VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

Department of Computer Engineering

Computer Graphics Lab(CGVR)
Mini Project Report on

Creation of an Animated Cartoon Character

Submitted in partial fulfilment of the requirements of Second Year Computer Engineering

By
Arnav Bagchi(D7C-04)
Jay Dulera(D7C-15)
Rohan Ghosalkar(D7C-20)

Supervisor: Mrs. Suvarna Bhatsangave

DEPARTMENT OF COMPUTER ENGINEERING
V.E.S INSTITUTE OF TECHNOLOGY
2019 - 2020

INDEX

ACKNOWLEDGEMENT

We would like to thank our Project Supervisor for the Computer Graphics Lab(CG) Mrs. Suvarna Bhatsangave for giving us the opportunity to present our idea in our Second Year. We have referred to the notes provided by her in CGVR along with using the information provided on the Internet.We have tried to put the essence of the mini project which focuses on our chosen idea by providing step by step procedure for implementation and explaining the working of the project in brief.

ABSTRACT

In our project, we want to create an animated character using the graphics function in C. For this we will be using different graphics functions along with drawing various shapes, using different colours and using the Flood Fill Algorithm to fill colour in shapes. The output of the project should be the animation which could make certain movements or hand gestures.

OBJECTIVE

To make use of graphics.h library and its basic functions to create a real time animated pixelated cartoon. We shall make sure we employ the following functions:

1. Use the graphics.h and conio.h header file
2. Use different colour schemes
3. Use various shapes and lines and learn their syntax
4. Use Flood Fill Algorithm to fill colour in the shapes

INTRODUCTION

Graphics in C

The first step in any graphics program is to include *graphics.h* header file. The *graphics.h* header file provides access to a simple graphics library that makes it possible to draw lines, rectangles, ovals, arcs, polygons, images, and strings on a graphical window.
The second step is initializing the graphics drivers on the computer using the *initgraph* method of graphics.h library.

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

This initializes the graphics system by loading the passed graphics driver then changing the system into graphics mode. It also resets or initializes all graphics settings like color, palette, current position etc, to their default values. Below is the description of input parameters of initgraph function.

- **graphicsDriver** : It is a pointer to an integer specifying the graphics driver to be used. It tells the compiler what graphics driver to use or to automatically detect the drive. In all our programs we will use the DETECT

macro of graphics.h library that instructs the compiler for auto detection of graphics drivers.

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

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

Colors in C Graphics Programming

There are 16 colors declared in the graphics.h header file. 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 (Foreground and Background Color). To specify a color, we can either use color constants like setcolor(RED), or their corresponding integer codes like setcolor(4). Below is the color code in increasing order.

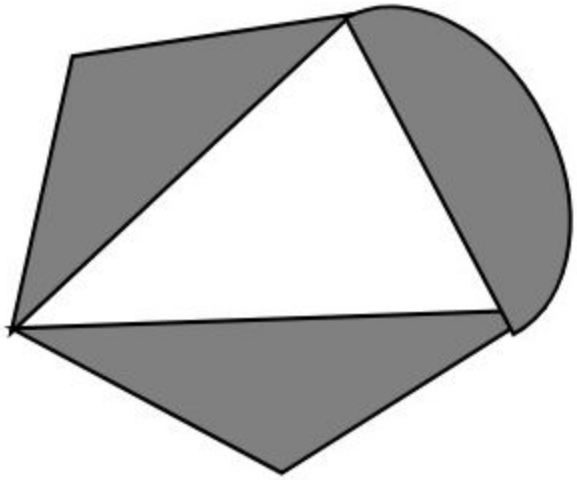
COLOR	VALUE
BLACK	0
BLUE	1
GREEN	2
CYAN	3
RED	4

MAGENTA	5
BROWN	6
LIGHT GRAY	7
DARKGRAY	8
LIGHTBLUE	9
LIGHT GREEN	10
LIGHTCYAN	11
LIGHT RED	12
LIGHTMAGENTA	13
YELLOW	14
WHITE	15
BLINK	128

Flood Fill Algorithm

In this method, a point or seed which is inside the region is selected. This point is called a seed point. Then four connected approaches or eight connected approaches are used to fill with specified color.

The flood fill algorithm has many characters similar to boundary fill. But this method is more suitable for filling multiple colors. When the boundary is of many colors and interior is to be filled with one color we use this algorithm.



In the fill algorithm, we start from a specified interior point (x, y) and reassign all pixel values that are currently set to a given interior color with the desired color. Using either a 4-connected or 8-connected approach, we then step through pixel positions until all interior points have been repainted.

Algorithm for Flood Fill:

```
1.  Procedure floodfill (x, y,fill_ color, old_color: integer)  {
2.      If (getpixel (x, y)=old_color)
3.     {
4.      setpixel (x, y, fill_color);
5.      fill (x+1, y, fill_color, old_color);
6.       fill (x-1, y, fill_color, old_color);
7.      fill (x, y+1, fill_color, old_color);
8.      fill (x, y-1, fill_color, old_color);
9.       }
10.}
```

HARDWARE AND SOFTWARE REQUIREMENTS

This project is put into operation in C language. A simple C compiler such as Turbo C or CodeBlocks can be used to execute the code. However it is important that the *graphics.h* library be included as we call functions from this Library. It is extremely simple to include this library in CodeBlocks however Turbo C has this library pre included. The hardware requirements are an Intel 386 or higher processor with a minimum of 4mb RAM with at least 25 mb disk space with a supporting operating system.

IMPLEMENTATION

CODE:

```
#include<graphics.h>
#include<dos.h>
#include<conio.h>
#include<alloc.h>
void  *buf;
void firstleft();
void secondleft();
void main()
{
int gd=DETECT,gm,i=0,x,y,area;
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
rectangle(0,0,getmaxx(),getmaxy());
arc(240,120,40,140,70);
ellipse(165,80,10,280,20,20);
ellipse(315,80,-100,170,20,20);
arc(235,120,163,215,70);
```

```
arc(245,120,-35,17,70);
ellipse(193,178,85,280,40,20);
ellipse(283,178,-100,95,40,20);
ellipse(238,199,180,0,39,50);
ellipse(213,123,44,240,33,40);
ellipse(262,123,-60,135,33,40);
ellipse(210,123,0,360,13,20);//left eye
ellipse(265,123,0,360,13,20);//right eye
ellipse(210,133,0,360,10,10);//left eye ball
ellipse(265,133,0,360,10,10);//right eye ball
ellipse(210,133,0,360,3,3);//left eye ball
ellipse(265,133,0,360,3,3);//right eye ball
ellipse(238,160,0,360,10,13);//nose
arc(240,125,228,312,68);//mouth
arc(240,120,230,310,72);//mouth
setfillstyle(1,4);
floodfill(238,160,15);//nose
setfillstyle(1,15);
floodfill(210,113,15);
floodfill(265,113,15);
setfillstyle(1,7);
floodfill(210,100,15);
setfillstyle(1,0);
floodfill(315,80,15);
moveto(203,220);
lineto(203,260);
lineto(183,260);
lineto(183,350);
lineto(293,350);
lineto(293,260);
lineto(273,260);
lineto(273,220);
moveto(183,350);
```

```
lineto(173,460);
lineto(213,460);
lineto(238,400);
lineto(263,460);
lineto(303,460);
lineto(293,350);
moveto(173,460);
lineto(143,478);
lineto(213,478);
lineto(213,460);
moveto(263,460);
lineto(263,478);
lineto(333,478);
lineto(303,460);
line(238,400,238,350);
//right hand
moveto(183,260);
lineto(113,310);
lineto(183,375);
moveto(183,280);
lineto(137,310);
lineto(181,353);
setfillstyle(2,0);
floodfill(190,300,15);
setfillstyle(1,4);
floodfill(223,400,15);
setfillstyle(1,4);
floodfill(253,400,15);
setfillstyle(1,14);
floodfill(173,470,15);
floodfill(303,470,15);
//fingers
secondleft();
```

```
ellipse(413.5,228,0,180,3.5,3.5);
line(420,240,433,240);
line(423,247,440,247);
line(413,240,410,228);
line(417,228,420,240);
ellipse(433,243.5,-90,90,3.5,3.5);
line(423,254,440,254);
ellipse(440,250.5,-90,90,3.5,3.5);
ellipse(430,257,-90,90,3,3);
line(413,260,430,260);
area=imagesize(409,224,444,261);
buf=malloc(area);
getimage(409,224,444,261,buf);
while(!kbhit())
{
if(i==0)
{
setfillstyle(1,15);
setcolor(15);
ellipse(210,133,0,360,10,10);//left eye ball
ellipse(265,133,0,360,10,10);//right eye ball
setcolor(0);
ellipse(210,133,0,360,3,3);//left eye ball
ellipse(265,133,0,360,3,3);//right eye ball
floodfill(210,133,15);
floodfill(265,133,15);
setcolor(0);
putimage(391,209,buf,1);
firstleft();
setcolor(15);
secondleft();
putimage(409,224,buf,0);
i=1;
```

```
}
else
{
setfillstyle(1,0);
setcolor(0);
ellipse(210,133,0,360,10,10);//left eye ball
ellipse(265,133,0,360,10,10);//right eye ball
floodfill(210,133,0);
floodfill(265,133,0);
setcolor(15);
ellipse(210,133,0,360,3,3);//left eye ball
ellipse(265,133,0,360,3,3);//right eye ball
floodfill(210,133,15);
floodfill(265,133,15);
setcolor(0);
putimage(409,224,buf,1);
secondleft();
setcolor(15);
firstleft();
putimage(391,209,buf,0);
i=0;
}
delay(300);
}
getch();
}
void firstleft()
{
moveto(293,260);
lineto(353,276);
lineto(395,223);
moveto(293,280);
lineto(355,296);
```

```
lineto(395,245);
}
void secondleft(){
moveto(293,260);
lineto(363,280);
lineto(413,240);
moveto(293,280);
lineto(363,300);
lineto(413,260);
}
```

RESULT


DOSBox 0.74, Cpu speed: max 100% cycles, Fra...

CONCLUSION

As we can see, different functions from the graphics.h library have been used to implement this project and the output is an animation of a cartoon character which seems to be waving his hand and moving his eyes. The colour of the animation and background is customisable as well as the size and movement. It is an effortless implementation to showcase the creation of a basic Computer Graphics project using the graphics.h library and its different functions. By including delays and more characters we can make the animation into a short animated video.

REFERENCES

We have referred to the notes provided to us by our supervisor along with using books by R.K. Maurya and Tata McGraw Hill. Along with this we have also referred to online sites.
Here are some of the references:

- https://tutorialspoint.dev/computer-science/computer-graphics/setfillstyle-floodfill-c
- https://web.stanford.edu/class/archive/cs/cs106b/cs106b.1126/materials/cppdoc/graphics.html
- https://stackoverflow.com/questions/15096609/c-compiler-for-ms-dos
- https://www.programmingsimplified.com/c/graphics.h