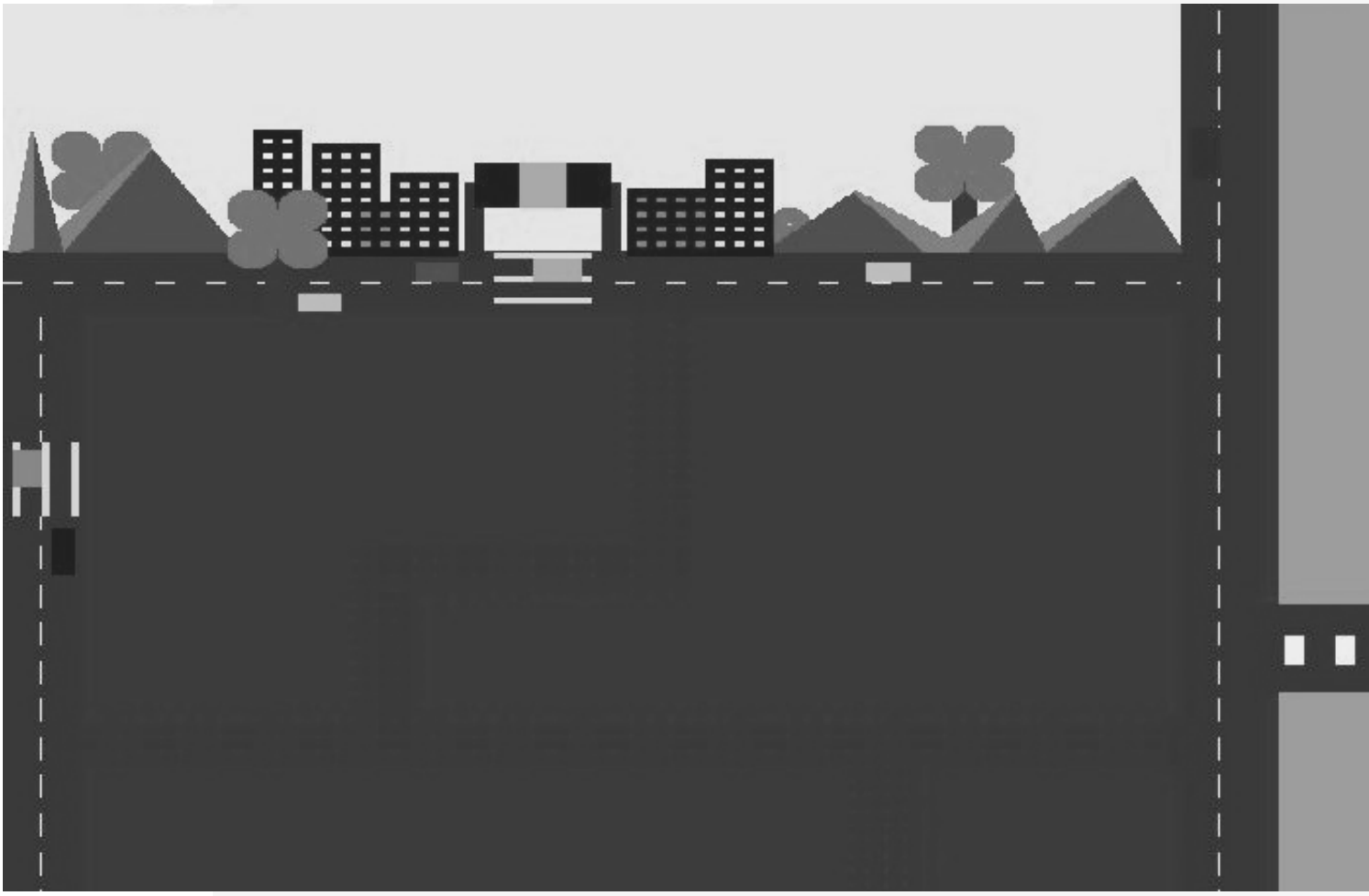
NATIONAL INSTITUTE OF TECHNOLOGY, SURATHKAL

PROJECT REPORT ON

CITY SIMULATION

Submitted to : Mrs Sangeetha Harikranta



PREPARED AND PRESENTED BY

ARJUN KUMAR(16IT108)
UTTAM KUMAR(16IT250)
SULABH TEMBHURKAR(16IT245)
TARUN PRATAP SINGH(16IT143)

Acknowledgement

*We would like to express our special thanks of gratitude to our teacher (**Mrs Sangeeta Harikranta**) as well as our Teammates who gave me the golden opportunity to do this wonderful project on the topic (**Siimulation Of a City 2D**), which also helped us in doing a lot of Research and we came to know about so many new things .we are really thankful to them. Secondly we would also like to thank our friends who helped me a lot in finalizing this project within the limited time frame.*

Abstract

Computer graphics has vast applications, one of them is creating the 2D model of objects This Project is a game built in C++ with OpenGl Libraries . Its built with basic 2d primitive Shapes and has an isometric perspective.City crowded with vehicles,and simulated traffic. This projects simulates the 2D art of a city. The city is big so only a small portion has being created in the form of various 2D objects using computer graphics. We can clearly see the 2D objects in the image below - the buildings, trees, flagpole, roads and small houses. These objects together comes to form a real like society.

Table Of Content

1. Introduction

Today there are very few aspects of our lives not affected by computers. Practically every cash or monetary transaction that takes place daily involves a computer. In many cases, the same is true of computer graphics. Whether you see them on television, in newspapers, in weather reports or while at the doctor's surgery, computer images are all around you.

"A picture is worth a thousand words" is a well-known saying and highlights the advantages and benefits of the visual presentation of our data.A well-chosen graph is able to transform a complex table of numbers into meaningful results. A range of tools and facilities are available to enable users to visualize their data, and this document provides a brief summary and overview.

The Particular Simulation Developed here is just a small piece demonstrate the power of Computer Graphics and its vast application across all fields.

For line drawing – ***DDA line algorithm***
For circle drawing – ***mid-point circle algorithm***
For color filling – ***Scanline color fill algorithm***
For plotting points – ***OpenGL inbuilt function***

All this algorithms have been combined to make some primitive shapes as Rectangle and triangles.Whole project is built using this two primitive shapes from the scratch.

2. Requirement Analysis

This Particular c++ application doesn't requires any high funcionality computers.Its just built in an environment having Opengl and glut installed on it.

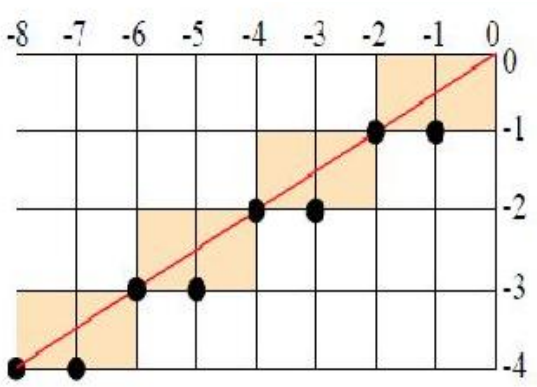
Operating Systems : - Windows /Linux having Opengl/Glut installed on it.

GPU :- If a system has a good GPU then the user might Observe a significant improve in the FPS .

3. Methods Used (Functions Implemented)

3.1. DDA Line Drawing Algorithm

Given coordinate of two points A(x1, y1) and B(x2, y2) such that x1 < x2 and y1 < y2. The task to find all the intermediate points required for drawing line AB on the computer screen of pixels. Note that every pixel has integer coordinates.
In DDA algorithm we do following :
Find smaller of dy or dx. Also slope is calculated as m=(y2-y1)/(x2-x1).
If (0<=m<=1), then $y_{k+1} = y_k + m$ *and* $x_{k+1} = x_k + 1$.
If (m>1), then $x_{k+1} = x_k + 1/m$ *and* $y_{k+1} = y_k + 1$.
Similar calculations are carried out to determine pixel positions for negative slope. Here, x_{k+1} *and* y_{k+1} *are pixel positions for next coordinate.*



3.1.1. Function:

```
double dx=(X2-X1);
double dy=(Y2-Y1);
double steps;
float xInc,yInc,x=X1,y=Y1;
glPointSize(size);
steps=(abs(dx)>abs(dy))?(abs(dx)):(abs(dy));
 xInc=dx/(float)steps;
 yInc=dy/(float)steps;
glBegin(GL_POINTS);
glVertex2d(x,y);
  int k;
for(k=0;k<steps;k++)
{
        x+=xInc;
        y+=yInc;
       glVertex2d(round_value(x), round_value(y));
}
glEnd();
glFlush();
```

3.2. Mid Point Circle Drawing

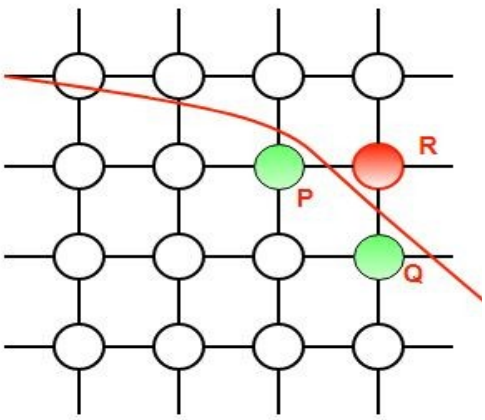
For any given pixel (x, y), the next pixel to be plotted is either **(x, y+1)** or **(x-1, y+1)**. This can be decided by following the steps below.

- ➔ Find the mid-poin **|p|** of the two possible pixels i.e (x-0.5, y+1)
- ➔ If **|p|** lies inside or on the circle perimeter, we plot the pixel (x, y+1), otherwise if it's outside we plot the pixel (x-1, y+1)

Boundary Condition : Whether the mid-point lies inside or outside the circle can be decided by using the formula:-

Given a circle centered at (0,0) and radius r and a point p(x,y)

$$F(p) = x2 + y2 - r2$$



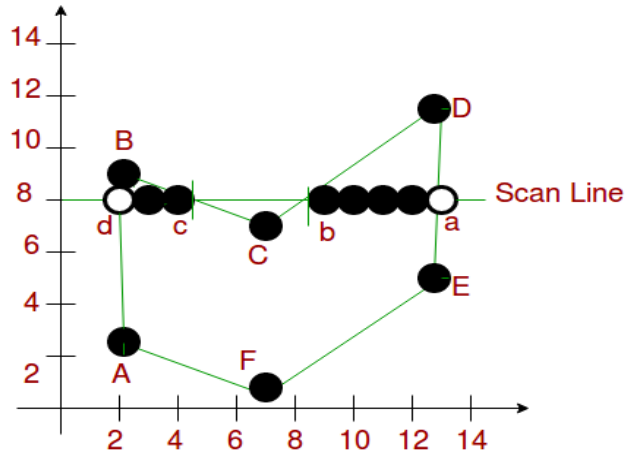
- ➔ if F(p)<0, the point is inside the circle
- ➔ F(p)=0, the point is on the perimeter
- ➔ F(p)>0, the point is outside the circle

3.2.1. Function:

```
void DrawCircle(float pntX1,float pntY1,float r,float size)
  {
    float x = 0;
    float y = r;
    float decision = 5/4 - r;
    plot(x, y,pntX1,pntY1,size);
    while (y > x)
     {
        if (decision < 0){x++; decision += 2*x+1;}
       else{y--;x++;decision += 2*(x-y)+1;}
        plot(x, y,pntX1,pntY1,size);
        plot(x, -y,pntX1,pntY1,size);
        plot(-x, y,pntX1,pntY1,size);
       plot(-x, -y,pntX1,pntY1,size);
       plot(y, x,pntX1,pntY1,size);
       plot(-y, x,pntX1,pntY1,size);
        plot(y, -x,pntX1,pntY1,size);
        plot(-y, -x,pntX1,pntY1,size);
                                  }
     }
```

3.3 Scanline Polygon Fill Algorithm

Scanline filling is basically filling up of polygons using horizontal lines or scanlines. The purpose of the SLPF algorithm is to fill (color) the interior pixels of a polygon given only the vertices of the figure. To understand Scanline, think of the image being drawn by a single pen starting from bottom left, continuing to the right, plotting only points where there is a point present in the image, and when the line is complete, start from the next line and continue.
This algorithm works by intersecting scanline with polygon edges and fills the polygon between pairs of intersections.



3.3.1. Function:

```
void scanfill(float x1,float y1,float x2,float y2,float x3,float y3,float x4,float y4,float size)
{
   int le[6000],re[6000];
   int i,y;
   for(i=0;i<6000;i++)   // initialize le and re array values
    {
            le[i]=6000;
             re[i]=0;
     }
    edgedetect(x1,y1,x2,y2,le,re);
    edgedetect(x2,y2,x3,y3,le,re);
    edgedetect(x3,y3,x4,y4,le,re);
     edgedetect(x4,y4,x1,y1,le,re);
for(y=0;y<6000;y++)
{
          if(le[y]<=re[y])
                for(i=le[y]+1;i<re[y];i++)
                          drawpixel(i,y,size);
}
}
```

```
void edgedetect(float x1,float y1,float x2,float y2,int *le,int *re)
{
 float mx,x,temp;
 int i;
  if((y2-y1)<0)
          {
         temp=x1;x1=x2;x2=temp;
          temp=y1;y1=y2;y2=temp;
          }
          if((y2-y1)!=0)
                  mx=(x2-x1)/(y2-y1);
         else
                   mx=x2-x1;
          x=x1;
          for(i=y1;i<y2;i++)
          {
                   if(x<le[i])
                            le[i]=x;
                  if(x>re[i])
                           re[i]=x;
                  x+=mx;
          }
}
```

4. Results & Discussion

This Project is written in c++ with added dependencies of opengl/Glut.The objects made in this Project are made completely from the scratch using the supporting functions and algorithms. This simulation is basically a game and is a good example of interactiveomputer graphics.
A lot more things are to be added in it in future and will be optimised as per knowledge of mine in this field increases.
The projectt has been developed on the windows platform and in CODEBLOCKS IDE,along with added dependencies.
To understand the depth of this parrticular topic i have avoided the use of inbuilt functions as much as possible,the functions for filling colors and drawing lines ,circles and other primitive shapes have been built using certain algorithms defined in this field.Its built with basic 2d primitive Shapes and has an isometric perspective.City crowded with vehicles,and simulated traffic. This projects simulates the 2D art of a city. The city is big so only a small portion has being created in the form of various 2D objects using computer graphics. We can clearly see the 2D objects in the image below - the buildings, trees, flagpole, roads and small houses. These objects together comes to form a real society. like .
Using the mentioned algorithms we were finally able to develop a working interactive simulation of a small city.The Functions used to built the Environment of the City are as follows:

```
void DrawLine(float,float,float,float,float);
void DrawTriangle(float,float,float,float,float,float);
void plot(float , float ,float ,float, float );
void pavement(float ,float ,float ,float );
void DrawRoad();
void glRoad(float,float,float,float);
void DrawZebraCrossingV(float,float,float,float);
void DrawZebraCrossingH(float,float,float,float);
void DrawCar();
void RandomCar();
void DrawRandomCar(int,bool,int,int,int[],int[],float);
void BackGround();
void Buildings(float,float,float,float);
void Mountains();
void tree(int,int,float,int);
void EnvironMent1();
void DrawBoat();
void DrawTriangleFan(float,float,float );
void Sun(float,float,float);
void Cloud(float,float,float);
void DrawSun();
void DrawCloud();
void Translate(int,int);
void Helicopter(float,float,float,float,float);
void DrawHeli();
```

4.1. Steps Followed to build the project mentioned above:

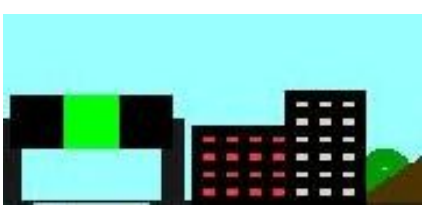
- ➔ *Every Other Object in this project has been built using lines.So after setting the ViewPorts and the background Colors,we drew the lines between appropriate edges in order to obtain the required figures.*
- ➔ *We have implemented Scanline while Drawing figures to fill colors in them.Somewhere for designing purposes we have drawn save figures multiple times to obtain a pattern.*
- ➔ *After drawing a static figure ,we added motion to it using self written functions and translating them.The speed of the motion **(FPS)** has been managed by a Timer_Callback function.*
- ➔ *After adding motion to the objects we had to draw a car which has interactive features.So we has functionality to the object and simulated it with the user input keys.*
- ➔ *In order to make the motion look more realistic we have added a relative motion to the viewport, when the car moves the viewport moves in the opposite direction.*
- ➔ *In order to look that city is moving continuosly we have added the objects in such a way that it appears to be randomized and objects appear continuosly.*
- ➔ *Even Simulating the Traffic was a bit hard nut to crack,but simulation worked perfetly and the cars vehicles moving stop in the red signal and move in the green signal.*
- ➔ *After reading these lines refer the code for the best understanding of this project.*

4.2. Results Obtained:

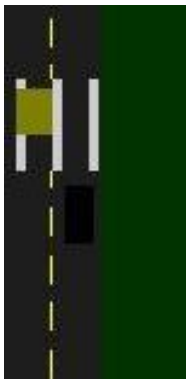
4.2.1. Trees and Mountains:

4.2.2. Clouds and Sun:

4.2.3. Buildings and Signal:



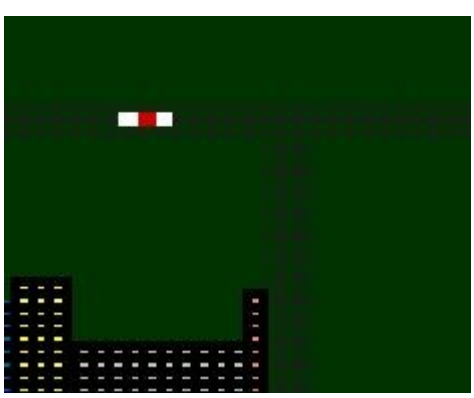
4.2.4. Cars and Road:



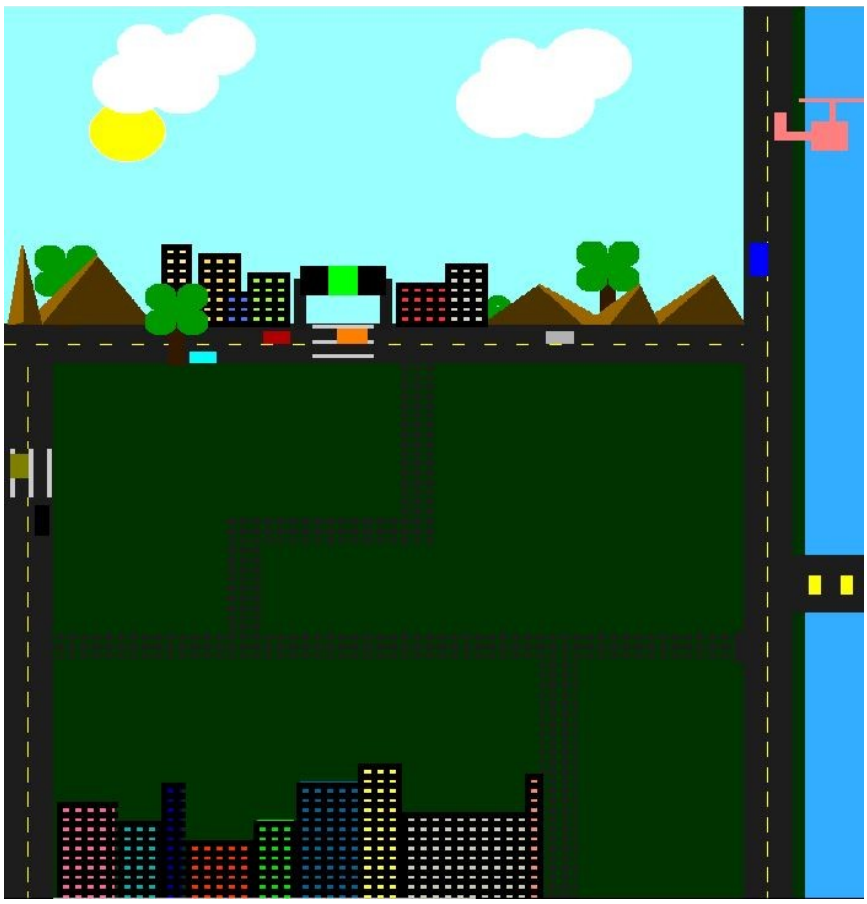
4.2.5. Helicopter:

4.2.6. Boats:

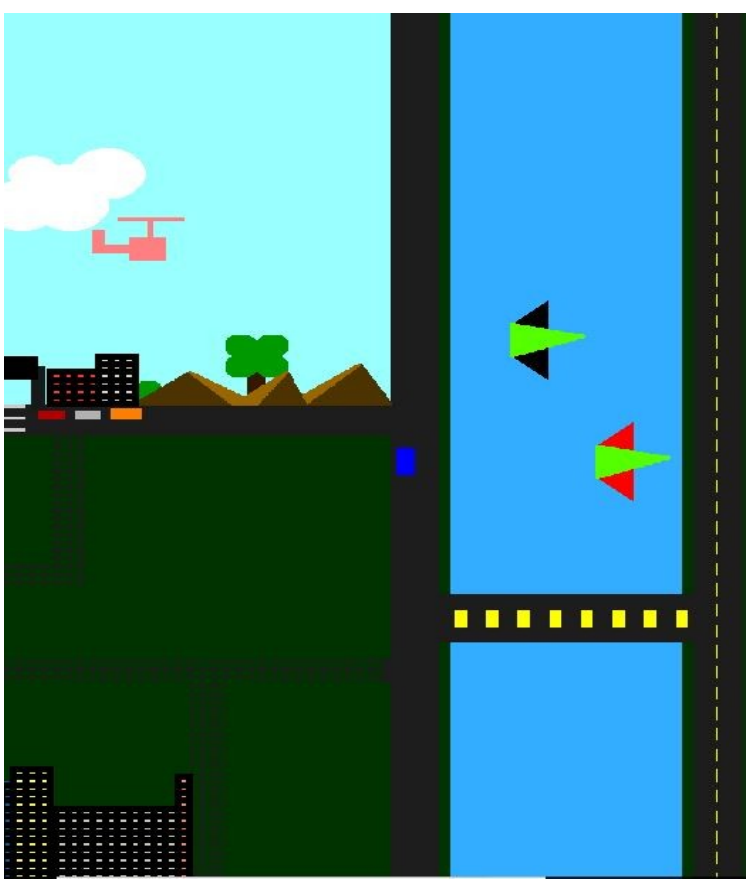
4.2.7. Interactive Car:



4.2.8. ViewPort 1:



4.2.9. ViewPort 2:



5. Conclusion

The Project can be useful at a large scale if developed in an optimised state.It can be used to study the detailed map of the city and can be used to manage traffic .As such simulations would give a proper idea of the faults in the infrastructure and and can be usedto manage resiources in a most beneficial way.
This project can also be seen as a traffic simulation becoz it has added functions for vehicles stopping for an interval and again continuing their motion.Hence if developed professionally it would prove to be helpful to the society.

6. References

1.) ***https://www.geeksforgeeks.org/***
2.) ***https://www.w3schools.com/colors/***
3.) ***https://www.opengl.org/***
4.) ***https://www.stackoverflow.com/***