**CSC 320 2020/2021 REVISION**

**QUESTION ONE (COMPULSORY) [30 MARKS]**

**a) Define the following terms as used in computer graphics.**

**i. Rotation**

In computer graphics, rotation refers to the transformation of an object or image by changing its orientation around a fixed point or axis.

**ii. Clipping**

Clipping is the process of selectively removing parts of an object or image that fall outside of a defined region or boundary, often the viewable area or a specified window.

**b) A video is one of the most common computer graphics in use, giving reasons, which category of computer graphics will you put it. [4Marks]**

A video falls under the category of **dynamic computer graphics**. Dynamic computer graphics involve elements that change over time, such as animations, videos, and interactive simulations.

**c) You are given a large graphic bigger than the device screen, how will you handle it so that it is seen clearly by the user.**

**[6Marks]**

To handle a large graphic so that it is seen clearly by the user, you can employ techniques such as scaling, panning, or providing interactive controls for zooming in and out. Additionally, you could implement algorithms for level of detail (LOD) rendering, where different levels of detail are displayed based on the viewer's distance from the graphic.

**d) C initgraph () function is one of the most important functions when working with computer graphics, Explain its use and the parameters it takes?**

**[4Marks]**

The `initgraph()` function in C initializes the graphics system. It sets up the graphics environment and opens a graphics window on the screen. The parameters it takes typically include the graphics driver to be used, the graphics mode (such as graphics mode number or resolution), and other settings specific to the graphics driver being used.

**e) It is said that the future of computing is computer graphics, explain areas that computer graphics can be used in modern day.**

**[4Marks]**

**Entertainment and media:** Video games, movies, virtual reality (VR), augmented reality (AR), and animation.

**Design and visualization:** CAD (computer-aided design), architectural visualization, product design, and simulation.

**Education and training:** Interactive learning tools, simulations, medical imaging, and virtual laboratories.

**Data visualization:** Graphs, charts, infographics, and visual analytics.

**Human-computer interaction (HCI):** User interfaces, gesture recognition, and touchscreens.

**Scientific and engineering visualization:** Molecular modeling, weather simulations, fluid dynamics, and computational biology.

**f) Suppose you are required to rotate a point (x, y) clockwise through an 60° about the origin of the coordinate system. Give the resulting points. [4Marks]**

\[

\begin{bmatrix}

\cos(\theta) & -\sin(\theta) \\

\sin(\theta) & \cos(\theta)

\end{bmatrix}

\]

where \(\theta\) is the angle of rotation in radians (60° = π/3 radians).

The resulting point (x', y') after rotation is calculated as:

x' = x \times \cos(\theta) - y \times \sin(\theta)

y' = x \times \sin(\theta) + y \times \cos(\theta)

Substituting the values, we get:

x' = x \times \cos(\pi/3) - y \times \sin(\pi/3)

y' = x \times \sin(\pi/3) + y \times \cos(\pi/3)

Calculating the values of \(\cos(\pi/3)\) and \(\sin(\pi/3)\), we have:

\cos(\pi/3) = \frac{1}{2}

\sin(\pi/3) = \frac{\sqrt{3}}{2}

So, the resulting points are:

x' = \frac{x}{2} - \frac{\sqrt{3}}{2} \times y

y' = \frac{x}{2} + \frac{\sqrt{3}}{2} \times y

**Write a C program that uses ellipse () to draw ¼ with radius of 50px g [4Marks]**

#include <graphics.h>

#include <conio.h>

int main() {

int gd = DETECT, gm;

initgraph(&gd, &gm, "");

// Drawing the quarter ellipse

int x = 100, y = 100; // Center of the ellipse

int rx = 50, ry = 50; // Radius of the ellipse

ellipse(x, y, 0, 90, rx, ry);

getch();

closegraph();

return 0;

}

**QUESTION TWO [20 MARKS]**

**a) Explain the following terms.**

**i) Singularity Problem**

The singularity problem in computer graphics occurs when a transformation operation, such as scaling or rotation, results in a degenerate or non-invertible transformation matrix. This can happen, for example, when an object is scaled to a size of zero or when two or more points become coincident after transformation, leading to a loss of information. The singularity problem can cause errors or unexpected behavior in rendering algorithms and needs to be addressed to ensure accurate and reliable graphics output.

**ii) Coherence**

Coherence in computer graphics refers to the property of spatial or temporal consistency in data or operations. Spatial coherence implies that nearby pixels or primitives in an image are likely to have similar attributes or be subject to similar operations. Temporal coherence implies that successive frames or updates in an animation or simulation are similar or closely related. Exploiting coherence can lead to more efficient algorithms and data structures, reducing redundant computation and memory access.

**b) Describe how Algorithm singularity works.**

**[4Marks]**

Algorithm singularity is a method used in computer graphics to handle the singularity problem during transformations. It involves detecting when a singularity is about to occur and applying special handling to avoid or mitigate the problem. For example, when scaling an object, if the scale factor approaches zero, instead of applying the scale directly, the algorithm might switch to a different approach, such as using a minimum non-zero scale factor or applying a different transformation altogether.

**c) The Direct View Storage Tube ensured that persistence of graphics on the screen was greatly improved, by using a suitable diagram, explain how this was accomplished.**

The Direct View Storage Tube (DVST) improved the persistence of graphics on the screen by employing a phosphor-coated screen that could retain the displayed image for an extended period. This was accomplished using a diagram of the DVST setup:

[Diagram description: The DVST consists of a cathode ray tube (CRT) with a phosphor-coated screen. When an electron beam from the CRT strikes the phosphor-coated screen, it excites the phosphor molecules, causing them to emit light. The emitted light forms the displayed image. The phosphor has a certain persistence, meaning it continues to emit light for a short time after the electron beam stops. This persistence ensures that the displayed image remains visible even after the electron beam moves on to draw the next frame.]

**d) Most CRTs have a problem of persistence which needs pictures to be refreshed after every 100 Milliseconds, Explain how this problem can be eliminated.**

The persistence problem in CRTs, where images need to be refreshed frequently to avoid flickering or degradation, can be addressed by increasing the refresh rate. This can be achieved by:

**Higher Refresh Rates:** Increasing the frequency at which the electron beam scans the screen, ensuring that each frame is redrawn more frequently, such as increasing the refresh rate from 100 milliseconds to 60 Hz (refreshing every 16.67 milliseconds).

**Double Buffering:** Using double buffering techniques where two frames are alternately drawn and displayed, while the electron beam continuously refreshes the screen. This reduces flickering and provides smoother animation by eliminating the visibility of the redraw process.

**Faster Electronics:** Utilizing faster electronics and signal processing capabilities to decrease the time required for each frame update, allowing for more frequent refreshing without sacrificing image quality or performance.

**QUESTION FOUR [20 MARKS]**

**a) Describe the following terms as used in Computer Graphics.**

**i) Rotation**

Rotation in computer graphics refers to the transformation of an object or image by changing its orientation around a fixed point or axis. This transformation involves rotating each point of the object around the specified axis by a certain angle. Rotation can occur in two-dimensional (2D) or three-dimensional (3D) space and can be clockwise or counterclockwise.

**ii) Translation**

Translation in computer graphics involves moving an object from one position to another in a straight-line path, without altering its orientation or shape. This transformation shifts every point of the object by a fixed distance in a specified direction. Translation can occur in 2D or 3D space along the x, y, and z axes.

**b) Given the differential equation of a circle if dy/dy=-x/y, write a C program that generates a circle using DDA**

#include <graphics.h>

#include <math.h>

int main() {

int gd = DETECT, gm;

initgraph(&gd, &gm, "");

int x\_center = 200, y\_center = 200, radius = 100;

float x, y, dx, dy, length;

x = 0;

y = radius;

dx = 1;

dy = -1;

length = radius \* sqrt(2);

while (x <= length) {

putpixel(x\_center + round(x), y\_center + round(y), WHITE);

x = x + dx;

y = y + dy;

}

getch();

closegraph();

return 0;

}

**c) Explain how Bresenham's algorithm works.**

**[6Marks]**

Bresenham's algorithm is a line-drawing algorithm used in computer graphics to efficiently draw lines on a raster display. It works by determining the pixels to be illuminated to approximate the line between two given points with integer coordinates. Here's how it works:

- Begin with the coordinates of the start and end points of the line.

- Calculate the slope of the line (m = Δy / Δx).

- Based on the slope, decide which direction to move in each step (either along x or y).

- Calculate the decision parameter (P) based on the distance between the actual midpoint and the ideal midpoint.

- Based on the decision parameter, determine which pixel to illuminate in each step.

- Repeat the process until the end point is reached.

**d) Using a Rectangle, write a C program that shows rotation of 80° about the origin of the coordinate system.**

**[4Marks]**

#include <graphics.h>

#include <math.h>

int main() {

int gd = DETECT, gm;

initgraph(&gd, &gm, "");

int x1 = 100, y1 = 100; // Top-left corner

int x2 = 200, y2 = 200; // Bottom-right corner

int angle = 80; // Rotation angle in degrees

// Convert angle to radians

float theta = angle \* (3.14159 / 180);

// Rotation matrix

int x1\_new = round(x1 \* cos(theta) - y1 \* sin(theta));

int y1\_new = round(x1 \* sin(theta) + y1 \* cos(theta));

int x2\_new = round(x2 \* cos(theta) - y2 \* sin(theta));

int y2\_new = round(x2 \* sin(theta) + y2 \* cos(theta));

// Draw the rotated rectangle

rectangle(x1\_new, y1\_new, x2\_new, y2\_new);

getch();

closegraph();

return 0;

}

**QUESTION FIVE [20 MARKS]**

**a) Define the following terms.**

**[2Marks]**

1. **Clipping**

Clipping in computer graphics refers to the process of selectively removing parts of an object or image that fall outside of a specified region or boundary, typically the viewable area or a designated window. Clipping ensures that only the portions of the object or image that are within the defined boundaries are displayed on the screen.

1. **Windowing**

Windowing in computer graphics involves defining a rectangular region on the screen known as a window. This window represents the portion of the screen where graphics operations are performed or displayed. Windowing allows for the isolation and manipulation of specific areas of the screen, enabling efficient management of graphical content.

**b) Why do you think there is need for Clipping and Windowing?**

**[4Marks]**

**Efficient Rendering:** Clipping ensures that only the visible portions of objects or images are rendered, reducing unnecessary computation and improving rendering performance.

-**Viewport Control:** Windowing allows for the definition of multiple viewports or regions on the screen, enabling the display of different graphical content simultaneously or the focus on specific areas of interest.

**- User Interaction:** Clipping and windowing facilitate user interaction by restricting graphical operations to designated areas, allowing users to interact with specific elements of the graphical interface.

**- Visual Presentation:** Clipping ensures that objects or images are displayed within the bounds of the screen or designated viewing area, preventing visual artifacts and enhancing the overall presentation of graphics.

**c) There exist many clipping algorithms, in your own view give the best algorithm, explain and justify your choice.**

**[6Marks]**

The best clipping algorithm depends on the specific requirements and constraints of the application. However, one commonly used and efficient clipping algorithm is the **Cohen-Sutherland line clipping algorithm.** This algorithm divides the 2D space into nine regions based on the clipping window and efficiently determines which portions of a line lie inside or outside the window.

Explanation and Justification:

- **Efficiency:** The Cohen-Sutherland algorithm is efficient as it categorizes each endpoint of a line into one of the nine regions using bitwise operations, allowing for quick determination of visibility.

- **Simplicity:** The algorithm is relatively simple to implement and understand, making it suitable for a wide range of applications.

- **Flexibility:** It can handle arbitrary window shapes and positions, providing flexibility in defining clipping regions.

- **Optimization:** Various optimizations, such as rejecting lines entirely outside the window, can be applied to improve performance further.

- **Widely Used:** The Cohen-Sutherland algorithm is widely used and well-studied, with many resources and implementations available, making it a practical choice for clipping in computer graphics.

**d) Positioning is one of the basic graphical input operations, explain using different techniques how this can be accomplished.**

**[6Marks]**

- **Coordinate Transformations:** Use mathematical transformations such as translation, rotation, scaling, and shearing to position objects within the coordinate system.

- **Relative Positioning:** Position objects relative to other objects or reference points using offsets, alignments, or anchor points.

- **User Interaction:** Allow users to position objects interactively through direct manipulation or input devices like mouse or touch input.

- **Layout Managers:** Employ layout managers or grids to automatically position and arrange graphical elements according to predefined rules or constraints.

- **Constraint-Based Positioning:** Define positioning rules or constraints to specify the relationships and arrangements of graphical elements dynamically.

**e) Event handling is a principal behind many computer graphics fields, state two important functions for handling events in computer graphics.**

**[2Marks]**

**- Event Listener/Handler:** A function or method responsible for receiving and processing events generated by user interactions or system events, such as mouse clicks, keyboard input, or window resizing.

**- Callback Function:** A function registered to be invoked in response to specific events. Callback functions allow developers to define custom behavior or actions to be executed when particular events occur, enhancing the interactivity and responsiveness of graphical applications.

**QUESTION THREE [20 MARKS]**

a) Explain the following concepts used in graphics programing.

[4Marks]

a. Coordinate System

b. Graph Mode

b) Write C program to draw the following figures using lines.

18Marks]

ii)

c) Write a C program that keeps drawing a circle of random centre and a radius of 20 Pixels throughout the screen until a user presses any key from the keyboard.

[4Marks]

d) Using fill effects of your choice, write a C program that draws a rectangle that has the fill effects of your choice.

[4Marks]

**a)**

**i) \*\*Coordinate System\*\*:**

**- In graphics programming, a coordinate system is a mathematical framework used to represent and manipulate the position of objects or points in a graphical environment.**

**- It typically consists of two or three axes (X, Y, and optionally Z) that intersect at a common origin point.**

**- Each axis represents a dimension, such as horizontal (X-axis), vertical (Y-axis), and depth (Z-axis) in three-dimensional space.**

**- Points in the coordinate system are specified using coordinates, which are ordered sets of numbers that indicate their position along each axis relative to the origin.**

**- Depending on the programming environment or application, the orientation and scaling of the coordinate system may vary.**

**ii) \*\*Graph Mode\*\*:**

**- Graph mode refers to a specific graphics rendering mode or configuration used by a graphics library or environment.**

**- It determines the characteristics of the graphical output, such as resolution, color depth, and available drawing functions.**

**- Graph mode settings include parameters such as screen size, color palette, pixel density, and supported drawing primitives.**

**- Different graph modes may offer varying levels of graphical fidelity and performance, allowing developers to choose the most suitable mode for their application's requirements.**

**- Switching between graph modes may involve initializing or reconfiguring the graphics environment, which can impact the appearance and behavior of graphical output.**

**b)**

**```c**

**#include <graphics.h>**

**#include <stdlib.h>**

**#include <time.h>**

**int main() {**

**int gd = DETECT, gm;**

**initgraph(&gd, &gm, "");**

**// Drawing a square**

**line(100, 100, 200, 100);**

**line(200, 100, 200, 200);**

**line(200, 200, 100, 200);**

**line(100, 200, 100, 100);**

**// Drawing a triangle**

**line(300, 100, 400, 100);**

**line(400, 100, 350, 200);**

**line(350, 200, 300, 100);**

**// Drawing a pentagon**

**line(500, 100, 600, 120);**

**line(600, 120, 620, 180);**

**line(620, 180, 550, 220);**

**line(550, 220, 480, 180);**

**line(480, 180, 500, 100);**

**getch();**

**closegraph();**

**return 0;**

**}**

**```**

**c)**

**```c**

**#include <graphics.h>**

**#include <stdlib.h>**

**#include <time.h>**

**int main() {**

**int gd = DETECT, gm;**

**initgraph(&gd, &gm, "");**

**srand(time(NULL)); // Seed for random number generation**

**while (!kbhit()) { // Continue drawing until key is pressed**

**int x\_center = rand() % getmaxx(); // Random x-coordinate within screen width**

**int y\_center = rand() % getmaxy(); // Random y-coordinate within screen height**

**int radius = 20; // Radius of the circle**

**circle(x\_center, y\_center, radius); // Draw the circle**

**delay(500); // Delay for 500 milliseconds before drawing the next circle**

**cleardevice(); // Clear the screen**

**}**

**getch();**

**closegraph();**

**return 0;**

**}**

**```**

**d)**

**```c**

**#include <graphics.h>**

**int main() {**

**int gd = DETECT, gm;**

**initgraph(&gd, &gm, "");**

**// Draw rectangle outline**

**rectangle(100, 100, 300, 200);**

**// Fill rectangle with a pattern**

**setfillstyle(HATCH\_FILL, BLUE);**

**floodfill(150, 150, WHITE);**

**getch();**

**closegraph();**

**return 0;**

**}**

**```**

QUESTION THREE [20 MARKS] [4Mar ) Explain the relationship between the following in computer graphics. a a. Pixel and Resolution b. Graphic Clarity and Pixels Gaure [6Mar

**a)**

**i) \*\*Pixel and Resolution\*\*:**

**- \*\*Pixel\*\*: A pixel (short for "picture element") is the smallest unit of display in a digital image or screen. It represents a single point in the image grid and contains color and brightness information. Pixels are arranged in a grid pattern, and each pixel's color and intensity contribute to the overall appearance of the image.**

**- \*\*Resolution\*\*: Resolution refers to the number of pixels contained in a digital image or display, typically expressed as the number of pixels along the width and height of the image. It determines the level of detail and clarity of the image, with higher resolutions providing finer detail and smoother images.**

**\*\*Relationship\*\*:**

**- The relationship between pixels and resolution is straightforward: the resolution of an image or display directly corresponds to the number of pixels it contains. A higher resolution means more pixels, while a lower resolution means fewer pixels.**

**- Resolution is often specified as the total number of pixels along the width and height of the image. For example, a resolution of 1920x1080 indicates an image with 1920 pixels in width and 1080 pixels in height.**

**- The density of pixels in an image affects its clarity and sharpness. Higher resolutions with more pixels per unit area result in sharper images with finer detail, while lower resolutions with fewer pixels per unit area may appear more pixelated or less detailed.**

**- In computer graphics, the resolution of an image or display determines the level of detail that can be represented and the clarity of the graphical content. Higher resolutions are desirable for producing high-quality images with crisp detail, especially in applications such as digital art, photography, and graphic design.**

**b)**

**\*\*Graphic Clarity and Pixels\*\*:**

**- \*\*Graphic Clarity\*\*: Graphic clarity refers to the visual quality and sharpness of an image or graphical content. It is influenced by factors such as resolution, pixel density, color accuracy, and display technology. A clear graphic is one that displays fine detail, smooth transitions, and accurate colors, resulting in a visually pleasing and easily recognizable image.**

**- \*\*Pixels\*\*: Pixels are the building blocks of digital images and graphical content. They represent individual points of color and brightness in the image grid and are responsible for forming the visual representation of the image. The arrangement, density, and quality of pixels determine the level of detail and clarity of the image.**

**\*\*Relationship\*\*:**

**- Pixels play a crucial role in determining the clarity of graphics. The density of pixels in an image affects its clarity, with higher pixel densities contributing to sharper and more detailed images.**

**- Higher resolutions with more pixels per unit area generally result in clearer graphics with finer detail and smoother edges. This is because higher pixel densities can represent more information and capture subtle variations in color and intensity.**

**- However, simply increasing the number of pixels does not guarantee graphic clarity. Factors such as the quality of the display technology, color accuracy, and image processing algorithms also play significant roles in determining the overall clarity and visual fidelity of graphics.**

**- In summary, while pixels are essential for forming the visual representation of graphics, graphic clarity depends on various factors, including pixel density, resolution, display quality, and image processing techniques. Increasing the number of pixels alone may improve clarity, but other factors must also be considered to achieve optimal graphic quality.**