18. Khan Academy: Computer Networking (https://www.khanacademy.org/computing/computer-science/internet-intro)
19. GeeksforGeeks: Computer Network (https://www.geeksforgeeks.org/computer-network-tutorials/)
20. Stanford OpenEdx: Computer Networking (https://openedx.stanford.edu/courses/course-v1:ComputerScience+Networking+SelfPaced/about)

THEORY OF COMPUTER AND COMPUTER GRAPHICS (ACtE06)

6.1 INTRODUCTION TO FINITE AUTOMATA

Introduction to Finite Automata and Finite State Machine:

The Finite Automata (FA) is a basic machine used for pattern recognition. It is an abstract machine that consists of five elements or tuples, including a set of states and rules that govern how it transitions between states based on input symbols.

Finite Automata (FA) can be classified into two types: Deterministic Finite Automata (DFA) and Non-Deterministic Finite Automata (NFA).

Deterministic Finite Automata (DFA)

A Deterministic Finite Automata (DFA) is a type of finite automata that operates on a finite set of input symbols and a finite set of states. It has the following characteristics

- Each state has exactly one transition for each possible input symbol.
- The transition function takes the current state and an input symbol as arguments and returns the next state.
- There is a unique start state and a set of accepting states.
- A DFA can read an input string one symbol at a time and transition from one state to another until it reaches the end of the string.
- If the final state is an accepting state, the DFA accepts the input string, otherwise it rejects the input string.

DFA consists of 5 tuples $\{Q, \Sigma, q, F, \delta\}$.

Q : set of all states.

Σ : set of input symbols. (Symbols which machine takes as input)

q : Initial state. (Starting state of a machine)

F : set of final state.

δ : Transition Function, defined as $\delta : Q \times \Sigma \rightarrow Q$.

Example 1: *Consider the string "abababb". The goal is to determine if the string is accepted by a Deterministic Finite Automaton (DFA) by processing the string with the DFA. The steps of the process are shown as follows:*

δ(q0, abababb) = δ(q1, bababb)
= δ(q2, ababb)
= δ(q1, babb)
= δ(q2, abb)
= δ(q1, bb)
= δ(q2, b)
= δ(q3, ε)
= q3 [ACCEPT]

The string is considered accepted by the DFA if, after processing all the symbols, the final state q3 is reached.

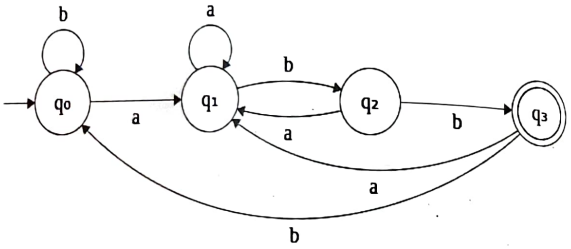


Figure: DFA

The same Deterministic Finite Automaton (DFA) can also be represented in the form of a table, as shown below. Both representations are equivalent and provide the same information.

Input	a	b
→ q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_3
q_3^*	-	-

Nondeterministic Finite Automata (NFA):

Nondeterministic Finite Automata (NFA) is a type of finite state machine where there are multiple transitions from a state for the same input symbol. The transition function δ maps the current state and input symbol to a set of next states, δ(qi, σ) = {qj1, ..., qjK}.

The following diagram shows an NFA that accepts all input strings ending with "abb". The alphabet, Σ, is {a, b}.

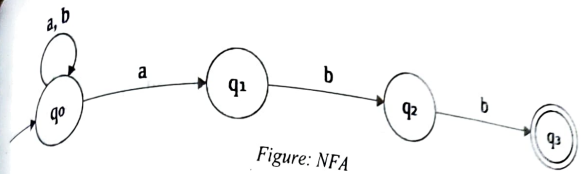


Figure: NFA

The same NFA can also be represented in the form of a table, as shown below. Both representations are equivalent and provide the same information.

Input	a	b
→q_0	{q_0, q_1}	{q_0}
q_1	{q_1}	{q_2}
q_2	{q_1}	{q_3}
q_3^*	φ	φ

It has the following characteristics:

- Each state can have multiple transitions for the same input symbol.
- The transition function takes the current state and an input symbol as arguments and returns a set of possible next states.
- There is a unique start state and a set of accepting states.
- An NFA can read an input string one symbol at a time and transition from one state to another until it reaches the end of the string.
- If there exists at least one path from the start state to an accepting state that matches the input string, the NFA accepts the input string.

An NFA is a type of finite state automaton that allows for null (ε) moves and the ability to transition to any number of states for a particular input. The null (ε) move allows the NFA to move from one state to another without consuming any input symbol.

Applications of DFA and NFA:

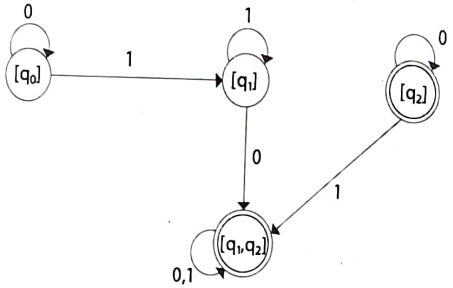
Deterministic Finite Automata (DFA)

a) Pattern matching: DFA can be used to match patterns in strings such as finding a specific word in a text document.
b) Lexical analysis: DFAs are used in lexical analysis to identify tokens in a programming language.
c) Validation of inputs: DFAs can be used to validate inputs in form fields, such as checking for valid email addresses or phone numbers.
d) Compiler design: DFAs are used in the design of compilers to identify keywords, operators, and identifiers in a source code.

The state [q1, q2] is the final state as well because it contains a final state q2. The transition table for the constructed DFA will be:

State	0	1
→[q0]	[q0]	[q1]
[q1]	[q1, q2]	[q1]
*[q2]	[q2]	[q1, q2]
*[q1, q2]	[q1, q2]	[q1, q2]

The transition diagram of the resulting DFA is shown as follows:



Minimization of Finite State Machines:

The process of reducing the number of states in a DFA is known as minimization. The goal of minimization is to obtain a smaller FSM that accepts the same language as the original FSM, but with fewer states. This process helps to simplify the design and improve the performance of the FSM.

Minimizing a DFA involves the following steps:

1. Start with Q' = ϕ
2. Add the initial state q0 of the NFA to Q', and find the transitions from this start state.
3. In Q', determine the possible set of states for each input symbol. If the set of states is not in Q', add it to Q'.
4. In the DFA, the final states will be all the states that contain the final states of the NFA (F).
5. Remove all states that are unreachable from the initial state through any set of transitions in the DFA.
6. Create a transition table for all pairs of states.
7. Divide the transition table into two tables, T1 and T2, with T1 containing all final states and T2 containing non-final states.
8. Find similar rows in T1, such that $\delta(q, a) = p$ and $\delta(r, a) = p$, and remove one of the states.
9. Repeat step 8 until no similar rows are found in the transition table T1.

10. Repeat steps 8 and 9 for table T2.
11. Combine the reduced T1 and T2 tables to obtain the transition table of the minimized DFA.

Regular Expressions:

- A regular expression is a simple expression used to describe the language accepted by a finite automaton. It is an effective way to represent any language and the languages accepted by these expressions are called Regular Languages. A regular expression is a sequence of patterns that define a string. They are used to match character combinations in strings and string searching algorithms use these patterns for operations on a string.

The language described by a regular expression is known as a Regular Language. These expressions provide an easy way to describe and recognize patterns in strings. The following are the key operations used in regular expressions:

- ϕ represents the regular expression for the language ϕ.
- ε represents the regular expression for the language $\{\varepsilon\}$.
- If $a \in \Sigma$ is a symbol from the input alphabet Σ, then a is a regular expression with the language {a}.
- If a and b are regular expressions, then a + b is also a regular expression with the language {a,b}.
- If a and b are regular expressions, then ab (concatenation of a and b) is also a regular expression.
- If a is a regular expression, then a* (0 or more times a) is also a regular expression.

Closure Properties of Regular Languages

a) **Union**: The union of two regular languages, L1 and L2, is also a regular language. For example, $L1 = \{a^n \mid n \geq 0\}$ and $L2 = \{b^n \mid n \geq 0\}$. The union of L1 and L2, $L3 = L1 \cup L2 = \{a^n \cup b^n \mid n \geq 0\}$, is also a regular language.

b) **Intersection**: The intersection of two regular languages, L1 and L2, is also a regular language. For instance, $L1 = \{a^m b^n \mid n \geq 0 \text{ and } m \geq 0\}$ and $L2 = \{a^m b^n \cup b^n a^m \mid n \geq 0 \text{ and } m \geq 0\}$. The intersection of L1 and L2, $L3 = L1 \cap L2 = \{am\ bn \mid n \geq 0 \text{ and } m \geq 0\}$, is also a regular language.

c) **Concatenation**: The concatenation of two regular languages, L1 and L2, is also a regular language. For example, $L1 = \{a^n \mid n \geq 0\}$ and $L2 = \{b^n \mid n \geq 0\}$. The concatenation of L1 and L2, $L3 = L1.L2 = \{a^m . b^n \mid m \geq 0 \text{ and } n \geq 0\}$, is also a regular language.

d) **Kleene Closure**: The Kleene closure of a regular language, L1, is also a regular language. For example, L1 = (a ∪ b). The Kleene closure of L1, L1*, is $(a \cup b)^*$.

e) **Complement:** The complement of a regular language, L(G), is also a regular language. The complement can be found by subtracting strings which are in L(G) from all possible strings. For example, $L(G) = \{a^n \mid n > 3\}$. The complement of L(G), L'(G) = $\{a^n \mid n <= 3\}$, is also a regular language.

6.2 INTRODUCTION TO CONTEXT FREE LANGUAGE

Introduction to Context Free Grammar (CFG), Derivative trees (Bottom-up and Top-down approach, Leftmost and Rightmost, Language of a grammar), Parse tree and its construction, Ambiguous grammar, Chomsky Normal Form (CNF), Greibach Normal Form (GNF), Backus-Naur Form (BNF), Push down automata, Equivalence of context free language and PDA, Pumping lemma for context free language, and Properties of context free Language. *(ACtE0603)*

6.3 TURING MACHINE

Introduction to Turing Machines (TM), Notations of Turing Machine, Acceptance of a string by a Turing Machines, Turing Machine as a Language Recognizer, Turing Machine as a Computing Function, Turing Machine as a enumerator of stings of a language, Turing Machine with Multiple Tracks, Turing Machine with Multiple Tapes, Non-Deterministic Turing Machines, Curch Turing Thesis, Universal Turing Machine for encoding of Turing Machine, Computational Complexity, Time and Space complexity of A Turing Machine, Intractability, Reducibility. *(ACtE0603)*

6.4 INTRODUCTION OF COMPUTER GRAPHICS

Overview of Computer Graphics, Graphics Hardware (Display Technology, Architecture of Raster-Scan Displays, Vector Displays, Display Processors, output device and Input Devices), Graphics Software and Software standards. *(ACtE0604)*

Introduction of computer graphics:

Computer graphics is a field of computer science that deals with the creation, manipulation, and representation of images using computers. The goal of computer graphics is to generate images that are visually appealing and communicate information effectively to the user.

Overview of Computer Graphics:

Computer Graphics is a branch of computer science that deals with the creation, manipulation, and representation of images using computers. Here is an overview of the field in the form of bullet points:

a) Computer graphics involves the use of algorithms, mathematical models, and data structures to generate, store, process, and display images.
b) Computer Graphics is a branch of computer science that involves the creation, manipulation, and representation of images using computers.
c) It encompasses a wide range of applications, including video games, films, scientific visualization, product design, and data visualization.

Applications:

Computer graphics has a wide range of applications, including:

a) Video Games: Computer graphics are used to create immersive and interactive environments in video games.

b) Films: Computer graphics are widely used in the film industry to create special effects, animations, and virtual environments.
c) Scientific Visualization: Computer graphics are used to visualize complex scientific data and simulations, such as weather patterns, medical imaging, and astronomical observations.
d) Product Design: Computer graphics are used in product design to create 3D prototypes, visualize products in virtual environments, and simulate product behavior.
e) Data Visualization: Computer graphics are used to represent data and information in a visual format, such as charts, graphs, and maps.
f) Virtual Reality and Augmented Reality: Computer graphics are used to create virtual environments for gaming and training simulations, as well as to overlay digital information onto the real world in augmented reality applications.
g) Advertising and Marketing: Computer graphics are used to create visual advertisements and marketing materials, such as logos, brochures, and product packaging.

Education and Training: Computer graphics are used to create educational materials, such as simulations, animations, and interactive tutorials.

Architecture and Urban Planning: Computer graphics are used to create virtual models of buildings, cities, and landscapes for architectural design, urban planning, and simulation.

Graphics Hardware (Display Technology, Architecture of Raster-Scan Displays, Vector Displays, Display Processors, output device and Input Devices):

Computer graphics hardware refers to the physical components of a computer system that are used to display, process, and input images. These components include displays, graphics cards, display processors, output devices, and input devices.

Displays are the primary output device used to display images in a computer graphics system. There are several types of displays, including CRT (Cathode Ray Tube), LCD (Liquid Crystal Display), OLED (Organic Light-Emitting Diode), and LED (Light-Emitting Diode).

Display Technology

Computer graphics display technology refers to the various types of displays that are used to display images generated by computer graphics systems. Here is an overview of some of the key display technologies:

a) Cathode-Ray Tube (CRT) displays: CRT displays were the first type of display used in computer graphics and work by using an electron beam to illuminate a phosphor screen, creating an image. CRT displays have largely been replaced by other types of displays, but are still used in some specialized applications.
b) Color CRT Monitor: Color CRT monitors are CRT displays that are capable of displaying images in color. Color CRT monitors use a combination of red, green, and blue phosphors to produce a full-color image.
c) Liquid Crystal Display (LCD) displays: LCD displays use liquid crystal technology to control the light passing through the display and create an image. LCD displays are used

in a wide range of applications, including computer monitors, televisions, and smartphones.

d) Light Emitting Diode (LED) displays: LED displays use arrays of light-emitting diodes to create an image. LED displays are used in a wide range of applications, including televisions, computer monitors, and digital signage.

e) Direct View Storage Tubes (DVST) displays: DVST displays use a beam of electrons to write and store an image on a phosphor screen. DVST displays are used in specialized applications, such as military and industrial displays.

f) Plasma Display: Plasma displays use arrays of cells containing a mixture of gases to produce light and create an image. Plasma displays are used in a wide range of applications, including televisions and computer monitors.

g) 3D Display: 3D displays are displays that are capable of displaying images in 3D providing a more immersive and interactive experience for the user. 3D displays use various techniques, such as polarized glasses, to create the illusion of depth.

Architecture of Raster-Scan Display:

Raster-scan displays are the most common type of graphics monitor, based on television technology. Raster-scan displays work by scanning an electron beam across the screen, from top to bottom, covering one row at a time. The resolution of a raster-scan display is determined by the number of pixels in the grid and the size of the display. Raster-scan displays are widely used in a range of applications and are capable of displaying images with high accuracy.

Vector Displays:

Vector displays are a type of display technology used in computer graphics. Unlike raster-scan displays, which work by scanning an electron beam across the screen to light up individual pixels, vector displays work by drawing lines between two points to form shapes. Vector displays are used in specialized applications, such as scientific and engineering graphics.

Table: Difference between Raster and Vector Technology

Random Scan	Raster Scan
1. It has high Resolution	1. Its resolution is low.
2. It is more expensive	2. It is less expensive
3. Any modification if needed is easy	3. Modification is tough
4. Solid pattern is tough to fill	4. Solid pattern is easy to fill
5. Refresh rate depends or resolution	5. Refresh rate does not depend on the picture.
6. Only screen with view on an area is displayed.	6. Whole screen is scanned.
7. Beam Penetration technology come under it.	7. Shadow mark technology came under this.
8. It does not use interlacing method.	8. It uses interlacing
9. It is restricted to line drawing applications	9. It is suitable for realistic display.

Display Processors:

The Display Processor is a hardware component or interpreter that transforms the code into a visual image. It takes the digital information from the CPU and converts it into analog signals for display on a screen.

The primary goal of the Display Processor is to relieve the CPU of graphic-related tasks. The processor digitizes the picture definitions provided by an application program and converts them into a series of pixel intensity values that are stored in the frame buffer. This process of conversion is referred to as Scan Conversion.

Output device and Input Devices:

Here is a list of input devices commonly used in computer graphics with an example for each:

a) Graphics tablet: A graphics tablet, also known as a digitizing tablet or a pen tablet, is an input device that allows the user to draw and sketch directly on the tablet surface, which is then translated into digital form on the computer screen. An example of a graphics tablet is the Wacom Intuos.

b) Joystick: A joystick is a handheld device that provides control inputs to a computer, typically used for gaming or simulation applications. An example of a joystick is the Microsoft Xbox One Controller.

c) Trackball: A trackball is a pointing device that consists of a ball housed in a socket that can be rotated with the thumb, fingers, or palm of the hand to control the movement of the cursor on the screen. An example of a trackball is the Kensington Orbit Trackball.

d) 3D Mouse: A 3D mouse is a specialized mouse that provides six degrees of freedom for 3D navigation and is often used in computer-aided design (CAD) and animation applications. An example of a 3D mouse is the 3Dconnexion Space Mouse Pro.

e) Light pen: A light pen is a pointing device that consists of a pen-shaped wand that emits light and is used to interact with the computer screen by pointing to the desired location on the screen. An example of a light pen is the Elo Touch Systems 1517L Touchscreen Monitor.

f) Stylus: A stylus is a pen-like device that is used to write, draw, or navigate on a touchscreen-based device, such as a tablet or smartphone. An example of a stylus is the Apple Pencil.

g) VR headset: A virtual reality (VR) headset is a device that provides an immersive 3D visual and audio experience, often used for gaming and simulation applications. An example of a VR headset is the Oculus Quest 2.

Here is a list of different types of output devices used in computer graphics, along with an example for each:

a) Monitor: A monitor is a display device that provides a visual output of the computer's graphics and video. An example of a monitor is the Dell UltraSharp U2720Q.

b) Printer: A printer is an output device that produces a hard copy of a computer's graphics and text. An example of a printer is the HP OfficeJet Pro 9015.

c) Plotter: A plotter is an output device that can produce high-quality, large-format prints, typically used in engineering, architecture, and design applications. An example of a plotter is the HP DesignJet Z6.

d) Projector: A projector is an output device that projects the computer's video and graphics onto a screen or wall, typically used in presentations and entertainment applications. An example of a projector is the Epson Home Cinema 2150.

e) Virtual Reality Headset: A virtual reality (VR) headset is a device that provides an immersive 3D visual and audio experience, often used for gaming and simulation applications. An example of a VR headset is the Oculus Quest 2.

f) Head-Mounted Display: A head-mounted display (HMD) is a device that is worn on the head and displays video and graphics directly in front of the user's eyes. An example of an HMD is the HTC VIVE Pro.

g) Sound System: A sound system is an output device that provides an audio output for the computer, often used for gaming and entertainment applications. An example of a sound system is the Logitech Z623 2.1 Speaker System.

Graphics Software and Software standards:

Graphics software is a type of computer program used to create, edit, and manipulate visual images. Graphics software provides tools and features to allow users to manipulate attributes such as color, texture, and shape.

GKS:

- Graphical Kernel System is a graphics software standard developed by ISO and various national standard organizations.
- GKS provides a common API for creating and manipulating 2D and 3D graphics.
- GKS includes functions for creating graphics primitives, controlling attributes, and managing the graphics pipeline.
- GKS was the first graphics software standard.

PHIGS:

- Programmer's Hierarchical Interactive Graphics System is an extension of GKS.
- PHIGS provides additional capabilities for object modeling, color specifications, surface rendering, and picture manipulation.
- PHIGS includes a hierarchical modeling system and support for 3D graphics.
- PHIGS+ is an extension of PHIGS that provides additional capabilities for 3D surface shading.

6.5 TWO-DIMENSIONAL TRANSFORMATION

Two-dimensional transformation:

Two-dimensional transformations refer to mathematical operations that are performed on two-dimensional objects, such as points, lines, and polygon shapes, to change their position, orientation, or size. The most commonly used two-dimensional transformations are:

a) **Translation**: Translation is the process of moving an object by a specified distance in a given direction. In two dimensions, translation is represented by a vector that specifies the distance and direction in which the object is to be moved.

We translate a 2D point by adding translation distances, tx and ty, to the original coordinate position (x, y):

$$x' = x + t_x,\ y' = y + t_y$$

Let's consider an example of translation in computer graphics. Suppose we have a point P at coordinates (2,3) in the Cartesian coordinate system, and we want to translate it by a vector (4, -1).

We can represent the point P as a column vector:

$$P = [2, 3, 1]$$

To perform the translation, we need to add the translation vector [4, -1, 0] to the vector P:

$$P' = P + [4, -1, 0] = [2+4, 3-1, 1+0]$$
$$P' = [6, 2, 1]$$

The resulting transformed point P' has coordinates (6,2), which is the original point (2,3) translated by a vector (4, -1).

b) **Rotation**: Rotation is the process of turning an object around a fixed point, called the origin. The angle of rotation and the origin can be specified to determine the new position of the object after rotation.

Let's consider an example of rotation in computer graphics. Suppose we have a point P at coordinates (2,3) in the Cartesian coordinate system, and we want to rotate it 45 degrees counterclockwise about the origin.

We can represent the point P as a column vector:

$$P = [2, 3, 1]$$

To perform the rotation, we need to multiply this vector by the rotation matrix R, which can be defined as:

$$R = [[\cos(45), -\sin(45), 0], [\sin(45), \cos(45), 0], [0, 0, 1]]$$

Multiplying P by R, we get:

$$P' = RP = [[\cos(45), -\sin(45), 0], [\sin(45), \cos(45), 0], [0, 0, 1]] \times [2, 3, 1]$$
$$P' = [(\cos(45) \times 2) + (-\sin(45) \times 3), (\sin(45) \times 2) + (\cos(45) \times 3), 1]$$
$$P' = [-0.707, 4.243, 1]$$

The resulting transformed point P' has coordinates (-0.707, 4.243), which is the original point (2,3) rotated 45 degrees counterclockwise about the origin.

c) **Scaling**: Scaling is the process of changing the size of an object by a given factor. The scale factor can be different for the x and y axes, allowing for non-uniform scaling in two dimensions.

Let's consider an example of scaling in computer graphics. Suppose we have a point P at coordinates (2,3) in the Cartesian coordinate system, and we want to scale it by a factor of 2 with respect to the origin.

We can represent the point P as a column vector:

$$P = [2, 3, 1]$$

To perform the scaling, we need to multiply the vector P by the scaling matrix S, which can be defined as:

$$S = [[2, 0, 0], [0, 2, 0], [0, 0, 1]]$$

Multiplying P by S, we get:

$$P' = SP = [[2, 0, 0], [0, 2, 0], [0, 0, 1]] \times [2, 3, 1]$$

$$P' = [2 \times 2, 2 \times 3, 1]$$

$$P' = [4, 6, 1]$$

The resulting transformed point P' has coordinates (4,6), which is the original point (2,3) scaled by a factor of 2 with respect to the origin.

d) **Shearing**: Shearing is the process of slanting an object by a given angle along one or both axes. This transformation is useful for creating a skewed or oblique effect.

Let's consider an example of shearing in computer graphics. Suppose we have a rectangle with vertices at (1,1), (3,1), (3,3), and (1,3) in the Cartesian coordinate system, and we want to apply horizontal shear to it by a factor of 1.

We can represent each vertex of the rectangle as a column vector:

$$A = [1, 1, 1]$$

$$B = [3, 1, 1]$$

$$C = [3, 3, 1]$$

$$D = [1, 3, 1]$$

To apply horizontal shear, we need to multiply the x-coordinates of each vertex by a factor of 1 and add the y-coordinates to them. This can be achieved by multiplying the vertices by the shear matrix H, which can be defined as:

$$H = [[1, 1, 0], [0, 1, 0], [0, 0, 1]]$$

Multiplying each vertex by H, we get:

$$A' = HA = [[1, 1, 0], [0, 1, 0], [0, 0, 1]] \times [1, 1, 1]$$

$$A' = [2, 1, 1]$$

$$B' = HB = [[1, 1, 0], [0, 1, 0], [0, 0, 1]] \times [3, 1, 1]$$

$$B' = [4, 1, 1]$$

$$C' = HC = [[1, 1, 0], [0, 1, 0], [0, 0, 1]] \times [3, 3, 1]$$

$$C' = [6, 3, 1]$$

$$D' = HD = [[1, 1, 0], [0, 1, 0], [0, 0, 1]] \times [1, 3, 1]$$

$$D' = [2, 3, 1]$$

The resulting transformed vertices A', B', C', and D' represent the rectangle after applying horizontal shear to it by a factor of 1. The transformed rectangle has vertices at (2,1), (4,1), (6,3), and (2,3).

e) **Reflection**: Reflection is the process of flipping an object over a specified line or axis. This transformation is useful for creating symmetrical shapes or objects.

Let's consider an example of reflection in computer graphics. Suppose we have a point P at coordinates (2,3) in the Cartesian coordinate system, and we want to reflect it about the y-axis.

We can represent the point P as a column vector:

$$P = [2, 3, 1]$$

To perform the reflection, we need to multiply the vector P by the reflection matrix R. which can be defined as:

$$R = [[-1, 0, 0], [0, 1, 0], [0, 0, 1]]$$

Multiplying P by R, we get:

$$P' = RP = [[-1, 0, 0], [0, 1, 0], [0, 0, 1]] \times [2, 3, 1]$$

$$P' = [-2, 3, 1]$$

The resulting transformed point P' has coordinates (-2,3), which is the original point (2,3) reflected about the y-axis.

Shear transformation

Shear transformation in computer graphics is a transformation that slants an object along one or both axes by a given angle. The mathematical representation of shear transformation in two-dimensional space is given by the following equation:

Let (x, y) be the coordinates of a point before shear and (x', y') be the coordinates of the same point after shear. The shear transformation along the x-axis by a factor kx can be represented as:

$$x' = x + kx * y$$

$$y' = y$$

The shear transformation along the y-axis by a factor ky can be represented as:

$$x' = x$$

$$y' = y + ky * x$$

These equations define a linear transformation that maps the original coordinates of a point to its new coordinates after shear. The parameters kx and ky determine the angle and direction of the shear along each axis.

For example, if a point (3, 4) is sheared along the x-axis by a factor of 0.5, its new coordinates would be (3 + 0.5 * 4, 4) = (5, 4).

Shear transformation is a useful tool in computer graphics for creating skewed or oblique effects in two-dimensional objects. It can be combined with other transformations, such as translation, rotation, and scaling, to create complex transformations and animations.

2D composite transformation:

We can combine multiple transformations or a sequence of transformations into a single transformation called composition, and represent it as a composite matrix. This combining process is known as concatenation.

If we want to perform a rotation about an arbitrary point, we can achieve it by applying a sequence of three transformations in the following order:

a) Translation

b) Rotation

c) Reverse Translation

It's important to keep this sequence of transformations in this order, as changing the order can lead to different results. When representing these transformations as matrices in column form, we can perform the composite transformation by multiplying the matrices in order from right to left. This means that we multiply the previous matrix output with the new matrix input to obtain the final output.

Example of composite transformations:

This statement describes a sequence of four steps to perform scaling of an object:

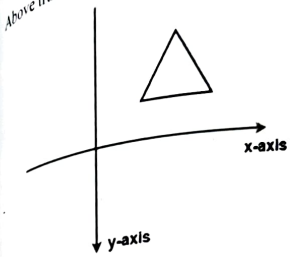
Step 1: The object is initially in some position (figure (a)).

Step 2: The object is then translated so that its center coincides with the origin (figure (b)).

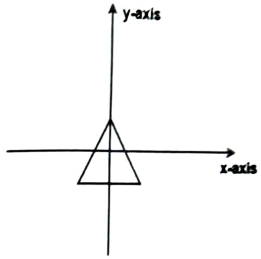
Step 3: The scaling of the object is performed while keeping it at the origin (figure (c)).

Step 4: Again, translation is done. This second translation is called a reverse translation. It will position the object at the origin location.

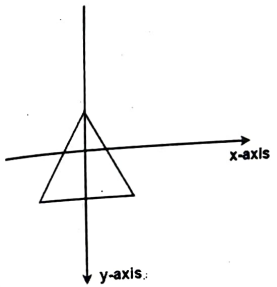
Above transformation can be represented as TV.STV-1



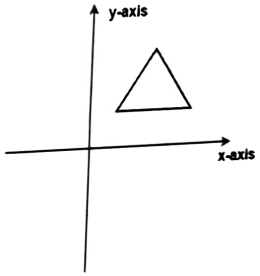
(original position of object)
Fig:(a)



(object is translated to origin)
Fig:(b)



(object is enlarged)
Fig:(c)



(object is retranslated to original position)
Fig:(d)

Row Method

1. Translation

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

2. Scaling

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ S_x & S_y & 1 \end{bmatrix}$$

Column Method

1.Translation

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

2. Scaling

$$\begin{bmatrix} 1 & 0 & S_x \\ 0 & 1 & S_y \\ 0 & 0 & 1 \end{bmatrix}$$

Advantage of composition or concatenation of matrix:

a) The advantage of matrix composition or concatenation is that the resulting transformation can be represented in a compact form

b) It can reduce the number of operations required to achieve a particular transformation.

c) Rules used for defining transformation in form of equations are complex as compared to matrix.

Let's consider an example of composite transformations using a sequence of translation, scaling, and rotation transformations.

Suppose we have a rectangle with vertices at (1,1), (1,-1), (-1,-1), and (-1,1) in the Cartesian coordinate system. We want to perform the following sequence of transformations:

Translate the rectangle by (2,3).

Scale the rectangle by a factor of 1.5 with respect to the origin.

Rotate the rectangle by 45 degrees counterclockwise about the origin.

Translate the rectangle by (-4,2).

We can represent each transformation as a matrix in column form:

Translation matrix T1 = [[1, 0, 2], [0, 1, 3], [0, 0, 1]]

Scaling matrix S = [[1.5, 0, 0], [0, 1.5, 0], [0, 0, 1]]

Rotation matrix R = [[cos(45), -sin(45), 0], [sin(45), cos(45), 0], [0, 0, 1]]

Translation matrix T2 = [[1, 0, -4], [0, 1, 2], [0, 0, 1]]

To perform the composite transformation, we multiply these matrices from right to left, i.e., T2RTS1.

Multiplying these matrices, we get the composite transformation matrix as:

[[1.06, -2.12, -2.6], [2.12, 1.06, 5.36], [0, 0, 1]]

To apply the composite transformation to the rectangle, we represent the vertices of the rectangle as column vectors and multiply them with the composite transformation matrix. The transformed vertices of the rectangle will be at:

(2.94, 5.79), (0.94, 3.79), (-1.12, 5.79), and (-2.12, 3.79)

These transformed vertices represent the rectangle after applying the sequence of transformations to it.

2D viewing pipeline:

The 2D viewing pipeline is a process used in computer graphics to convert a 2D object or scene into an image that can be displayed on a computer screen.

The main steps involved in the 2D viewing pipeline include object definition, world coordinates, viewing coordinates, normalized coordinates, device coordinates, and rasterization. 2d viewing pipelining can be achieved by the following way:

a) **Object Definition:**

Define the 2D object or scene using geometric primitives such as lines, circles, and polygons.

b) ***World Coordinates:***

Define the world coordinate system to represent the position and orientation of the objects in the scene.

c) ***Viewing Coordinates:***

Define the viewing coordinate system to represent the position and orientation of the viewer in relation to the world coordinate system.

Use viewing transformations to position the objects within the viewing coordinate system.

d) **Normalized Coordinates:**

Transform the viewing coordinates to normalized coordinates that lie within a standard coordinate range, such as between 0 and 1 or between -1 and 1.

Use a normalization transformation that scales and translates the viewing coordinates to fit within the normalized coordinate range.

e) **Device Coordinates:**

Transform the normalized coordinates to device coordinates that can be displayed on the screen.

Use a device transformation that scales and translates the normalized coordinates to fit within the device coordinate range.

Rasterize the vector graphics primitives into a pixel-based format that can be displayed on the screen.

window to viewport coordinate transformation:

a) **Define the window:**

The window is the rectangular area in the object space that is to be displayed on the screen.

The window is defined by its lower left corner (xmin, ymin) and its upper right corner (xmax, ymax).

b) **Define the viewport:**

The viewport is the rectangular area on the screen where the image of the window will be displayed.

The viewport is defined by its lower left corner (xvmin, yvmin) and its upper right corner (xvmax, yvmax).

c) **Perform scaling:**

The object coordinates in the window need to be scaled to fit within the viewport.

Calculate the scaling factors Sx and Sy as follows:

Sx = (xvmax - xvmin) / (xmax - xmin)

Sy = (yvmax - yvmin) / (ymax - ymin)

d) **Perform translation:**

The scaled object coordinates need to be translated to align with the viewport.

Calculate the translation factors Tx and Ty as follows:

$Tx = xvmin - (xmin * Sx)$

$Ty = yvmin - (ymin * Sy)$

e) **Apply the transformation:**

Apply the scaling and translation factors to the object coordinates to obtain the corresponding viewport coordinates.

Clipping (Cohen Sutherland line clipping, Liang-Barsky Line Clipping):

Clipping is a process used in computer graphics to remove or discard any parts of an object or scene that fall outside of a specified region, or viewing frustum, in order to optimize rendering and improve performance. There are several types of clipping, including:

a) **Point clipping:**

Point clipping is used to remove any points that fall outside of the viewing frustum or viewport.

This is typically done by comparing the x and y coordinates of the point to the boundaries of the viewport and discarding any points that fall outside of these boundaries.

b) **Line clipping:**

Line clipping is used to remove any parts of a line that fall outside of the viewing frustum or viewport.

This is typically done by dividing the line into smaller segments and testing each segment to determine if it intersects with the viewing frustum or viewport. Any segments that do not intersect are discarded.

c) **Polygon clipping:**

Polygon clipping is used to remove any parts of a polygon that fall outside of the viewing frustum or viewport.

This is typically done by dividing the polygon into smaller triangles and testing each triangle to determine if it intersects with the viewing frustum or viewport. Any triangles that do not intersect are discarded.

d) **Curve clipping:**

Curve clipping is used to remove any parts of a curve that fall outside of the viewing frustum or viewport.

This is typically done by dividing the curve into smaller segments and testing each segment to determine if it intersects with the viewing frustum or viewport. Any segments that do not intersect are discarded.

e) **Text clipping:**

Text clipping is used to remove any parts of text or characters that fall outside of the viewing frustum or viewport.

This is typically done by testing each character to determine if it intersects with the viewing frustum or viewport. Any characters that do not intersect are discarded.

Line Clipping Algorithm:

Line clipping algorithms are used in computer graphics to remove any parts of a line that fall outside of the viewing frustum or viewport. Some commonly used line clipping algorithms include:

a) **Cohen-Sutherland line clipping algorithm:**

The Cohen-Sutherland algorithm is a simple and efficient line clipping algorithm that uses a 4-bit binary code to represent the location of a point relative to the viewport boundaries.

The viewport is divided into 9 regions, with 4 regions in the corners, 2 regions on the sides, and 1 region in the center.

Each point is assigned a 4-bit binary code based on its position relative to the viewport boundaries.

The line is tested against the viewport boundaries to determine if it is inside or outside the viewport. If the line is outside, the algorithm computes the intersection points of the line with the viewport boundaries and clips the line accordingly.

b) **Liang-Barsky line clipping algorithm:**

The Liang-Barsky algorithm is a more efficient line clipping algorithm than the Cohen-Sutherland algorithm and uses parameterization to calculate the intersections of the line segment with the viewport boundaries.

The viewport is defined by its four boundaries: xmin, xmax, ymin, and ymax.

The algorithm first calculates the values of the parameter t that define the intersection points of the line with the viewport boundaries.

The algorithm then checks these values to determine if the line is inside or outside the viewport. If the line is outside, the algorithm computes the intersection points of the line with the viewport boundaries and clips the line accordingly.

c) **Sutherland-Hodgman algorithm:**

The Sutherland-Hodgman algorithm is a polygon clipping algorithm that can also be used to clip lines.

The algorithm clips the line by computing the intersections of the line with each clipping edge of the viewport.

6.6 THREE-DIMENSIONAL TRANSFORMATION

Three-dimensional transformations are used to manipulate the positions and orientations of objects in a 3D space. Similar to 2D transformations, 3D transformations can also be represented using matrices, which allows multiple transformations to be combined and applied as a single transformation.

There are various types of 3D transformations, including translation, rotation, scaling, shearing, and reflection.

a. **Translation**: Translating a 3D object involves moving it in space along one or more of the three axes. This can be achieved by adding a translation vector to the position of each vertex of the object.

Three-dimensional translation is the process of moving a point or an object in a 3D space along one or more of the three axes. The general equation for a 3D translation can be written as:

$$T(x, y, z) = (x + dx, y + dy, z + dz)$$

where (x, y, z) are the original coordinates of the point or object, and (dx, dy, dz) are the translation distances along the x, y, and z axes, respectively. The resulting coordinates of the point or object after the translation are (x + dx, y + dy, z + dz).

To understand this equation, let's consider an example. Suppose we have a point P with coordinates (1, 2, 3), and we want to translate it by a distance of (2, 3, 4) units along the x, y, and z axes, respectively. Using the general equation for 3D translation, we can calculate the new coordinates of the point as follows:

$$T(1, 2, 3) = (1 + 2, 2 + 3, 3 + 4) = (3, 5, 7)$$

Therefore, the new coordinates of the point after the translation are (3, 5, 7).

b. **Rotation**: Rotating a 3D object involves changing its orientation in space. This can be achieved by multiplying the position of each vertex of the object by a rotation matrix that represents the desired rotation.

c. **Scaling**: Scaling a 3D object involves changing its size in space. This can be achieved by multiplying the position of each vertex of the object by a scaling matrix that represents the desired scaling.

d. **Shearing**: Shearing a 3D object involves changing its shape by shifting its vertices along one or more axes while keeping the other axes fixed. This can be achieved by multiplying the position of each vertex of the object by a shear matrix that represents the desired shear.

e. **Reflection:** Reflecting a 3D object involves flipping it across a plane in 3D space. This can be achieved by multiplying the position of each vertex of the object by a reflection matrix that represents the desired reflection.

3D viewing pipeline:

a) **Modeling transformation:**

The modeling transformation is used to create the 3D model of the scene or object using various modeling tools and techniques.

The model is defined in a local coordinate system and must be transformed to a global coordinate system for further processing.

The modeling transformation involves translating, rotating, and scaling the model as necessary.

b) **Viewing transformation:**

The viewing transformation is used to transform the model from the global coordinate system to the camera coordinate system.

This transformation involves setting up the camera parameters such as position, orientation, and field of view.

The viewing transformation is used to position and orient the camera in the virtual world so that it captures the desired view of the scene.

c) **Projection transformation:**

The projection transformation is used to transform the camera coordinates to normalized device coordinates.

This involves mapping the camera coordinates to a 2D plane that represents the image on the screen.

The projection transformation can be either orthographic or perspective depending on the desired projection type.

d) **Workstation transformation:**

The workstation transformation is used to transform the normalized device coordinates to device coordinates for display on the screen.

This involves mapping the normalized device coordinates to the pixel coordinates on the screen.

The workstation transformation takes into account the characteristics of the display device such as resolution, aspect ratio, and pixel density.

Projection concepts (Orthographic, parallel, perspective projection):

Projection is the process of converting a 3D scene into a 2D image that can be displayed on a screen. There are several types of projections that are commonly used in computer graphics, including orthographic, parallel, and perspective projections. Here are some key concepts and examples of each type:

Orthographic Projection: In an orthographic projection, the image is projected onto a plane parallel to the viewing direction. This type of projection is useful for technical drawing, engineering, and architectural applications. Some key concepts include:

- All parallel lines in the scene remain parallel in the projection.
- The size of an object in the projection is proportional to its size in the scene.
- There is no foreshortening or perspective distortion in the projection.
- An example of an orthographic projection would be a floor plan or elevation view of a building, where all the lines are parallel and the sizes of the objects are proportional to their actual sizes.

Parallel Projection: In a parallel projection, the image is projected onto a plane at an angle perpendicular to the viewing direction. This type of projection is used in computer graphics and video games to create a 2D representation of a 3D scene. Some key concepts include:

- All lines that are parallel to the viewing direction appear parallel in the projection.

- The size of an object in the projection is not proportional to its size in the scene.
- There is no foreshortening, but there can be perspective distortion if the angle of the projection is not perpendicular to the viewing direction.
- An example of a parallel projection would be an isometric or axonometric view of a 3D object in a video game, where the sizes of the objects are not proportional to their actual sizes, but the parallel lines appear parallel in the projection.

Perspective Projection: In a perspective projection, the image is projected onto a plane at an angle that is not perpendicular to the viewing direction. This type of projection is used to create a sense of depth and realism in computer graphics and virtual reality applications. Some key concepts include:

- Objects that are farther away from the viewer appear smaller in the projection.
- Parallel lines that are not parallel to the viewing direction appear to converge at a vanishing point in the projection.
- There is foreshortening and perspective distortion in the projection.

MULTIPLE CHOICE QUESTIONS

Theory of Computation

1. **Let R_1 and R_2 be regular sets defined over alphabet $\sum$ then**
   A. $R_1 \cup R_2$ is regular
   B. $\sum \cap R_2$ is not regular
   C. $R_1 \cap R_2$ is not regular
   D. R_2^* is not regular

2. **Consider the production of the grammar**
   $S \rightarrow AA$
   $A \rightarrow aa$
   $A \rightarrow bb$
   Describe the language specified by the production grammar
   A. L = {aaaa, aabb, bbaa, bbbb}
   B. L = {abab, abaa, aaab, baaa}
   C. L = {aaab, baba, bbaa, bbbb}
   D. L = {aaaa, ababa, bbaa, aaab}

3. **Give a production grammar that specified language**
   $L = \{a^i b^{2i} / i \geq 1\}$
   A. {S → aSbb, S → abb}
   B. {S → aA, S → b, A → b}
   C. {S → aSb, S → b
   D. None of the above

4. **Which of the following string can be obtained by the language**
   $L = \{a^i b^{2i} / i \geq 1\}$
   A. Aaabbbbbb
   B. Abbabbba
   C. aabb
   D. aaaabbbabb

5. **Give a production grammar for the lanaguae L ={x/x ∈ (a, b)*, the number of a's in x is multiple of 3}**
   A. {S → bS, S → b, S → aA, S → bA, A → aB, B → bB, B → aS, S → a}
   B. {S →aS, S → bA, A → bB, B → bBa, B → bB}
   C. {S → aaS, S → bbA, A →bB, B → ba}
   D. None of the above

6. **Let L1 = $\{a^i b^j / i > j\}$ and L2 = $\{a^i b^j / i < j\}$. the union of L1 and L2 is given by**
   A. $\{a^i b^j / i, > j \geq 1\}$
   B. $a^i b^j / i > j \geq 1\}$
   C. $\{a^i b^j / i, j \geq 1\}$
   D. $\{a^i b^j / i, j \geq 1, i \neq j\}$

7. **Give a production grammar for the language**
   $L = \{a^i b^j / I, j \geq 1, i \neq j\}$
   A. {S → aS, S → aB, B → ab, A →aaB, B → b}
   B. {S → A, S → C, A → aA A → aB, B → aBb, B → ab, C → Cb, ∈ → Bb}
   C. {S → A, A → aA, A → aB, B → ab}
   D. None of the above

8. **The production grammar is {S → aSbb, S → abb} is**
   A. Type-3 grammar
   B. Type-1 grammar
   C. Type-3 grammar
   D. Type-0 grammar