail, the line segment intersects at least one clipping boundary and it may or may not cross into the interior of the clipping window.

(Both endpoints inside: trivial accept; One inside: find intersection and clip; Both outside: either clip or reject)
Cohen-Sutherland line Clipping

The Cohen-Sutherland two-dimensional line clipping algorithm basically divides the clipping region into; number of sections (specifically nine regions), each line endpoint is assigned with its own unique 4-bit binary number, called an out code or region code.

And each bit position is used to indicate whether the point is inside or outside one of the clipping-window boundaries.

<table>
<thead>
<tr>
<th>1001</th>
<th>1000</th>
<th>1010</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>Clipping Window 0000</td>
<td>0010</td>
</tr>
<tr>
<td>0101</td>
<td>0100</td>
<td>0110</td>
</tr>
</tbody>
</table>

The nine binary region codes for identifying the position of a line endpoint, relative to the clipping window boundaries

<table>
<thead>
<tr>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top</td>
<td>Bottom</td>
<td>Right</td>
<td>Left</td>
</tr>
</tbody>
</table>

A possible ordering for the clipping window boundaries corresponding to the bit positions in the Cohen-Sutherland endpoint region code

One possible ordering with the bit positions numbered 1 through 4 from right to left.

Thus, for this ordering, the rightmost position (bit 1) references the left clipping-window boundary, and the leftmost position (bit 4) references the top window boundary.

A value of 1 (or true) in any bit position indicates that the endpoint is outside of that window border. Similarly, a value of 0 (or false) in any bit position indicates that the end point is not outside (it is inside or on) the corresponding window edge.
**Polygon clipping**

Polygon is a collection of lines. Therefore we might think that the line clipping can be used directly for polygon clipping.

However, when a closed polygon is clipped as a collection of lines with the line clipping algorithm, the original closed polygon becomes one or more open polygon or discrete lines. Thus we need to modify the line clipping algorithm to clip polygon.

![Polygon Clipping Diagram](image)

We consider a polygon as a closed solid area. Hence after clipping it should remain closed. To achieve this we require an algorithm that will generate additional line segment which make the polygon as a closed area.

The lines a-b, c-d, d-e, f-g, g-h, i-j are added to polygon description to make it closed. **Sutherland Hodgeman polygon clipping**
A polygon can be clipped by processing its boundary as a whole against each window edge. This is achieved by processing all polygon vertices against each clip rectangle boundary in turn.

Beginning with the original set of polygon vertices, we could first clip the polygon against the left rectangle boundary to produce a new sequence of vertices.

The new set of vertices could then be successively passed to a right boundary clipper, a top boundary clipper & a bottom boundary clipper as shown in fig.

At each step a new set of polygon vertices is generated and passed to the next window boundary clipper. This is the fundamental idea in the Sutherland Hodgeman algorithm.

The output algorithm is a list of polygon vertices all of which are on the visible side of the clipping plane. This is achieved by processing 2 vertices of each edge of the polygon around the clipping boundary or plane.

**Curves**
A Curve is a continuous map from a one-dimensional space to an n-dimensional space.

Properties of Curves

Local properties:

- continuity.
- position at a specific place on the curve.
- direction at a specific place on the curve.
- curvature.

Global properties:

- whether the curve is open or closed.
- whether the curve ever passes through a particular point, or goes through a particular region.
- whether the curve ever points in a particular direction.

Types of Curves

Quadratic Curves: They are curves of 2nd order. Equation for quadratic curve is

\[ x(t) = at^2 + b \cdot t + c \]

Cubic Curves: They are curves of 3rd order. Equation for cubic curve is
\[ x(t) = at^3 + bt^2 + ct + d \]

Coefficients a, b, c, d are known as control points.

**Relationship between Control points and Order of the curves.**

- If there are only two points they define a line (1\(^{\text{st}}\) order).
- If there are three points they define a quadratic curve (2\(^{\text{nd}}\) order).
- Four points define a cubic curve (3\(^{\text{rd}}\) order);

In general \( k+1 \) points can be used to define a curve of \( k \)-order curve.

**Bezier Curve**

A Bezier curve is a parametric curve which are used to model smooth curves that can be scaled indefinitely.

**Types of Bezier Curves**

- **Linear**
- **Quadratic**
- **Cubic**

**Cubic Bezier Curve:**

- It is defined by four control points.
- Two interpolated endpoints (Points are on the curve).
- Two points control the tangents at the end.
- Points $x$ on curve defined as function of a parameter $t$.

**Degree of Bezier Curves:**
A degree $n$ Bezier curve has $n+1$ control points. For example, a degree 2 Bezier curve will have 2+1 i.e. 3 control points.
Projection is mapping 3D coordinates to 2D coordinates. It is to transform points from camera coordinate system to the screen.

**Parallel projection**

Center of projection is at infinity. Direction of projection (DOP) same for all points

**Properties of Parallel Projection**

- Not realistic looking.
- Good for exact measurement.
- Are actually affine transformation
  - parallel lines remain parallel
  - ratios are preserved
  - angles are often not preserved
- Most often used in CAD, architectural drawings, etc. where taking exact measurement is important
**Perspective projection**
- Maps points onto “view plane” along projectors emanating from “center of projection” (COP).

**Properties of Perspective Projection**
- Perspective projection is an example of projective transformation
  - lines maps to lines
  - parallel lines do not necessarily remain parallel
  - ratios are not preserved
- One of advantages of perspective projection is that size varies inversely proportional to the distance—looks realistic