♦ACKNOWLEDGEMENTS♦

First and foremost, we would like to thank God Almighty for bringing this opportunity in our lives to create this project.

We express our sincere gratitude to Prof. M. M. Bhajibhakre who motivated us to develop this project.

We also thank her for all the support that she has provided in order to bring about our creative and technical abilities through this project.

We thank our parents for being our continuous source of encouragement.

Title	To create an animation clip using the graphics library in C++
Aim / Problem Statement	To depict vehicle locomotion as an animation clip which follows the traffic signal constraints
CO Mapped	**CO4:** Define the concepts of windowing and clipping and apply various algorithms to generate polygons. **CO5:** Apply the logic to implement, curves, fractals, animation or gaming programs
Pre –requisite	1. Basic programming skills of C++ 2. 64-bit Open source Linux 3. Open Source C++ Programming tool like G++/GCC 4. Basic knowledge about the definitions included in the graphics library
Learning Objective	To understand the concepts of animation using computer graphics

Theory:

The Advent of Animation:

Hundreds of years before the introduction of true graphics and animation, people enjoyed moving figures that were created and manipulated manually in puppetry, automata, shadow play and the magic lantern.

Cinematography broke through in 1895 and the enormous success of Mickey Mouse is seen as the start of the golden age of **American animation**.

With introduction of computers by **Charles Babbage** and contributions in **graphical user interfaces, graphics processing units, computer graphics, animation softwares, special fantasy effects** and **simulations** became possible.

- Animation, simulation and other virtual real world experiences are now possible using some recent graphics softwares.
- They can create prototypes, models, study graphs and testing environments with reusability to create a working model as per application.
- Real world experiences are now possible from any corner of this world.

ANIMATION:

- Animation covers any change of appearance of any visual effect that is time based.
- The animation defines a mapping of the time to values for the target attribute.
- It includes change of **position, transparency, time varying changes in shape & even changes of the rendering techniques**.
- **Animation classifies into 2 types-**

a) Frame animation which is non-interactive
b) Sprite animation which is interactive

In traditional animation, images are drawn or painted by hand on transparent celluloid sheets to be photographed and exhibited on film. Today, most animations are made with **computer-generated imagery (CGI).**

2D computer animation can be used for stylistic reasons, low bandwidth or faster real-time renderings.

Commonly the effect of animation is achieved by a rapid succession of sequential images that minimally differ from each other.

♦The Scope of this mini project♦

This mini project can implement an animation clip of the locomotion of a car using the graphics library which is cost effective for studying the behaviour of the car in motion in conjuction with obedience to the traffic signal.

In order to follow traffic rules the car stops at the traffic signal if it is red and continues it's motion on perceiving the green signal.

Music can be played by the driver.

The rotation of the car wheels indicates the motion of the car.

Appropriate messages are displayed on the screen to enlighten the viewer.

Car will stop when it's fuel tank is empty.

There is extensive use of Computer graphics concepts with primary C++ programming skills in this animation.

♦ The Cause of Locomotion ♦

Vehicle:

It is a mobile machine that transports people or cargo. Typical vehicles include bicycles, motor vehicles (motorcycles, cars, trucks, buses), railed vehicles (trains, trams), watercraft (ships, boats), aircraft and spacecraft. Land vehicles are classified broadly by the principle used to apply steering and driving forces against the ground: wheeled, tracked, railed or skied.

The locomotive principle:

Movement from one place to another is defined as the ability to locomote. In the human body, the locomotive systems permit locomotion and consist of bones that are the framework of the skeleton, joints hold the bones together and the muscles contract and relax for movement.

In this program, we will display the **road** and draw a **car** and color it. Then we will generate the **traffic signal** and the supporting **background**. In every iteration by the while loop we keep on **incrementing the x** coordinate of every point of car at the same vertical distance ie**. y coordinate remains** constant at the level of the road to perceive the car moving from left to right (forward) along the length of the road.

Similarly, for the same y value the traffic signal and the background objects will move right to left (backward) to realise the effect of relative motion.

The car is enabled to stop before the signal and when it's fuel tank is empty as per the manipulation of the value of the iterator variable used in the while loop.

: We can use the following graphics functions in this program :

Functions of Graphics Library used in the program

Function	Description
initgraph	It initializes the graphics system by loading the passed graphics driver then changing the system into graphics mode.
getmaxx	It returns the maximum X coordinate in current graphics mode and driver.
getmaxy	It returns the maximum X coordinate in current graphics mode and driver.
setcolor	It changes the current drawing colour. Default colour is white. Each color is assigned a number, like BLACK is 0 and RED is 4. Here we are using colour constants defined inside graphics.h header file.
setfillstyle	It sets the current fill pattern and fill color.
circle	It draws a circle with radius r and centre at (x, y).
line	It draws a straight line between two points on screen.
arc	It draws a circular arc from start angle till end angle.
cleardevice	It clears the screen, and sets current position to (0, 0).

Function	Description
floodfill	It is used to fill a closed area with current fill pattern and fill color. It takes any point inside closed area and color of the boundary as input.
delay	It is used to suspend execution of a program for a M milliseconds.
closegraph	It unloads the graphics drivers and sets the screen back to text mode.

: Conclusion :

Thus, the implementation of the locomotion of an object is possible by means of the concepts used in computer graphics animation.

PROGRAM CODE

```
// header files or C++ standard libraries ie. preprocessor derectives

#include<iostream>        //for standard input/output in C++
#include<graphics.h>     //to use the graphics library readymade functions
#include<dos.h>    /*dos.h is a header file of C Language. This library has
                    functions that are used for handling interrupts, producing
                    sound, delay, date and time functions, etc.*/

int main()
{                                  //j is used in pieslice, i is iterator in while loop
int i=0,j=0,page=0;                //page variable is used for double buffering
initwindow(2000,800);   /* defines the size of graphics output window ie.
                                   Initialise window to (width,height) */

settextstyle( DEFAULT_FONT,HORIZ_DIR , 2 );     /* textstyle set to default
   font, text is displayed horizontally ie. left to right with font size=2
       ie. void settextstyle(int font, int direction, int font_size);*/

outtextxy(25,240,"Press any key to start the car");
                                    /*displays the text parameter on graphics
                                      console at x=25, y=240 */

getch();   /*to hold the output sceen and wait until user gives any type of input (i.e.
until user presses any key ) so that after the input we are able to see the output on
the screen. */
```

```
while(1)    // infinite loop since condition is always true, can be terminated by
                                                    goto, break, return statements
{
setactivepage(page);            // rear page on which actual drawing / rendering is
                                done
setvisualpage(1-page);          // front page which is visible, to which contents of
                    active page are transferred, similar to openGL double buffering

   cleardevice();      //clears the output screen

//SUN
setfillstyle(1,14);         // It sets the current fill pattern (1 = SOLID_FILL) and
                            fillcolor (14 = YELLOW )

circle(100+i,100,50);             // circle( centre_x , centre_y , radius )
floodfill(102+i,102,15);     /* floodfill( x , y, color ) colours the closed polygons
                    where (x,y) is within the boundary of the closed polygon */

//SUN RAYS
setcolor(14);                   // sets the colour of the lines drawn to 14=YELLOW
                                since the sun is yellow
if(i%10==0)          /* for rendering the sun rays. So only when the iterator is
completely divisible by 10 the sun rays will be shown, this produces a flickering or
blinking effect  */
{
setlinestyle(2,0,3);              // void setlinestyle(int linestyle, unsigned upattern,
                                                    int thickness);
```

```
line(100+i,50,100+i,0);            // line(x1,y1,x2,y2)
                                   //draws a line from (x1,y1) to (x2,y2)
line(150+i,100,220+i,100);
line(100+i,150,100+i,215);
line(50+i,100,-20+i,100);
line(0+i,30,75+i,75);
line(135+i,75,200+i,30);
line(75+i,125,0+i,168);
line(138+i,130,210+i,170);
setlinestyle(0,0,1);               //sets linestyle back to original format
}
```

//CAR BODY

```
setcolor(15);
line(50+i,370,90+i,370);
arc(110+i,370,0,180,20);/* void arc(int x, int y, int stangle, int endangle, int radius);
draws a circular arc at (x,y) with a radius given by radius. The arc travels from
stangle to endangle. Angles are anticlockwise*/
line(90+i,370,130+i,370);
line(130+i,370,220+i,370);
arc(240+i,370,0,180,20);
line(220+i,370,300+i,370);
line(260+i,370,300+i,370);
line(300+i,350,300+i,370);
line(300+i,350,240+i,330);
line(240+i,330,200+i,300);
line(200+i,300,110+i,300);
line(110+i,300,80+i,330);
line(80+i,330,50+i,340);
line(50+i,340,50+i,370);
line(165+i,365,165+i,335);
line(167+i,342,180+i,342);
```

```
setfillstyle(1,4);              //colours the car body red
floodfill(52+i,368,15);
```

//CAR Windows

```
setfillstyle(HATCH_FILL,15);     //Sets the Car window colour and it's filling style
                                 // setfillstyle(fill_style , fill_colour)
// first window
line(165+i,305,165+i,330); // line(x1 , y1 , x2 , y2 )
line(165+i,330,230+i,330);  // where start point is (x1 , y1) and end point is (x2 , y2)
line(230+i,330,195+i,305);
line(195+i,305,165+i,305);
floodfill(170+i,323,WHITE);       // floodfill (x,y coordinates of initial seed pixel ,
                                  boundary_colour)
// second window
line(160+i,305,160+i,330);
line(160+i,330,95+i,330);
line(95+i,330,120+i,305);
line(120+i,305,160+i,305);
floodfill(158+i,323,WHITE);
```

//Traffic signal

```
//pole for signal
setfillstyle(2,14);
rectangle(545-i,375,570-i,392);
floodfill(546-i,375,15);
rectangle(550-i,200,565-i,387);   //rectangle(upper_left_x, upper_left_y,
                                  lower_right_x, lower_right_y )
```

```
//signal
rectangle(530-i,100,585-i,200);
circle(557.5-i,125,20);        //Upper traffic light for STOP (red color)
floodfill(532-i,126,15);
circle(557.5-i,170,20);        //Lower traffic light for GO (green color)
floodfill(551-i,201,15);

if(i>120 && i<=300)      // when car is just before the signal
{
setfillstyle(SOLID_FILL,BLACK);     //upper red signal should turn black
floodfill(557.5-i,125,15);


outtextxy(535-i,50,"GO");              //lower signal should turn green from black
setfillstyle(SOLID_FILL,GREEN);
floodfill(557.5-i,170,15);

      if(i%10==0)       //for sun rays at the signal crossing
       {
       setcolor(14);
       setlinestyle(2,0,3);
       line(100+i,50,100+i,0);
       line(150+i,100,220+i,100);
       line(100+i,150,100+i,215);
       line(50+i,100,-20+i,100);
       line(0+i,30,75+i,75);
       line(135+i,75,200+i,30);
       line(75+i,125,0+i,168);
       line(138+i,130,210+i,170);
       setlinestyle(0,0,1);
        setcolor(15);
       }  //end inner if statement
```

```
}    // end outer if statement
else                // when car is not before the signal
   {
   outtextxy(525-i,50,"STOP");       //when car is not before the signal it should
                                                    be red for STOP
   setfillstyle(SOLID_FILL,RED);
   floodfill(557.5-i,125,15);
   setfillstyle(SOLID_FILL,BLACK);   //lower green signal should turn black
   floodfill(557.5-i,170,15);                //since red signal is on for STOP
   }
```

// CAR WHEELS

```
setfillstyle(SOLID_FILL,3);
pieslice(110+i,370,359-j,360-j,15);
pieslice(110+i,370,179-j,180-j,15);
pieslice(110+i,370,89-j,90-j,15);/* void pieslice(int x, int y, int stangle, int endangle,
int radius); draws and fills a pie slice centered at (x,y) with a radius given by radius.
The slice travels from stangle to endangle.*/
pieslice(110+i,370,269-j,270-j,15);
pieslice(240+i,370,359-j,360-j,15);
pieslice(240+i,370,179-j,180-j,15);
pieslice(240+i,370,89-j,90-j,15);
pieslice(240+i,370,269-j,270-j,15);
circle(111+i,370,17);  //rear wheel
circle(241+i,370,17);  //front wheel
floodfill(111+i,370,15);   //to colour the wheels in cyan colour
floodfill(241+i,370,15);
```

```
if(i<120 || i>121)
  ++j;                    //wheels should show rotation only at these vales of iterator
else
{
  j=0;                   //else wheels don't show rotation to show that car is at rest
                            position
  delay(1000);        // for clear visibility of rotation of wheels
}
```

// SCENE

```
line(0,390,2000,390);   //road

rectangle(800-(i),380,810-(i),390);  //stones on the road
rectangle(830-(i),375,840-(i),390); //(x1-(i),y1,x2-(i),y2)

setfillstyle(11,10);
rectangle(1550-i,200,1670,390);          //background walls
floodfill(1570-i,220,15);                 //colouring background walls

setfillstyle(9,13);
rectangle(1000-(i),300,1240-(i),390); //background walls
floodfill(1002-i,320,15);    //colouring background walls

if( i == 998 )        //to break infinite while loop so that car stops at the end of the
      break;             road

i++;                  //for incrementing iterator to execute the while loop
```

```
page=1-page;    //to switch between active and visual pages
delay(10);        //to see clear locomotion of the car


//to play music
if (i<500)
  outtextxy(100,25,"PLAYING MUSIC"); /*displays the text parameter on graphics
console at x=100, y=25 */
if (i>500)
  outtextxy(100,25,"MUSIC STOP");


if (i>985)
    outtextxy(100,125,"FUEL  TANK  IS  EMPTY");  //So car will stop

}  // end while

delay(1900);       //to hold the drawing on the screen for 1900 milliseconds after
                            car stops


exit(0);                 //to exit from the program
getch();//function to wait for some time until a key is hit and hold the output
window , given after running of program
closegraph();   //It unloads the graphics drivers and sets the screen back to text
                          mode
}
```

: REFERENCE TABLES :

Reference for parameters in setfillstyle (int pattern , int color)

PATTERN	INT VALUES
EMPTY_FILL	0
SOLID_FILL	1
LINE_FILL	2
LTSLASH_FILL	3
SLASH_FILL	4
BKSLASH_FILL	5
LTBKSLASH_FILL	6
HATCH_FILL	7
XHATCH_FILL	8
INTERLEAVE_FILL	9
WIDE_DOT_FILL	10
CLOSE_DOT_FILL	11
USER_FILL	12

COLOR	INT VALUES
BLACK	0
BLUE	1
GREEN	2
CYAN	3
RED	4
MAGENTA	5
BROWN	6
LIGHTGRAY	7
DARKGRAY	8
LIGHTBLUE	9
LIGHTGREEN	10
LIGHTCYAN	11
LIGHTRED	12
LIGHTMAGENTA	13
YELLOW	14
WHITE	15

Reference for parameters in

settextstyle (int font, int direction, int font_size)

COLOR	INT VALUES
DEFAULT_FONT	0
TRIPLEX_FONT	1
SMALL_FONT	2
SANS_SERIF_FONT	3
GOTHIC_FONT	4
SCRIPT_FONT	5
SIMPLEX_FONT	6
TRIPLEX_SCR_FONT	7
COMPLEX_FONT	8
EUROPEAN_FONT	9
BOLD_FONT	10

Reference for parameters in

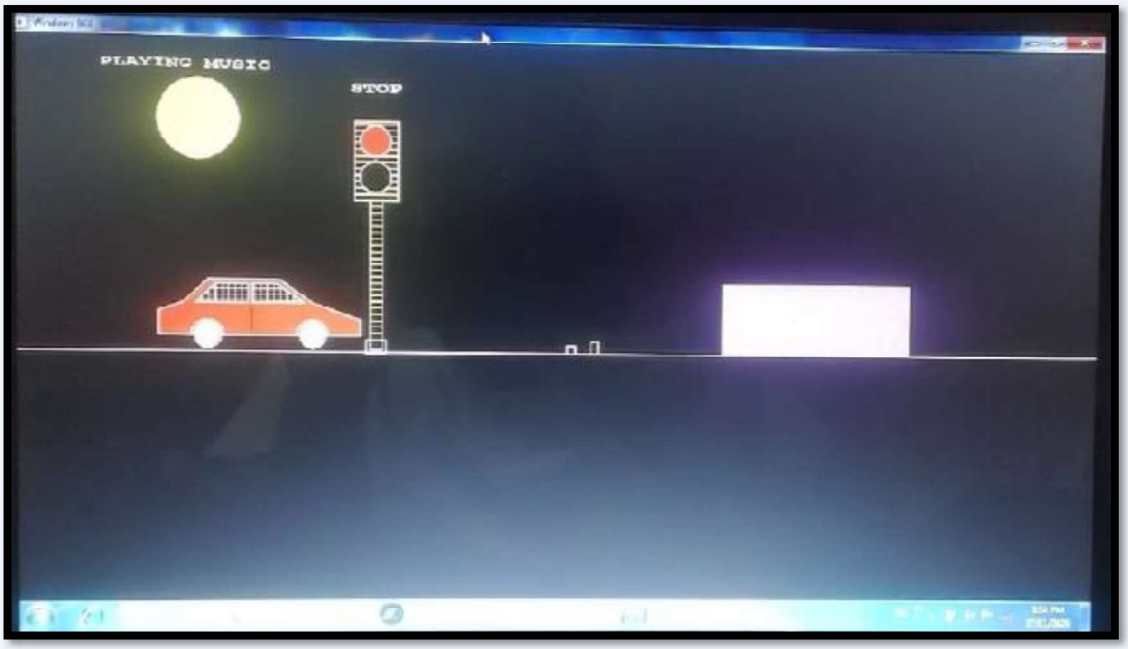
setlinestyle(int linestyle, unsigned pattern, int thickness)

linestyle	Value	Description
SOLID_LINE	0	Solid line
DOTTED_LINE	1	Dotted line
CENTER_LINE	2	Centered line
DASHED_LINE	3	Dashed line
USERBIT_LINE	4	User-defined line style

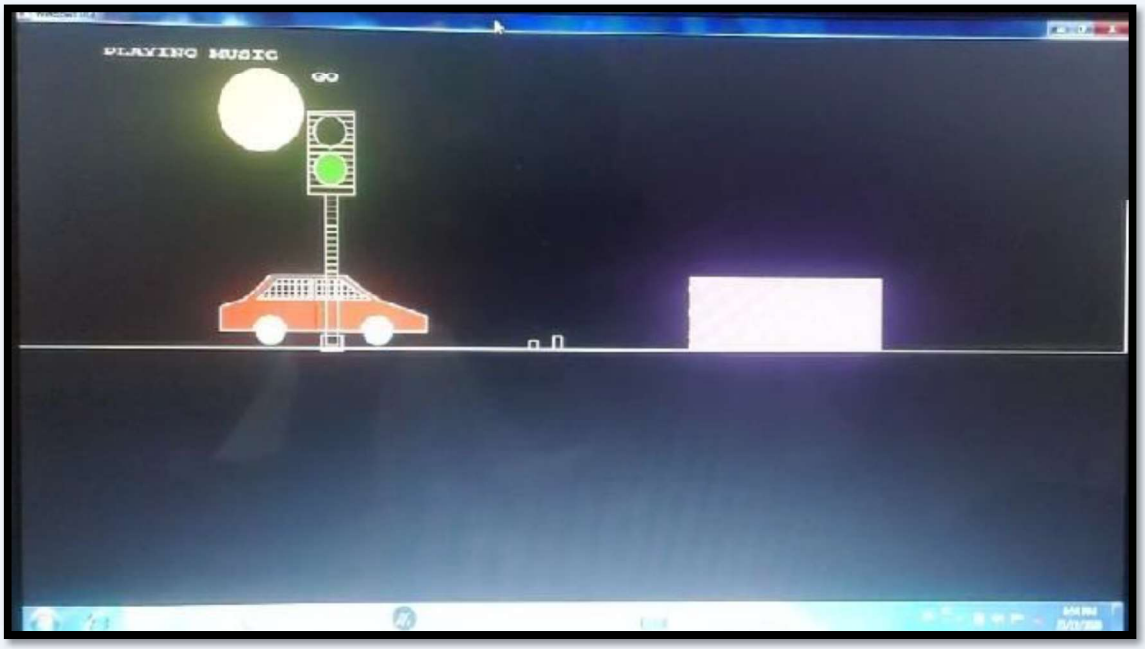
thickness	Value	Description
NORM_WIDTH	1	1 pixel wide
THICK_WIDTH	3	3 pixels wide

unsigned pattern is simply ignored if ‘linestyle’ is not USERBIT_LINE

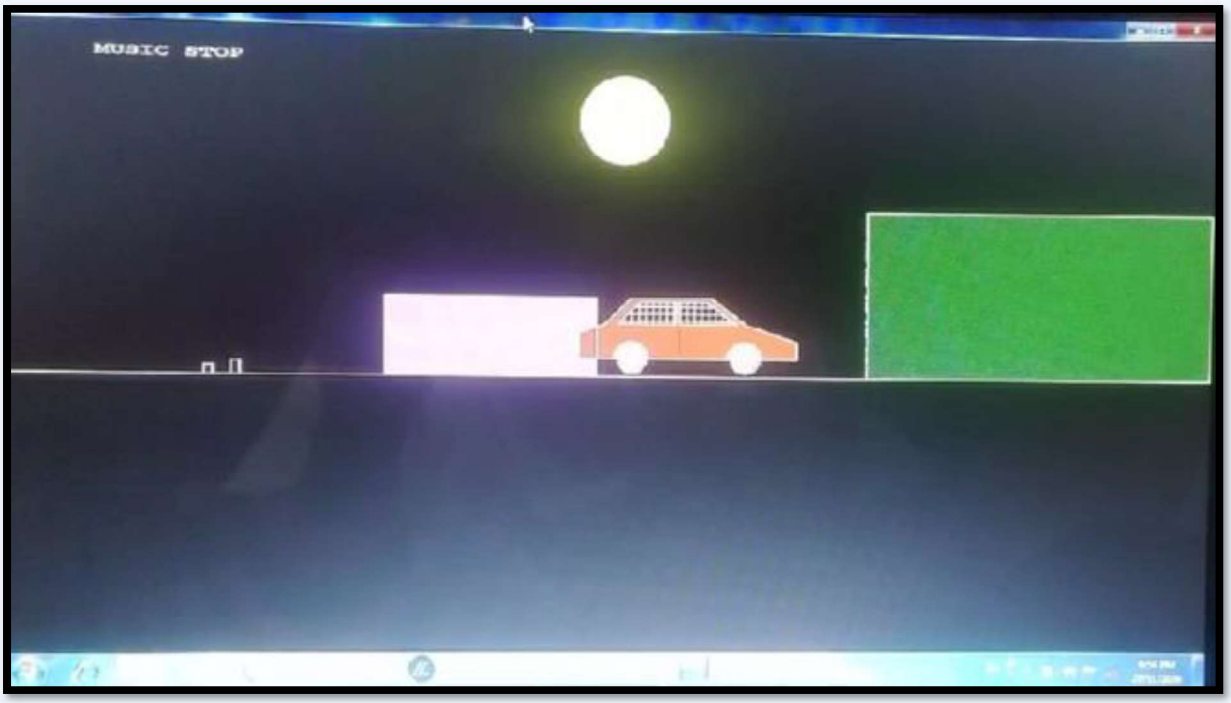
OUTPUT SCREENSHOTS



Car stops before the red signal



Car continues it’s motion once the signal turns green



: REFERENCE LINKS :

https://www.cs.colorado.edu/~main/bgi/doc/arc.html

https://www.cs.colorado.edu/~main/bgi/doc/initwindow.html

https://www.cs.colorado.edu/~main/bgi/doc/outtextxy.html

https://www.cs.colorado.edu/~main/bgi/doc/setcolor.html

https://www.cs.colorado.edu/~main/bgi/doc/setfillpattern.html