MASTER OF COMPUTER APPLICATIONS

SEMESTER 1

PROGRAMMING & PROBLEM-SOLVING USING C

Unit 14
Graphical Programming in C

Table of Contents

1. INTRODUCTION

In the previous unit, we learnt about Pointers to pointers, also known as double pointers, which store another pointer variable's memory address. They provide a way to access and manipulate memory locations indirectly. Essentially, a pointer to a pointer holds the address of another pointer, allowing for multiple levels of indirection. This concept is particularly useful in scenarios where dynamic memory allocation and manipulation are necessary, such as in data structures like linked lists and trees. We also saw Pointers to functions, which are variables that store the address of a function rather than a data value. They enable the creation of more flexible and adaptable code structures by allowing functions to be passed as arguments to other functions or stored within data structures.

This unit deals with Graphical programming in C and introduces developers to the environment of creating visual interfaces and graphics within the C programming language. It includes a variety of concepts essential for designing graphical user interfaces (GUIs) and visual representations of data. Understanding these concepts is crucial for building interactive applications and games. To begin programming with graphics in the C computer language, developers must first set up the graphics environment by installing and configuring necessary libraries such as SDL (Simple DirectMedia Layer) or Allegro. This includes selecting appropriate Integrated Development Environments (IDEs) and compilers and understanding header files required for graphics programming.

Once the environment is established, developers can create simple programs to explore basic drawing operations like drawing points, lines, and shapes. We will explore the techniques for manipulating colours and filling shapes while mastering coordinate systems and transformations. These fundamental skills lay the groundwork for creating more complex graphical applications, including games, scientific visualisations, and multimedia presentations, highlighting the versatility and practical applications of graphics programming in C.

1.1. Objectives:

At studying this unit, you should be able to:

- ❖ *List the reasons why graphical programming in C is important.*

- ❖ *Identify the steps in Setting up Graphics Environment in C.*
- ❖ *Explain IDEs (Integrated Development Environments).*
- ❖ *Demonstrate the basic Drawing Operations.*

2. INTRODUCTION TO GRAPHICAL PROGRAMMING IN C

Graphical programming in C involves using libraries and frameworks to create user interfaces (UIs) and graphical applications using the C programming language. While C is traditionally known for its system-level and high-performance capabilities, modern advancements in libraries have enabled developers to build sophisticated graphical applications with C. Graphical programming in C opens a world of possibilities for creating interactive software, ranging from simple GUIs to complex multimedia applications. Developers can create efficient and portable solutions that fulfil unique user needs across many platforms by utilising the capabilities of C and graphics libraries.

2.1. Overview of Graphical Programming Concepts

Graphical programming in C involves creating visual output on a display device like a monitor or screen. It encompasses a set of techniques and tools for rendering graphics, handling user input, and creating interactive user interfaces.

Let us see some key concepts:

- Drawing Primitives: Graphical programming allows you to draw basic shapes such as points, lines, rectangles, circles, and polygons on the screen. These primitives are the building blocks for creating more complex images and graphical elements.
- Coordinate Systems: Graphics programming typically involves working with coordinate systems to specify the position and size of graphical elements on the screen. Understanding how coordinates are mapped to pixels on the display is essential for accurate rendering.
- Color and Style: Graphical programming lets you specify colours and line styles and fill patterns when drawing shapes and lines. This gives you control over the appearance of graphical elements and allows for creative expression in visual design.

- Input Handling: Interactivity is a crucial aspect of graphical programming. This involves handling user input such as mouse clicks, keyboard input, and touch events to create responsive and interactive applications.
- Animation and Multimedia: Graphics programming extends beyond static images to include animation, video playback, and multimedia capabilities. Techniques like frame-based animation and multimedia integration enrich the user experience and enable dynamic content creation.
- Hardware Acceleration: Modern graphics programming often leverages hardware acceleration to improve performance and efficiency. Graphics libraries and APIs provide access to hardware features such as graphics processing units (GPUs) for faster rendering and advanced graphical effects.

2.2. Importance and applications of graphical programming in C

Graphical programming in C is significant in various fields and applications due to its versatility and performance.

Key reasons why graphical programming in C is important:

- Game Development: C-based graphics libraries like SDL, Allegro, and OpenGL are widely used in game development for creating 2D and 3D games. Graphical programming allows game developers to bring their creative visions to life, from designing game worlds to animating characters and implementing gameplay mechanics.
- User Interfaces: Graphical programming is essential for creating graphical user interfaces (GUIs) in software applications. C-based GUI libraries enable developers to design intuitive and visually appealing interfaces with features such as buttons, menus, dialogs, and widgets.
- Data Visualization: Graphical programming is used in scientific, engineering, and business applications to visualise data and analyse results. Graphically representing complex data sets through charts, graphs, and diagrams, graphical programming aids in data interpretation and decision-making.

- Computer-Aided Design (CAD): CAD software relies heavily on graphical programming for creating and editing 2D and 3D designs. C-based graphics algorithms power the visualisation and manipulation of geometric shapes, allowing engineers and designers to draft architectural plans, mechanical drawings, and electronic schematics.
- Multimedia Applications: Graphical programming is integral to multimedia applications such as video players, image editors, and audiovisual software. By leveraging C-based graphics libraries, developers can implement features such as video playback, image processing, and audio rendering with high performance and efficiency.

Graphical programming in C enables developers to create visually engaging applications, games, simulations, and interactive experiences across various domains. Its importance and versatility make it a valuable skill for programmers seeking to develop graphical applications and multimedia software.

SELF-ASSESSMENT QUESTIONS - 1

1. What are drawing primitives in graphical programming?

   (a) Basic shapes used for creating complex images

   (b) Advanced rendering techniques

   (c) Input handling functions

   (d) Multimedia integration features

2. Why is understanding coordinate systems important in graphics programming?

   (a) To specify the colors and styles of graphical elements

   (b) To handle user input such as mouse clicks and keyboard events

   (c) To accurately position and size graphical elements on the screen

   (d) To integrate multimedia content into graphical applications

3. In which field is graphical programming in C widely used for creating 2D and 3D games?

   (a) Data visualisation

   (b) Multimedia applications

   (c) Game development

   (d) Computer-aided design (CAD)

3. SETTING UP GRAPHICS ENVIRONMENT IN C

3.1. Installing and configuring necessary libraries (e.g., SDL, Allegro)

Before diving into graphical programming in C, it's essential to install and configure the necessary libraries that provide the tools and functions for rendering graphics.

Two popular libraries for this purpose are SDL (Simple DirectMedia Layer) and Allegro. Here's how you can install and configure them:

- **SDL (Simple DirectMedia Layer):**

SDL is a cross-platform development library designed to provide low-level access to audio, keyboard, mouse, joystick, and graphics hardware.

To install SDL, you can typically use package managers like “apt” for Ubuntu-based systems or “brew” for macOS.

For example:

```
sudo apt-get install libsdl2-dev
```

Once installed, you need to include the SDL header file in your C programs and link it against the SDL library during compilation.

- **Allegro:**

Allegro is another popular multimedia library for C/C++ programming, offering support for graphics, sound, input devices, and more.

Installing Allegro varies depending on your platform. On Linux, you can typically install it using package managers:

```
sudo apt-get install liballegro5-dev
```

Similar to SDL, after installation, you include the Allegro header file in your C programs and link against the Allegro library during compilation.

3.2. Setting up a development environment (IDEs, compilers)

Once you have installed the necessary graphics libraries, setting up a development environment for writing and compiling your graphical C programs is next.

Let us see some options for IDEs and compilers:

IDEs (Integrated Development Environments):

IDEs provide an integrated environment for writing, compiling, and debugging code. Some popular IDEs for C programming include:

1. Visual Studio: A powerful IDE with comprehensive features for C/C++ development. It supports Windows development and offers a graphical interface for managing projects and debugging.
2. Code:Blocks: A cross-platform IDE that is lightweight and customisable. It supports multiple compilers and offers features like code completion and project management.
3. Eclipse: Although commonly associated with Java development, Eclipse supports C/C++ development with plugins like CDT (C/C++ Development Tools). It offers features like syntax highlighting, refactoring, and version control integration.
4. CLion: A commercial IDE by JetBrains, known for its intelligent code analysis and refactoring tools. It supports C/C++ development and integrates with CMake, a popular C-project build system.

Compilers:

In addition to choosing an IDE, you must select a C compiler to translate your C code into machine-readable instructions.

Let us see some common C compilers include:

1. GCC (GNU Compiler Collection): A widely used compiler suite that supports various programming languages, including C and C++. It is available on multiple platforms and is known for its optimisation capabilities.
2. Clang: Another popular compiler known for its fast compilation speed and comprehensive error messages. Clang is part of the LLVM project and is compatible with GCC in many aspects.

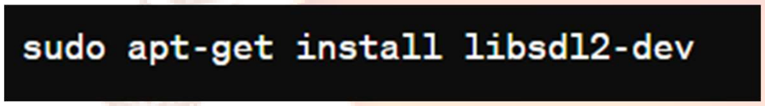
3. Microsoft Visual C++ Compiler: If you are using Visual Studio as your IDE, you can also use the Microsoft Visual C++ Compiler, which is included with Visual Studio installations on Windows.

Example:

Suppose you want to set up a development environment on Ubuntu for writing graphical programs using SDL.

Here's how you would do it:

- Install SDL using the package manager.



```
sudo apt-get install libsdl2-dev
```

- Choose an IDE, such as Visual Studio Code, and install it on your system.
- Install the C/C++ extension for Visual Studio Code to enable C/C++ development support.

SELF-ASSESSMENT QUESTIONS - 2

4. Which library is commonly used for graphical programming in C and provides low-level access to audio, keyboard, mouse, and graphics hardware?
   (a) GLFW
   (b) SDL (Simple DirectMedia Layer)
   (c) SFML (Simple and Fast Multimedia Library)
   (d) FreeGLUT

5. How can you install SDL on Ubuntu-based systems?
   (a) Using the package manager "apt-get install libsdl2-dev."
   (b) Using the package manager "yum install SDL2-devel."
   (c) Using the package manager "brew install sdl2"
   (d) Downloading the SDL source code and compiling it manually

5. Which IDE provides comprehensive features for C/C++ development, supports Windows development, and offers a graphical interface for managing projects and debugging?
   (a) Code: Blocks
   (b) Visual Studio
   (c) Eclipse
   (d) CLion

4. HEADER FILES

Libraries of Graphic Program in C

Some popular graphic libraries for C programming:

- OpenGL: It provides a powerful set of tools for developing 3D graphics applications.
- Simple Direct Media Layer: It is a multimedia library created to offer Direct3D and OpenGL low-level access to hardware for audio, keyboard, mouse, joystick, and graphics.
- Allegro: It is a popular multimedia library for C/C++ programming that provides basic functionality for graphics, sound, and input devices.
- GTK+ (GIMP Toolkit): It is a popular open-source widget toolkit for creating graphical user interfaces (GUIs) that are often used for developing desktop applications.
- Cairo: This is a vector graphics library that provides support for a wide range of output devices, including PNG, PDF, and SVG.
- SFML (Simple and Fast Multimedia Library): This is a multimedia library that provides a simple interface for creating 2D and 3D games and multimedia applications.
- GLUT (OpenGL Utility Toolkit): It is a utility library for OpenGL that provides a simple API for creating windows, handling input, and drawing 3D graphics.
- Qt: It is a GUI toolkit that provides a comprehensive set of tools for creating desktop and mobile applications.

<graphics.h>

graphics.h is a header file in C programming language which is used to work with graphics. It provides a set of functions and data types that allow you to create graphical applications, such as drawing shapes, lines, and text on the screen. Graphics.hh is typically used in DOS-based systems and older versions of the Windows operating system.

Let us see the key points about graphics.h:

- Drawing Functions: graphics.h provides functions to draw basic shapes such as lines, circles, rectangles, ellipses, and polygons. These functions allow you to specify the coordinates, size, and color of the shapes you want to draw.
- Color: You can set the color of the drawing pen and the fill color for shapes using functions provided by graphics.h. Colors are typically specified using integer values or predefined color constants.
- Text Output: graphics.h allows you to display text on the screen using functions like outtext() and outtextxy(). You can specify the position, font, and size of the text to be displayed.
- Coordinate System: Graphics in graphics. h are drawn on a coordinate system where the top-left corner of the screen is the origin (0,0). Positive x-values increase to the right, and positive y-values increase downwards.
- Initialization and Cleanup: Before using graphics.h functions, you need to initialise the graphics system using initgraph(). Similarly, after finishing your graphical operations, you should close the graphics system using closegraph().
- Mouse and Keyboard Input: graphics.h provides limited support for handling mouse and keyboard input. You can detect mouse clicks and key presses using functions like getch() and getmouseclick().

It's important to note that graphics.h is specific to certain compilers and environments, such as Turbo C++ for DOS or Borland C++ for Windows. It may not be compatible with modern compilers or operating systems, so its usage is limited to legacy systems or for educational purposes.

5. SIMPLE PROGRAM

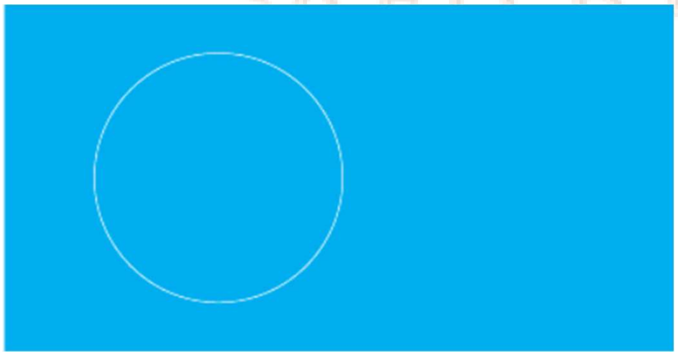
Syntax of Graphics Program in C

```
initgraph(int *gm, int *gm, char *driverDirectoryPath);
```

This function initialises the graphics driver and sets the graphics mode. The first parameter (gd) is a pointer to the graphics driver, which is set to DETECT to detect the graphics driver automatically. The second parameter (gm) is the graphics mode, which specifies the resolution and color depth of the screen. The last parameter is a string that can be used to pass additional arguments to the graphics driver, but in this case, it’s left empty.

```
#include <graphics.h>
#include <iostream>
#include <math.h>
#include <stdlib.h>
#include <ctype.h>
int main()
{
int gd = DETECT, gm;
char data[] = "C:\\MinGW\\lib\\libbgi.a";
initgraph( & gd, & gm, data);
//you can also pass NULL for 3rd parameter
//example: initgraph(&gd, &gm, NULL);
circle(400, 400, 100);
getch();
closegraph();
return 0;
}
```

Output:



Color values:

COLOR MACRO	INTEGER VALUE
BLACK	0
BLUE	1
GREEN	2
CYAN	3
RED	4
MAGENTA	5
BROWN	6
LIGHT GRAY	7
DARK GRAY	8
LIGHT BLUE	9
LIGHT GREEN	10
LIGHT CYAN	11
LIGHT RED	12
LIGHT MAGENTA	13
YELLOW	14
WHITE	15

SELF-ASSESSMENT QUESTIONS - 3

7. Which of the following is not a popular graphics library for C programming?
   (a) OpenGL
   (b) DirectX
   (c) Simple Direct Media Layer (SDL)
   (d) Allegro

8. What is the primary purpose of the graphics.h header file in C programming?
   (a) Working with multimedia files
   (b) Handling input events
   (c) Creating graphical applications
   (d) Managing memory allocation

9. What does the initgraph() function do in a graphics program using graphics.h?
   (a) Displays text on the screen
   (b) Initializes the graphics driver and sets the graphics mode
   (c) Handles mouse and keyboard input
   (d) Draws basic shapes such as lines and circles

6. BASIC DRAWING OPERATIONS

6.1. Drawing points, lines, and shapes

Drawing basic shapes like points, lines, and polygons is essential in graphical programming. Here's how you can achieve this using graphics.h in C:

Drawing Points:

Points can be drawn using the putpixel() function, which plots a single pixel at the specified coordinates.

```
putpixel(x, y, color);
```

Drawing Lines:

Lines can be drawn using the line () function, which draws a straight line between two given points.

```
line(x1, y1, x2, y2);
```

Drawing Shapes:

graphics.h provides functions to draw various shapes like rectangles, circles, ellipses, and polygons.

```
rectangle(x1, y1, x2, y2);
circle(x, y, radius);
ellipse(x, y, start_angle, end_angle, x_radius, y_radius);
```

6.2. Colour manipulation and filling shapes

Color manipulation and filling shapes with colors can enhance the visual appeal of graphical applications. Here's how you can manipulate colors and fill shapes using graphics.h:

Setting Color:

Colors can be set using the setcolor() function, which specifies the drawing color for subsequent operations.

```
setcolor(color);
```

Filling Shapes:

graphics.h provides functions to fill shapes with colors. For example, floodfill() fills an enclosed area starting from a given point with the current fill color.

```
floodfill(x, y, boundary_color);
```

6.3. Handling coordinates and transformations

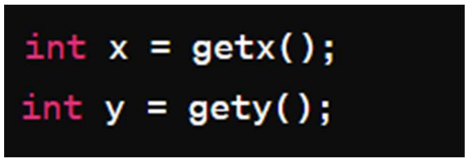
Efficient handling of coordinates and transformations is crucial for precise graphical rendering. Here's how you can manage coordinates and perform transformations using graphics.h:

- **Coordinate System:**
  graphics.h typically uses a coordinate system where the origin (0,0) is at the top-left corner of the screen. Positive x-values extend to the right, and positive y-values extend downwards.
- **Coordinate Transformation:**
  graphics.h provides functions for coordinate transformations like translation, rotation, scaling, and shearing. For example, translate() translates the current position by the specified offsets.

  ```
  translate(dx, dy);
  ```

Coordinate Conversion:

Functions like getx() and gety() can be used to retrieve the current drawing position. This is useful for performing calculations based on screen coordinates.

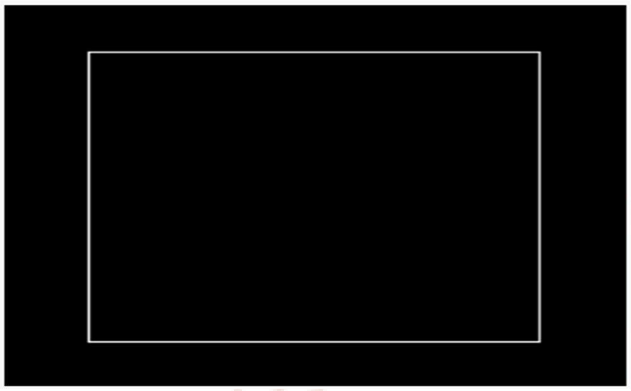


```c
int x = getx();
int y = gety();
```

Below is the implementation of the rectangle function :

```c
// C program to draw a rectangle
#include <graphics.h>

// Driver code
int main()
{
    // gm is Graphics mode which is a computer display
    // mode that generates image using pixels.
    // DETECT is a macro defined in "graphics.h" header file
    int gd = DETECT, gm;

    // location of left, top, right, bottom
    int left = 150, top = 150;
    int right = 450, bottom = 450;

    // initgraph initializes the graphics system
    // by loading a graphics driver from disk
    initgraph(&gd, &gm, "");

    // rectangle function
    rectangle(left, top, right, bottom);

    getch();

    // closegraph function closes the graphics
    // mode and deallocates all memory allocated
    // by graphics system .
    closegraph();

    return 0;
}
```

Output:



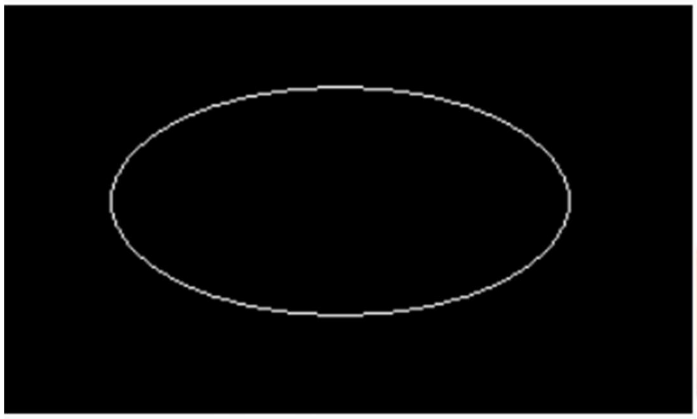
Example:

Program to draw ellipse in C using graphics.h header file.

The header file graphics.h contains ellipse() function which is described below : void ellipse(int x, int y, int start_angle, int end_angle, int x_radius, int y_radius) In this function x, y is the location of the ellipse. x_radius and y_radius decide the radius of form x and y. start_angle is the starting point of angle and end_angle is the ending point of angle. The value of angle can vary from 0 to 360 degree.

```
// C Implementation for drawing ellipse

#include <graphics.h>

 int main()

{

   // gm is Graphics mode which is a computer display

   // mode that generates image using pixels.

   // DETECT is a macro defined in "graphics.h" header file

   int gd = DETECT, gm;

     // location of ellipse

   int x = 250, y = 200;
  // here is the starting angle
   // and end angle
   int start_angle = 0;
   int end_angle = 360;
     // radius from x axis and y axis
   int x_rad = 100;
   int y_rad = 50;
     // initgraph initializes the graphics system
   // by loading a graphics driver from disk
   initgraph(&gd, &gm, "");
     // ellipse function
   ellipse(x, y, start_angle,
    end_angle, x_rad, y_rad);
     getch();
     // closegraph function closes the graphics
   // mode and deallocates all memory allocated
   // by graphics system .
   closegraph();
     return 0;
}
```

Output



7. APPLICATIONS OF GRAPHICS PROGRAM IN C

Graphics programming in C has a wide range of applications.

Let us see some examples:

- **Game Development:**

  Graphics programming is essential in game development for creating visually appealing environments, characters, and effects.

  C-based libraries like SDL and Allegro commonly handle graphics rendering, input handling, and other game-related tasks.

  Games utilise graphics programming for both 2D (e.g., side-scrolling platformers) and 3D (e.g., first-person shooters) environments.

- **Computer-Aided Design (CAD):**

  CAD software heavily relies on graphics programming to create, manipulate, and visualise 2D and 3D designs.

  Engineers, architects, and designers use CAD tools to draft plans, simulate structures, and precisely model complex systems.

- **Visualization and Data Analysis:**

  Graphics programming is crucial for creating visual representations of data in science, engineering, and finance fields.

  Data visualisation tools generate charts, graphs, maps, and interactive visualisations to help analysts and researchers interpret complex datasets.

- **Multimedia Applications:**

  Graphics programming plays a significant role in multimedia applications like video players, image viewers, and video editors.

  These applications use graphics to display and manipulate images, videos, and audio in various formats.

- **User Interfaces:**

  Graphics programming is essential for designing graphical user interfaces (GUIs) for software applications.

  GUI elements such as buttons, menus, sliders, and windows are created and rendered using graphics programming techniques.

- **Animation and Special Effects:**

  Graphics programming is widely employed in the creation of animations and special effects for movies, TV shows, and other media.

  Animators use techniques like keyframing, rigging, and rendering to bring characters and scenes to life with realistic motion and visual effects.

Advantages of using graphics programming in C:

- **Efficient Memory Management:**

  Graphics programming in C allows for efficient memory management, which is crucial for optimising performance in resource-intensive applications. Developers have fine-grained control over memory allocation and deallocation, reducing memory overhead and improving overall application efficiency.

- **Cross-Platform Compatibility:**
  Many graphics libraries available for C, such as SDL (Simple DirectMedia Layer) and OpenGL, are designed to be cross-platform. This allows developers to write graphics applications in C that can run on various operating systems without major modifications, enhancing portability and reaching a broader audience.
- **Visually Appealing Applications:**
  Graphics programming in C enables the creation of visually appealing applications with rich graphical interfaces, animations, and special effects. Developers can leverage advanced graphics techniques to enhance user experience and create immersive visual environments in games, multimedia applications, and design software.
- **Availability of Graphics Libraries:**
  C boasts a wide range of graphics libraries, providing developers with numerous tools and resources for creating diverse graphical applications. Libraries like SDL, Allegro, and OpenGL offer comprehensive features for rendering graphics, handling input, and managing multimedia resources, speeding up development and reducing the need for low-level coding.
- **Low-Level Control for Advanced Graphics:**
  Graphics programming in C offers low-level control over hardware resources, allowing developers to implement advanced graphics algorithms and optimisations. This level of control enables fine-tuning of rendering pipelines, shader programming, and direct access to hardware features, resulting in high-performance graphics applications tailored to specific requirements.

SELF-ASSESSMENT QUESTIONS - 4

10. Which of the following fields extensively relies on graphics programming in C for creating, manipulating, and visualising 2D and 3D designs?
    (a) Multimedia Applications
    (b) Visualization and Data Analysis
    (c) Computer-Aided Design (CAD)
    (d) User Interfaces

11. What advantage does graphics programming in C offer in terms of memory management?
    (a) Automatic memory allocation
    (b) Fine-grained control over memory allocation and deallocation
    (c) Dynamic memory resizing
    (d) Limited memory access
12. Which library is commonly used in game development for handling graphics rendering, input handling, and other game-related tasks?
    (a) OpenGL
    (b) Cairo
    (c) GTK+
    (d) Allegro

8. SUMMARY

- Graphical programming in C involves rendering visual output on display devices, utilising techniques for graphics rendering and user interaction. Key concepts include drawing primitives for basic shapes, working with coordinate systems, specifying colors and styles, handling input events, enabling animation and multimedia, and leveraging hardware acceleration. These concepts form the foundation for creating interactive and visually appealing applications across various domains.
- Graphical programming in C is crucial for diverse applications such as game development, user interfaces, data visualisation, CAD, and multimedia applications.
- C-based graphics libraries like SDL, Allegro, and OpenGL empower developers to create 2D/3D games and intuitive GUIs, visualise complex data, design CAD models, and develop multimedia software efficiently. Its versatility and performance make graphical programming in C a valuable skill for creating visually engaging applications and interactive experiences in numerous domains.
- Graphical programming in C offers flexibility, performance, and cross-platform compatibility, making it suitable for various applications. Its importance lies in enabling developers to realise creative visions, design intuitive interfaces, analyse complex data, and develop multimedia software efficiently. Mastering graphical programming in C equips programmers with essential skills for developing graphical applications and multimedia software across various domains, enhancing their career prospects and contribution to the software industry.
- Installing and Configuring Graphics Libraries: Before starting graphical programming in C, it's crucial to install and configure necessary libraries such as SDL and Allegro, which provide tools for rendering graphics. For SDL, installation typically involves using package managers like "apt" for Ubuntu-based systems, followed by including the SDL header file and linking against the SDL library during compilation. Similar steps apply to installing Allegro, but the process may vary depending on the platform.
- Setting up Development Environment (IDEs, Compilers): Once the graphics libraries are installed, setting up a development environment is essential for writing and compiling graphical C programs. Options for IDEs include Visual Studio, Code: Blocks, Eclipse, and

CLion, providing features like project management, debugging, and syntax highlighting. Additionally, selecting a C compiler such as GCC, Clang, or Microsoft Visual C++ Compiler is necessary to translate C code into machine-readable instructions.

- Graphics programming in C finds applications in various fields such as game development, computer-aided design (CAD), data visualisation, multimedia applications, user interfaces, and animation/special effects.
- It is essential for creating visually appealing environments, manipulating 2D/3D designs, analysing and presenting data, developing multimedia software, designing intuitive interfaces, and producing animations and visual effects for media.
- Graphics programming in C offers efficient memory management, cross-platform compatibility, visually appealing applications, availability of graphics libraries, and low-level control for advanced graphics.
- These advantages enable developers to optimise performance, reach a broader audience, enhance user experience, accelerate development, and implement custom graphics solutions tailored to specific requirements.

9. TERMINAL QUESTIONS

1. How does graphical programming in C contribute to game development, and what are the key features it enables developers to implement?
2. Discuss the significance of graphical programming in C for user interface design and its impact on software usability.
3. How does graphical programming in C contribute to data visualisation, and what advantages does it offer in data analysis and decision-making?
4. How do you install the SDL (Simple DirectMedia Layer) library for graphical programming in C, and what functionalities does it provide?
5. Explain the installation process of Allegro library for graphical programming in C and its features.
6. What are the key considerations for setting up a development environment for graphical programming in C?

7. How would you set up a development environment on Ubuntu for writing graphical programs using SDL?
8. How does graphics programming in C contribute to the field of game development?
9. Explain the role of graphics programming in multimedia applications.
10. How does efficient memory management contribute to the advantages of using graphics programming in C?

10. ANSWERS TO SELF ASSESSMENT QUESTIONS

1. (a) Basic shapes used for creating complex images.
2. (c) To accurately position and size graphical elements on the screen
3. (c) Game development
4. (b) SDL (Simple DirectMedia Layer)
5. (a) Using the package manager "apt-get install libsdl2-dev."
6. (b) Visual Studio
7. (b) DirectX
8. (c) Creating graphical applications
9. (b) Initializes the graphics driver and sets the graphics mode
10. (c) Computer-Aided Design (CAD)
11. (b) Fine-grained control over memory allocation and deallocation
12. (d) Allegro

11. ANSWERS TO TERMINAL QUESTIONS

1. Graphical programming in C, facilitated by libraries like SDL and Allegro, plays a crucial role in game development by enabling the creation of visually immersive 2D and 3D environments. Developers can implement features such as rendering graphics, handling user input, animation, and multimedia integration, allowing them to bring their creative visions to life and deliver engaging gameplay experiences.
2. Graphical programming in C is essential for designing intuitive and visually appealing graphical user interfaces (GUIs) in software applications. With C-based GUI libraries, developers can create interactive interfaces with features like buttons, menus,

dialogs, and widgets, enhancing user experience and making software more accessible and user-friendly.

3. Graphical programming in C is instrumental in visualising complex data sets through charts, graphs, and diagrams, aiding in data interpretation and decision-making in scientific, engineering, and business applications. By leveraging C-based graphics algorithms, developers can represent data visually, making trends and patterns more accessible and facilitating insights extraction for informed decision-making processes.
4. SDL can be installed using package managers like "apt" for Ubuntu-based systems or "brew" for macOS. SDL offers low-level access to audio, keyboard, mouse, joystick, and graphics hardware, making it suitable for developing cross-platform multimedia applications.
5. Allegro can be installed using package managers on Linux systems. It provides support for graphics, sound, input devices, and more. Allegro is versatile and suitable for creating multimedia applications, offering functionalities beyond basic graphics rendering.
6. Setting up a development environment involves selecting an IDE (Integrated Development Environment) and a C compiler. IDE options include Visual Studio, Code: Blocks, Eclipse, and CLion, each offering different features for writing, compiling, and debugging C code. Choosing a suitable C compiler like GCC, Clang, or Microsoft Visual C++ Compiler is essential for translating C code into executable programs.
7. Set up a development environment on Ubuntu, first install SDL using the package manager. Then, choose an IDE such as Visual Studio Code and install the C/C++ extension to enable C/C++ development support. This setup allows developers to write, compile, and debug graphical C programs efficiently on the Ubuntu platform.
8. Graphics programming in C is essential for creating visually appealing environments, characters, and effects in games. C-based libraries like SDL and Allegro handle graphics rendering, input handling, and other game-related tasks, facilitating the development of both 2D and 3D games.

9. Graphics programming plays a significant role in multimedia applications such as video players, image viewers, and video editors. These applications rely on graphics to display and manipulate images, videos, and audio in various formats, enhancing the user experience and enabling multimedia content creation.
10. Efficient memory management in graphics programming allows for optimised performance in resource-intensive applications. With fine-grained control over memory allocation and deallocation, developers can reduce memory overhead and improve overall application efficiency, leading to smoother graphics rendering and better user experience.

12. REFERENCES

1. Computer Graphics: Principles and Practice in C, James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes, Addison-Wesley Professional.
2. Graphics Programming in C: A Comprehensive Resource for Every C Programmer" by Roger T. Stevens.
3. C Graphics Programming by Pradeep K. Bhatia.