OBJECTIVES:

This project presents a graphical representation of the Taj Mahal using Turbo C++ and Borland Graphics Interface (BGI). The program utilizes fundamental graphics functions such as line(), circle(), rectangle(), and ellipse() to create an artistic depiction of the monument. Advanced filling techniques like setfillstyle() and floodfill() are used to enhance the visual appeal of the structure. The project demonstrates a combination of computer graphics, mathematics, and programming logic to design a detailed architectural model. Additionally, text rendering and animation effects using settextstyle() and delay() are implemented to create an engaging user experience.

This project serves as a learning tool for students and enthusiasts interested in computer graphics programming in C++, providing insights into coordinate geometry, shape manipulation, and animation. The successful execution of the program showcases the capabilities of Turbo C++ in creating complex graphical structures, despite its limitations in modern computing environments

The objectives of the project can be outlined as follows:

- **Implement computer graphics concepts** using Turbo C++ and the BGI (Borland Graphics Interface) library.

- **Create an artistic representation of the Taj Mahal** with structured geometric shapes such as lines, rectangles, ellipses, and circles.
- **Enhancing visualization skills** by developing complex architectural structures using fundamental graphic primitives.
- **Exploring color and fill patterns** in graphics programming using setfillstyle() and floodfill().
- **Applying animation concepts** by using looping structures for dynamic text display and color transitions.
- **Integrating text and artistic elements** within the graphical output for an aesthetically appealing presentation.
- **Developing problem-solving skills** by debugging issues related to the display, rendering, and mouse interactions.

- **Implementing structured programming techniques** in C++ to improve code readability, maintainability, and modularity.
- **Gaining hands-on experience in legacy graphics programming** and understand the limitations and possibilities of early computer graphics development.

PREREQUISITES:

1. **Basic Understanding of C++ Programming** – Knowledge of data types, control structures (loops, conditions), and functions.
2. **Familiarity with Turbo C++ IDE** – Understanding how to set up, compile, and execute programs in Turbo C++.
3. **Graphics Programming in C++** – Knowledge of the Borland Graphics Interface (BGI) and functions like initgraph(), line(), rectangle(), circle(), ellipse(), setfillstyle(), and floodfill().
4. **Mathematical and Geometric Concepts** – Basic understanding of coordinates, angles, and dimensions for drawing shapes accurately.
5. **File Handling & Path Configuration** – Setting up the correct BGI path (e.g., "C:\\TurboC3\\BGI") for graphics execution.
6. **Debugging Skills** – Ability to troubleshoot errors, such as linker errors (undefined symbol issues) and display-related bugs.
7. **Understanding of Color and Fill Patterns** – Using setcolor(), setfillstyle(), and floodfill() for visual enhancements.
8. **Looping and Delays in C++** – Using loops and delay() functions for animated effects and dynamic elements.
9. **Hardware and Software Compatibility** – Ensuring Turbo C++ runs correctly on modern systems using DOSBox or similar emulators.

Libraries Used:

<graphics.h>

- Purpose: Enables graphics functions for drawing shapes, lines, colors, and text.
- Functions Used:
    - initgraph() – Initializes graphics mode.
    - setlinestyle(), setfillstyle(), setcolor() – Sets line styles, fill patterns, and colors.
    - ellipse(), rectangle(), circle(), line() – Used to draw various shapes.

- floodfill() – Fills a closed shape with a specific color.
- outtextxy() – Displays text at specified coordinates.
- closegraph() – Closes the graphics mode.

2. <conio.h>

- Purpose: Used for console input and output operations.
- Functions Used:
  - getch() – Waits for a key press before closing the program (prevents abrupt exit).

3. <dos.h> (Deprecated but used in old Turbo C++ for delays and system calls)

- Purpose: Provides functions for system delays and interrupts.
- Functions Used:
  - delay() – Pauses execution for a specified time (used in animation)

COMPUTER GRAPHICS:

Computer graphics can be a series of images which is most often called a video or single image. Computer graphics is the technology that concerns with designs and pictures on computers. That's why, computer graphics are visual representations of data shown on a monitor made on a computer. "Computer graphics is the use of a computer to define, store, manipulate, interrogate, and represent the pictorial output." An image in computer graphics is made up of a number of pixels.

Types of Computer Graphics

There are two kinds of computer graphics are–

- Interactive Computer Graphics
- Non-Interactive Computer Graphics

Interactive Computer Graphics In interactive computer graphics, users have some controls over the image, i.e., the user can make any changes to the image produced.

Interactive Computer Graphics involves computer-user two-way communication.

For Example:

- Ping-pong game.
- Drawing on touch screens.
- Display weather forecast or other moving charts/graphs on the screen.
- Animating pictures or graphics in movies.

- Graphics animation in video games

Non- Interactive Computer Graphics Non-interactive computer graphics are also known as passive computer graphics. It is a type of computer graphics in which the user has no control over the image. The photo is completely controlled by the instructions of the program, not by the user.

For Example:

- Screen savers.
- Map representation of the data.
- Graphic elements are used in the text, document, and PDF presentation.
- Static images are used in mobile applications and websites.
- Business graphics are used as brochures, business cards, menu of the hotel.

Representation of graphics

We can represent the graphics by following two ways:

1. Raster (Bitmap) Graphics
2. Vector Graphics

1. Raster Graphics: In raster graphics, the image is presented as a rectangular grid of colored squares. Raster images are also called bitmap images. Bitmap images are stored as the collection of small individual dots called pixels. Bitmap images require high resolution and anti-aliasing for a smooth appearance. For example– Paint, Photoshop, etc.

2. Vector Graphics: In vector graphics, the image is represented in the form of continuous geometric objects: line, curve, etc. Vector images are not based on pixel pattern. They use mathematical formulas to draw line and curves. The lines and curves can be combined to create an image. For Example– PowerPoint, Corel Draw, etc.

The first step in any graphics program is to initialize the graphics

drivers on the computer using initgraph method of graphics.h library.

void initgraph(int *graphicsDriver, int *graphicsMode, char

*driverDirectoryPath)

The first step in any graphics program is to initialize the graphics

drivers on the computer using initgraph method of graphics.h library.

void initgraph(int *graphicsDriver, int *graphicsMode, char *driverDirectoryPath)

The first step in any graphics program is to initialize the graphicsdrivers on the computer using initgraph method of graphics.h library.void initgraph(int *graphicsDriver, int *graphicsMode, char*driverDirectoryPath)

•graphicsDriver : It is a pointer to an integer specifying the graphics driver to be used. It tells the compiler that what graphics driver to use or to automatically detect the drive. In all our programs we will use DETECT macro of graphics.h library that instruct compiler for auto detection of graphics driver.

•graphicsMode : It is a pointer to an integer that specifies the graphics mode to be used. If *graphdriver is set to DETECT,then initgraph sets *graphmode to the highest resolution available for the detected driver.

•driverDirectoryPath : It specifies the directory path where graphics driver files (BGI files) are located. If directory path is not provided, then it will search for driver files in current working directory directory. In all our sample graphics programs, you have to change path of BGI directory accordingly where you turbo C compiler is installed.

There are 16 colors declared in C Graphics. We use colors to set the current drawing color, change the color of background, change the color of text, to color a closed shape etc. To specify a color, we can either use color constants like setcolor(RED), or their corresponding integer codes like setcolor(4).At the end of our graphics program, we have to unloads the graphics drivers and sets the screen back to text mode by calling closegraph() function.

PROJECT METHODOLOGY:

A basic overview of the project that will be made:From a distance, the Taj Mahal appears to be a large white marble structure with four tall minarets surrounding a large central dome. The overall design is symmetrical and balanced, with intricate details and carvings covering every surface. The four minarets are 3 step structures 2 on left and 2 on rightThe co-orinates chosen are precise and exact absolute values for smallscreens, and were chosen purely on the hit and trial method and was implemented using rectangle(),line(),circle() and arc() in-built functions

COMPONENTS:

Variables

- gd – Graphics driver, set to DETECT to allow the compiler to automatically detect the appropriate graphics driver.
- gm – Graphics mode, used to store the mode of the detected graphics driver.
- i – Loop variable used in the for loop for text color animation.

Functions

Graphics Initialization Functions

1. initgraph(&gd, &gm, "C:\\TurboC3\\BGI");

    - Initializes the graphics mode using the detected graphics driver.

Basic Shape Drawing Functions

2. ellipse(x, y, startAngle, endAngle, xRadius, yRadius);

    - Draws an elliptical arc with specified coordinates, start and end angles, and radii.

3. rectangle(x1, y1, x2, y2);

    - Draws a rectangle with the given top-left (x1, y1) and bottom-right (x2, y2) coordinates.

4. circle(x, y, radius);

    - Draws a circle with center (x, y) and the specified radius.

5. line(x1, y1, x2, y2);

    - Draws a straight line between the given points (x1, y1) and (x2, y2).

Fill and Color Functions

6. setcolor(color);

- Sets the current drawing color.

7. setfillstyle(pattern, color);

- Sets the fill style with a pattern and color for filling shapes.

8. floodfill(x, y, boundaryColor);

- Fills a closed area starting from (x, y) up to the specified boundary color.

9. setlinestyle(style, pattern, thickness);

- Sets the line style for drawing (e.g., solid, dashed).

Text Functions

10. settextstyle(font, direction, size);

- Sets the text style for displaying text in the graphics window.

11. outtextxy(x, y, "text");

- Displays text at the specified (x, y) coordinates.

Animation and Delay Functions

12. delay(milliseconds);

- Pauses the program for a specified number of milliseconds.

13. for(i=1; i<=115; i++) { setcolor(i); ...}

- A loop that changes the text color over time to create an animation effect.

Graphics Cleanup Functions

14. getch();

- Waits for a key press before closing the graphics window.

15. closegraph();

- Closes the graphics mode and releases resources

CODE:

```
#include <conio.h>
#include <graphics.h>
#include <dos.h>

int main()
{
   int gd = DETECT, gm, i;

   // int points[] = {300,340,330,340, 412, 430, 220, 430, 300,340};
   int points[] = {300,340,330,340, 432, 450, 200, 450, 300,340};



   initgraph(&gd, &gm, "C:\\TurboC3\\BGI");



   //left side trees
setcolor(BROWN);                    // Set the color for the tree trunk
setfillstyle(SOLID_FILL, BROWN);       // Brown for the trunk
rectangle(205, 370, 215, 410);         // Tree trunk (slightly wider for better proportions)
floodfill(206, 400, BROWN);            // Fill the trunk with brown color

setcolor(GREEN);                    // Set the color for the tree canopy
setfillstyle(SOLID_FILL, GREEN);       // Green for the canopy
circle(210, 360, 13.5);                // Tree canopy (repositioned and larger for better appearance)
```

```
floodfill(210, 360, GREEN);
delay(300);       // Fill the canopy with green color

setcolor(BROWN);                    // Set the color for the tree trunk
setfillstyle(SOLID_FILL, BROWN);
rectangle(170, 370, 180, 430);  // Second tree trunk
floodfill(175, 400, BROWN);     // Fill trunk
setcolor(GREEN);                    // Set the color for the tree canopy
setfillstyle(SOLID_FILL, GREEN);
   circle(175, 370, 15);          // Second tree canopy
   floodfill(175, 370, GREEN);     // Fill canopy
   delay(300);



   // Adding trees on the right side of the trapezium
    setcolor(BROWN);                    // Set the color for the tree trunk
setfillstyle(SOLID_FILL, BROWN);
   rectangle(405, 370, 415, 410);
floodfill(406, 400, BROWN);           // Fill the trunk with brown color

    // Tree trunk
   setcolor(GREEN);                    // Set the color for the tree canopy
setfillstyle(SOLID_FILL, GREEN);    // Fill trunk
   circle(410, 360, 13.5);          // Tree canopy
   floodfill(405, 360, GREEN);     // Fill canopy

   delay(300);
```

```
//second tree :right
   setcolor(BROWN);                    // Set the color for the tree trunk
setfillstyle(SOLID_FILL, BROWN);
rectangle(440, 370, 450, 430);  // Second tree trunk
floodfill(445, 400, BROWN);     // Fill trunk
setcolor(GREEN);                    // Set the color for the tree canopy
setfillstyle(SOLID_FILL, GREEN);
circle(445, 370, 15);           // Second tree canopy
   floodfill(445, 370, GREEN);     // Fill canopy
   delay(300);


      // Define the trapezium (where "TAJ MAHAL" is written)
   setfillstyle(SOLID_FILL, BLUE); // Blue color for the trapezium

   fillpoly(5, points); // Fill the trapezium with blue color

   // Write "TAJ MAHAL" text within the trapezium
   settextstyle(7, 0, 2); // Text style and size
   setcolor(WHITE); // Set text color to white for visibility


// Base water reflection pool
   setfillstyle(11, WHITE); // Blue for water
   rectangle(150, 320, 500, 340); // Water rectangle
   floodfill(152, 322, WHITE);
   // Greenery (bushes/grass)
```

```
    setfillstyle(10, WHITE); // Green for greenery
    rectangle(120, 320, 145, 340); // Left greenery
    floodfill(122, 322, WHITE);

    rectangle(505, 320, 530, 340); // Right greenery
    floodfill(507, 322, WHITE);



    setlinestyle(0, 0, 3);
    ellipse(230, 160, 0, 180, 30, 30); // left house top
    ellipse(390, 160, 0, 180, 30, 30); // right house top
    ellipse(353, 120, 270, 90, 15, 40); // right arc
    ellipse(269, 120, 100, 270, 17, 40); // left arc

    // Design
    // ellipse(315, 120, 0, 180, 65, 10);
    //  ellipse(327, 132, 0, 180, 60, 4);
    // ellipse(322, 140, 0, 180, 65, 5);

    rectangle(265, 75,357,80);
    delay(300);



ellipse(310, 75, 0, 180,40,10);
ellipse(310, 68, 0, 180,30,10);
ellipse(310, 60, 0, 180,20,10);
```

```
ellipse(310, 50, 0, 180,10,5);


circle(310,43,2);
circle(310,40,2);
circle(310,36,3);
rectangle(200,190,260,300);


//arc design
    ellipse(207,190,0,180,5,20);
    ellipse(217,190,0,180,5,20);
    ellipse(227,190,0,180,5,20);
    ellipse(237,190,0,180,5,20);
    ellipse(247,190,0,180,5,20);
    ellipse(257,190,0,180,3,20);
    rectangle(201,160,262,195);
    delay(300);
// Arc design 2
    ellipse(365,190,0,180,5,20);
    ellipse(375,190,0,180,5,20);
    ellipse(385,190,0,180,5,20);
    ellipse(395,190,0,180,5,20);
    ellipse(405,190,0,180,5,20);
    ellipse(415,190,0,180,5,20);


     rectangle(360,190,420,300);  // Right pillar
    rectangle(360,160,420,195); // Right pillar top//
    delay(300);
// // Window right
```

```
    rectangle(365,200,415,240);
    floodfill(367,205,WHITE);
    ellipse(390,220,0,180,25,20);
    rectangle(365,250,415,300);
    floodfill(367,255,WHITE);
    ellipse(390,270,0,180,25,20);


line(150, 200, 150, 300); // extreme left
line(200, 190, 150, 200); // left join top
line(200, 195, 150, 205); // design left join top



line(160, 215, 160, 245);  // window extreme1
line(190, 210, 190, 240);
line(160, 245, 190, 240);
line(160, 215, 190, 210);

line(160, 290, 190, 290);
line(160, 255, 160, 300); // window extreme2
line(190, 250, 190, 300);
line(160, 255, 190, 250);

delay(300);
//     //window left
    rectangle(205,200,255,240);
```

```
    floodfill(210,205,WHITE);
    ellipse(230,220,0,180,25,20);
    rectangle(205,250,255,300);
    floodfill(210,255,WHITE);
    ellipse(230,270,0,180,25,20);

    line(470,200,470,300); // extreme ryt
    line(420,190,470,200); // ryt join top
    line(420,195,470,205); // design ryt join top
    line(430,210,430,240); // window extreme1
    line(460,215,460,245);
    line(430,240,460,245);
    line(430,210,460,215);
// not this

    line(430,290,460,290);
    line(430,250,430,300); // window extreme2
    line(460,255,460,300);
    line(430,250,460,255);
    delay(300);
// gate essential
    line(285,220,285,295);
    line(335,220,335,295);
    line(285,220,305,200);
    line(335,220,315,200);
    line(305,200,315,200);
    delay(300);
```

```
// pillar 1
    ellipse(38,120,0,180,14,20);
    rectangle(20,120,55,125);
    line(25,125,25,140);
    line(50,125,50,140);

    setfillstyle(2,RED);
    rectangle(25,150,50,195);
    floodfill(27,155,WHITE);
    ellipse(37,140,180,0,23,10);
    line(15,140,60,140);

    rectangle(25,205,50,240);
    floodfill(27,215,WHITE);

    ellipse(37,195, 180,0,24,10);
    line(14,195,61,195);
    rectangle(25,250,50,300);
    floodfill(27,255,WHITE);
    ellipse(37,240,180,0,23,10);
    line(14,240,61,240);

    delay(300);
// pillar 2
    ellipse(100,140,0,180,10,15);
    rectangle(87,140,113,145);
    rectangle(85,170,115,205);
    line(90,145,90,160);
```

```
    line(110,145,110,160);
    floodfill(87,175,WHITE);
    ellipse(100,160,180,0,20,10);
    line(80,160,120,160);
    rectangle(85,215,115,250);
    floodfill(87,225,WHITE);
    ellipse(100,205,180,0,20,10);
    line(80,205,120,205);
    rectangle(85,260,115,300);
    floodfill(87,275,WHITE);
    ellipse(100,250,180,0,20,10);
    line(80,250,120,250);

    delay(300);
// pillar 3
    ellipse(513,130,0,180,10,17);
    rectangle(500,130,525,135);
    rectangle(500,160,525,195);
    floodfill(505,175,WHITE);

    line(503,135,503,150);
    line(523,135,523,150);

    ellipse(513,150,180,0,20,10);
    line(493,150,533,150);
    rectangle(500,205,525,250);
    floodfill(505,215,WHITE);
```

```
    ellipse(513,195,180,0,24,10);
    line(490,195,537,195);
    rectangle(500,260,525,300);
    floodfill(505,275,WHITE);

    ellipse(512,250,180,0,20,10);
    line(493,250,532,250);
    delay(300);
// pillar 4
 ellipse(579,120,0,180,14,20);
rectangle(563,120,595,125);

line(565,125,565,140);
line(590,125,590,140);

rectangle(565,150,595,185);
floodfill(570,175,WHITE);
ellipse(579,140,180,0,23,10);
line(556,140,603,140);

rectangle(565,195,595,240);
floodfill(567,200,WHITE);
ellipse(579,185,180,0,24,10);
line(556,185,603,185);

rectangle(565,250,595,300);
floodfill(567,252,WHITE);
ellipse(579,240,180,0,23,10);
```

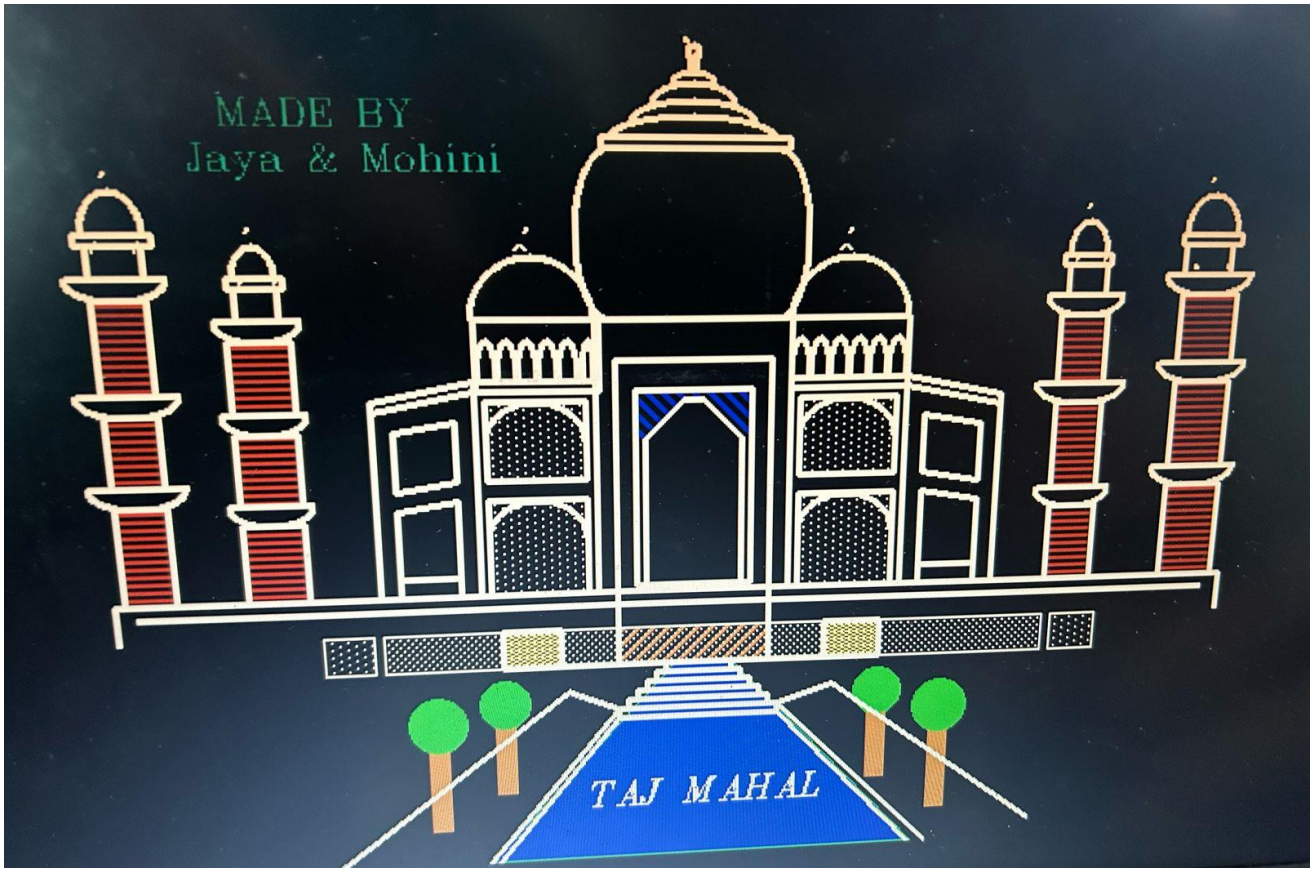
```
line(556,240,603,240);//
delay(300);
// gate
setfillstyle(5,BLUE);
rectangle(270,180,350,300);

rectangle(280,195,340,297);
floodfill(285,200,WHITE);
rectangle(260,160,360,300);

delay(300);
//base

// middle
line(20,300,600,300);

// end
line(20,300,20,320);
line(600,300,600,320);

setfillstyle(9,YELLOW);
line(30,307,590,307);
line(210,340,410,340);

rectangle(210,320,240,340);
floodfill(220,330,WHITE);

rectangle(380,320,410,340);
```

```
floodfill(385,325,WHITE);

rectangle(270,300,350,340);
delay(300);
// left foot
setfillstyle(4,BROWN);
line(244,354,120,450);
line(270,365,200,450);
line(270,370,205,450);
delay(300);
// ryt foot
line(385,355,500,440);

line(355,370,430,450);
line(355,365,440,450);
delay(300);
// middle rect
rectangle(270,365,355,370);
floodfill(285,325,WHITE);
delay(300);
// join left
line(245,355,270,365);

// join ryt
line(355,365,385,355);

// stairs
rectangle(275,365,350,360);
```

```
rectangle(280,355,345,360);
rectangle(285,350,340,355);
rectangle(290,345,335,350);
rectangle(295,340,330,345);
//

delay(300);
// text
settextstyle(7,0,2);
outtextxy(255,400,"TAJ MAHAL");
outtextxy(305,12,".");
outtextxy(570,117,".");
outtextxy(588,117,".");


outtextxy(505,125,"'");
outtextxy(520,125,"'");
outtextxy(28,117,"'");
outtextxy(47,117,"'");
outtextxy(92,135,"'");
outtextxy(108,135,"'");

settextstyle(0,0,1);
outtextxy(222,124,"^");
outtextxy(383,124,"^");
outtextxy(35,92,"'");
outtextxy(575,92,"'");
outtextxy(510,105,"'");
```

```
outtextxy(97,117,"");
outtextxy(227,117,"");
outtextxy(387,117,"");
delay(300);



settextstyle(8, 0, 1);
for (i = 1; i <= 115; i++) {
   setcolor(i);
   outtextxy(90, 50, "MADE BY");
   outtextxy(75, 70, "Jaya & Mohini");
   delay(50);
}

// ART



getch();
closegraph();
return 0;

}
```

OUTPUT:



CONCLUSION:

The Taj Mahal Graphics Project successfully demonstrates the use of Turbo C++ graphics to create a visually appealing representation of the Taj Mahal using fundamental graphics functions such as ellipse(), rectangle(), circle(), and line(). The implementation of flood fill and color styling enhances the aesthetics of the design, while text and animation add a dynamic effect to the final output.

Through this project, we have explored various graphics functions, object filling techniques, and text styling in Turbo C++. The use of loops and delays further introduces basic animation concepts, making the project more interactive.

Despite the limitations of the Turbo C++ graphics library, the project effectively showcases structured programming, coordinate-based drawing, and color management. It serves as an excellent example of how basic graphical elements can be combined to create a complex and detailed structure**.**

Overall, this project enhances problem-solving skills in computer graphics programming and provides a strong foundation for more advanced graphics-based applications.

REFERENCES:

 Programming in ANSI C

- Author: E. Balagurusamy
- Publisher: McGraw Hill Education
- Relevance: Covers C programming essentials including graphics libraries and functions.

 Computer Graphics with C

- Author: Donald Hearn and M. Pauline Baker
- Publisher: Pearson Education
- Relevance: Core concepts of drawing shapes, lines, and using graphics primitives in C.

 Turbo C Graphics Programming

- Website: https://www.tutorialspoint.com/cprogramming/c_graphics.htm
- Relevance: Examples and syntax of functions like line(), rectangle(), circle(), floodfill() etc.

 GeeksforGeeks – Graphics in C

- URL: https://www.geeksforgeeks.org/graphics-in-c/
- Relevance: Guides for beginners on using <graphics.h> in Turbo C++ and examples of geometric patterns.

 Taj Mahal Architectural References

- Wikipedia: https://en.wikipedia.org/wiki/Taj_Mahal
- Relevance: Used for understanding the structural layout and features to replicate them in graphics.