JOURNAL OF MULTIDISCIPLINARY SCIENCES AND INNOVATIONS
VOLUME 5, ISSUE 03
MONTHLY JOURNALS
ISSN NUMBER: 2751-4390
IMPACT FACTOR: 9,08

GRAPHICAL CAPABILITIES OF THE C++ PROGRAMMING LANGUAGE

Aslonov Kodir Ziyodullayevich
Asia International University

Abstract

This article analyzes the graphical capabilities of the C++ programming language and the methods for constructing complex geometric figures. Computer graphics is considered an essential component of modern information technologies, and both the theoretical and practical foundations of graphic programming are discussed. The paper examines how to initialize graphic mode in C++, draw basic geometric primitives such as points, line segments, circles, rectangles, and ellipses, and create complex images by combining these elements.

In addition, the study explores techniques for working with colors, filling styles, and generating multi-element figures using iterative operators. The creation of patterns based on trigonometric functions, as well as issues related to symmetry and geometric transformations, are also analyzed. Special attention is given to constructing complex objects using polygons, generating fractal figures, and understanding coordinate systems and mathematical foundations of computer graphics. Furthermore, the article describes methods for creating animated graphic objects, producing 3D visual effects, and generating decorative patterns using algorithmic approaches.

The materials presented in the article have practical significance for students learning graphic programming, software developers, and specialists interested in computer graphics. The results of the study can be applied in the development of graphic applications, computer games, engineering design systems, animation, and robotics.

Keywords

line, circle, c++, rectangle,graphics.h, initgraph, detect.

Computer graphics is one of the important areas of modern information technologies. With the help of graphic systems, various geometric shapes, complex figures, animations, and visual models can be created. The C++ programming language provides wide opportunities for creating graphic programs, especially in the educational process where working with graphic objects is taught through libraries such as BGI (Borland Graphics Interface), WinAPI, and OpenGL.

Complex geometric figures are composed of simple shapes — point, line segment, circle, rectangle, and ellipse. This lecture discusses methods of launching graphic mode in C++, drawing simple figures, and creating complex images by combining them.

Graphic mode is a state that allows working with images rather than text on the computer screen. In C++, graphic mode is usually implemented through the graphics.h library.

Starting graphic mode:

```
#include <graphics.h>
int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "");
   getch();
   closegraph();
   return 0;
}
```

JOURNAL OF MULTIDISCIPLINARY SCIENCES AND INNOVATIONS
VOLUME 5, ISSUE 03
MONTHLY JOURNALS
ISSN NUMBER: 2751-4390
IMPACT FACTOR: 9,08

This code opens a graphics window and keeps it until the program end:

1. Point: putpixel(x, y, color);
2. Line segment (straight line): line(x1, y1, x2, y2);
3. Rectangle: rectangle(x1, y1, x2, y2);
4. Circle: circle(x, y, radius);
5. Ellipse: ellipse(x, y, start, end, xr, yr);

Color and style control

Complex images appear clearer with the use of colors.

Setting color: setcolor(RED);
Setting background color
setbkcolor(BLUE);
Filling
setfillstyle(SOLID_FILL, GREEN);
floodfill(x, y, WHITE);
Principle of forming complex figures
A complex figure is a combination of several simple figures.
For example:
House → rectangle + triangle + door + window
Car → rectangle + circle + ellipse
Robot → several geometric shapes
Drawing a house shape

```
rectangle(200,200,400,350);   // Body of the house
line(200,200,300,120);        // Roof
line(300,120,400,200);
rectangle(270,270,330,350);   // Door
circle(240,250,20);           // Window
circle(360,250,20);
```

Here the complex figure is built from simple elements.
Car image

```
rectangle(150,250,450,320);   // Body
circle(220,330,30);           // Wheel
circle(380,330,30);
rectangle(260,210,350,250);   // Cabin
```

Increasing complexity using repetition operators
Loops allow creating figures with many elements.
Example: Concentric circles

```
for(int r = 20; r <= 120; r += 20)
   circle(300, 200, r);
```

Figures based on mathematical formulas
Complex patterns can be generated using trigonometric functions.
Spiral

```
for(float a=0; a<20; a+=0.1) {
   int x = 300 + a*10*cos(a);
```

```
    int y = 200 + a*10*sin(a);
    putpixel(x, y, WHITE);
}
```

Symmetry and transformations

Complex figures are obtained through reflection, translation, and rotation.

Example: Snowflake-like figure

```
for(int i=0;i<360;i+=30){
    line(300,200,
         300+100*cos(i*3.14/180),
         200+100*sin(i*3.14/180));
}
```

Practical applications

Computer games

Engineering design systems

Architectural models

Robotics

Animation and design

Creating complex figures using polygons

Polygons play an important role in creating complex objects in C++ graphics programming. A polygon is a closed figure formed by connecting three or more points sequentially.

Polygon function

```
int points[] = {200,200, 300,150, 400,200, 350,300, 250,300};
drawpoly(5, points);
```

This code draws a pentagon.

Filled polygon

```
setfillstyle(SOLID_FILL, YELLOW);
fillpoly(5, points);
```

Using polygons, it is possible to create:

stars

irregular shapes

maps

complex design elements

Fractal figures

Fractals are complex geometric shapes characterized by self-similarity. They are drawn using mathematical formulas and recursion.

Simple recursive tree

```
void tree(int x, int y, int len, int angle) {
    if(len < 5) return;
    int x2 = x + len * cos(angle * 3.14 / 180);
    int y2 = y - len * sin(angle * 3.14 / 180);
    line(x, y, x2, y2);
    tree(x2, y2, len-10, angle-20);
    tree(x2, y2, len-10, angle+20);
}
```

Applications of fractals:

modeling natural objects

computer graphics landscapes

animation and games

Coordinate system and mathematical foundations
Each point on the graphic screen is determined by coordinates.
(0,0) — top-left corner
X axis — to the right
Y axis — downward
Complex curves are drawn using mathematical formulas.
Sine wave

```
for(int x=0; x<640; x++) {
   int y = 240 + 100 * sin(x * 3.14 / 180);
   putpixel(x, y, WHITE);
}
```

Animated complex figures
Moving figures are an important part of graphics programming.
Simple moving circle

```
for(int i=0; i<400; i++) {
   cleardevice();
   circle(50+i, 200, 30);
   delay(20);
}
```

Animation is used in:
games
simulations
robotics interfaces
Methods for creating a 3D effect
Although BGI is 2D graphics, a 3D effect can be achieved using perspective.
Cube image

```
rectangle(200,200,300,300);
rectangle(230,170,330,270);
line(200,200,230,170);
line(300,200,330,170);
line(300,300,330,270);
line(200,300,230,270);
```

This produces a 3D view of a cube.
Complex patterns and ornaments

Decorative patterns are generated using repetitive algorithms.
Flower shape

```
for(int i=0;i<360;i+=10){
   int x = 300 + 100*cos(i*3.14/180);
   int y = 200 + 100*sin(i*3.14/180);
   circle(x,y,20);
}
```

Relation to modern graphic libraries
BGI is convenient for educational purposes, but real projects use:
OpenGL
DirectX
SDL
SFML

Qt
All of them rely on basic geometric primitives.

References

1. Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
2. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
3. Russell, S., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach. Pearson.
4. Vapnik, V. N. (1998). Statistical Learning Theory. Wiley.
5. Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective. MIT Press.