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()
{
int i=0,j,page=0;
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 straight 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);
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

```
line(165+i,305,165+i,330);
line(165+i,330,230+i,330);
line(230+i,330,195+i,305);
line(195+i,305,165+i,305);

setfillstyle(HATCH_FILL,15);    //Car window colour
floodfill(170+i,323,WHITE);

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);

//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);
         }
}
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

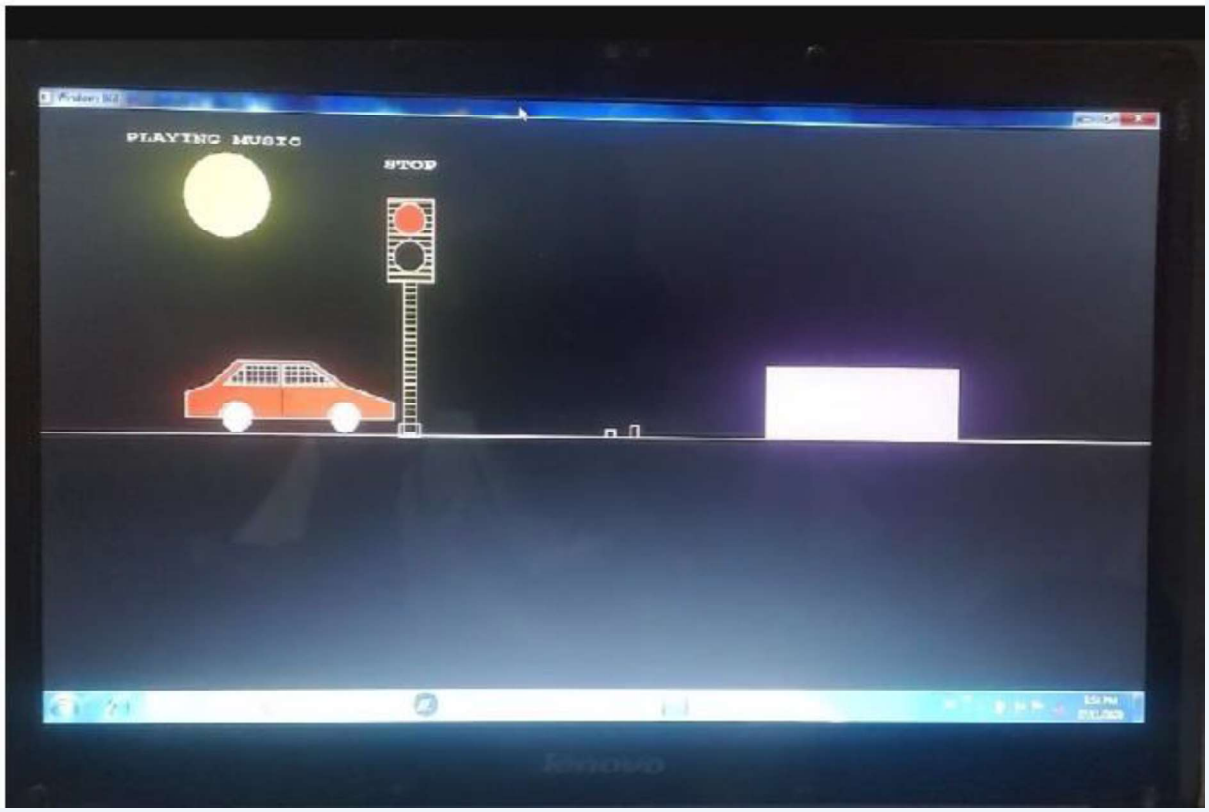
```
setfillstyle(SOLID_FILL,3);     //it sets the current fill pattern
pieslice(110+i,370,359-j,360-j,15); //(x1+i,y1,x1-j,y1-j,colour)
pieslice(110+i,370,179-j,180-j,15);
pieslice(110+i,370,89-j,90-j,15);
pieslice(110+i,370,269-j,270-j,15);
pieslice(240+i,370,359-j,360-j,15);
pieslice(240+i,370,179-j,180-j,15);
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
}
```

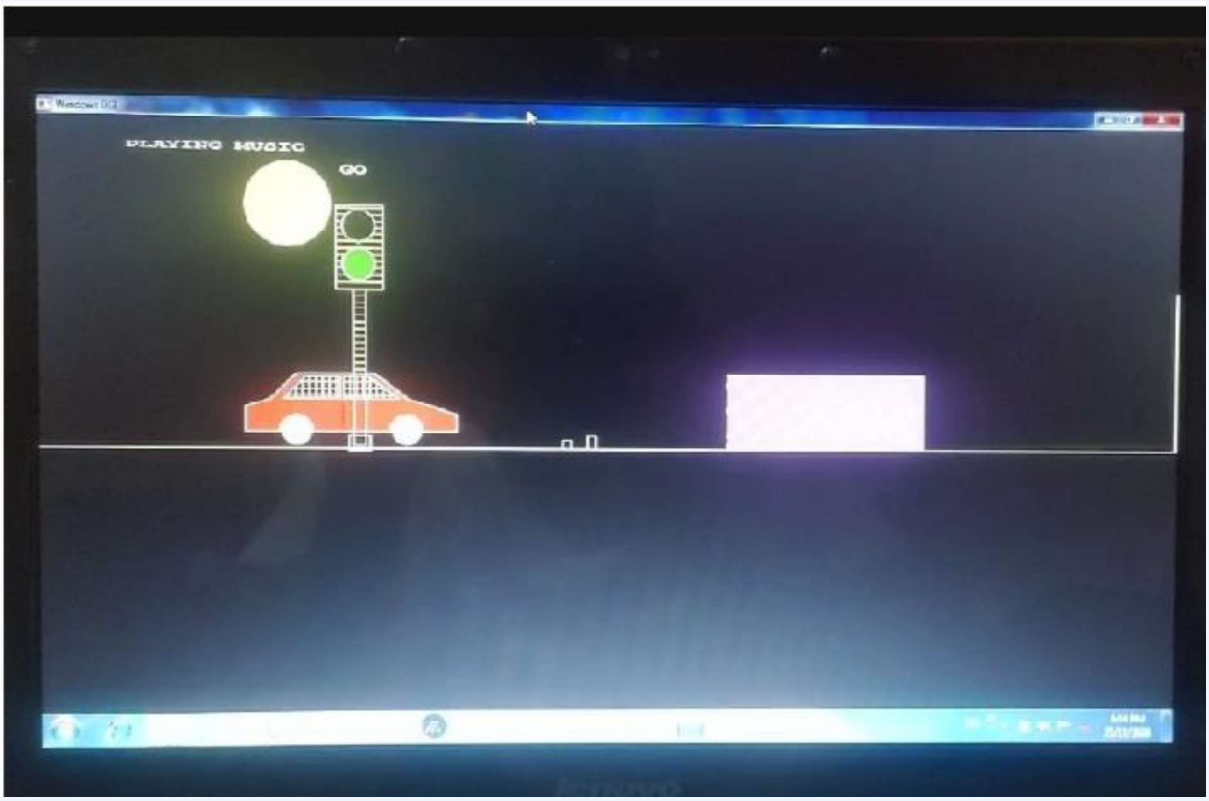
// SCENE

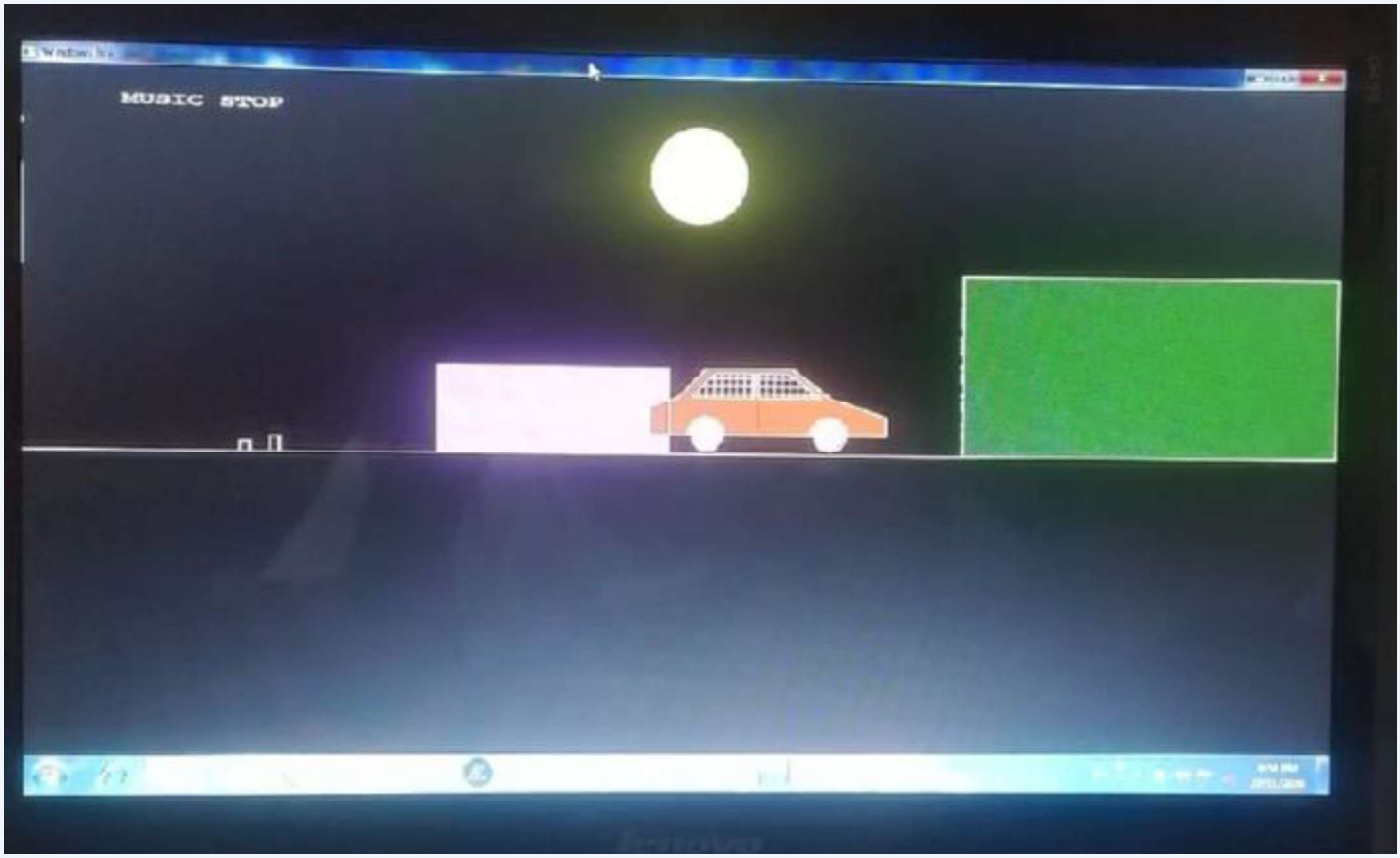
```
line(0,390,2000,390);   //road

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
}
```

OUTPUT SCREENSHOTS







: 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 upattern, 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 upattern is simply ignored if ‘linestyle’ is not USERBIT_LINE