**Software Project report**

**On**

****

**Department of Computer Science and Engineering**

CSE-1211

Software Project Report

On

“Sky Roads”

**Submitted to:**

1. Dr. Muhammad Asif Hossain Khan

Associate Professor, Department of CSE.

1. Ms. Anna Fariha

Lecturer, Department of CSE.

**Submitted by:**

1. Tanzir Islam.

Roll no: 01.

1. Shahreen Salim Aunti.

Roll no: 33.

**Date of submission:** 04.12.2014.

**Contents**

1. Introduction 4
2. Objective 4
3. The game 4
4. Main features 5
5. A flowchart on how the game works 8
6. Variables and data types 9
7. Data structures 9
8. User defined functions 10
9. Algorithm 11
10. Source code 12
11. Constraints 32
12. Augmentation 33
13. Conclusion 33
14. References 34

**Introduction:**

C and C++ are general purpose programming languages. They can be used to create graphical interface. In our project we tried to make “Sky Roads”, a replica of the renowned game “Skyroads”, using BGI functions.

Skyroads is a game of MS-DOS platform developed in 1993 by Bluemoon software developers. We are pretty sure that almost everyone played this video game.

It’s a kind of futuristic racing game where players have to use space bar and directions keys to jump and move. It’s an easy game to play, and it shows a 3D perspective to the players.

**Objective:**

Our objective was to create a simplified version of “Skyroads”. It is programmed in C++ using simple BGI functions. The compiler used is Microsoft visual C++ 2010.

**The game:**

“SKY ROADS” is a very simple game. It shows a road that goes for a certain period of time. After that a new road is created and the player has to take his/her aircraft with certain moves on that road to survive. There are also obstacles on those roads. The player has to jump across the obstacle or move aside to avoid collision. If he/she fails to do so, the aircraft will collide with the obstacle and the game will be over. The objective of the players should be to survive as long as possible and to complete all the levels in shortest time possible to win.

**Main features:**

1. **Main menu:**

* **Play tab:** Goes to Levels menu when enter is pressed.
* **Instructions tab:** Goes to a page where the game instructions are written.
* **Leader-board tab:** Goes to a page where the best 10 players records are stored.
* **Exit tab:** Exits the game when enter is pressed.

****

**Figure:** Screenshot of the game menu.

1. **The Levels:**

* **Level 1:** This is the simpler level of the game. Here a an aircraft goes on a road and faces same type of obstacle. The player has to make his aircraft jump to avoid collision.
* **Level 2:** The second level is comparatively difficult from the first one. Here multiple roads are created and the player has to take his aircraft on those roads when it’s necessary. The obstacles are also of different kind where the player not only can jump across but also can move aside. A player has to play more efficiently than the previous level to complete this one.



**Game levels**

The player can choose any one level to start the game. That means he/she doesn’t have to complete one level to unlock another.

1. **Leader board:**

We included a leader board that contains the best 10 players records. If anyone fails to break the 10th best record, he will not be able to enter his name. That means your name will be asked if and only if your score ( which is counted in seconds) is good enough!



**Leader board**

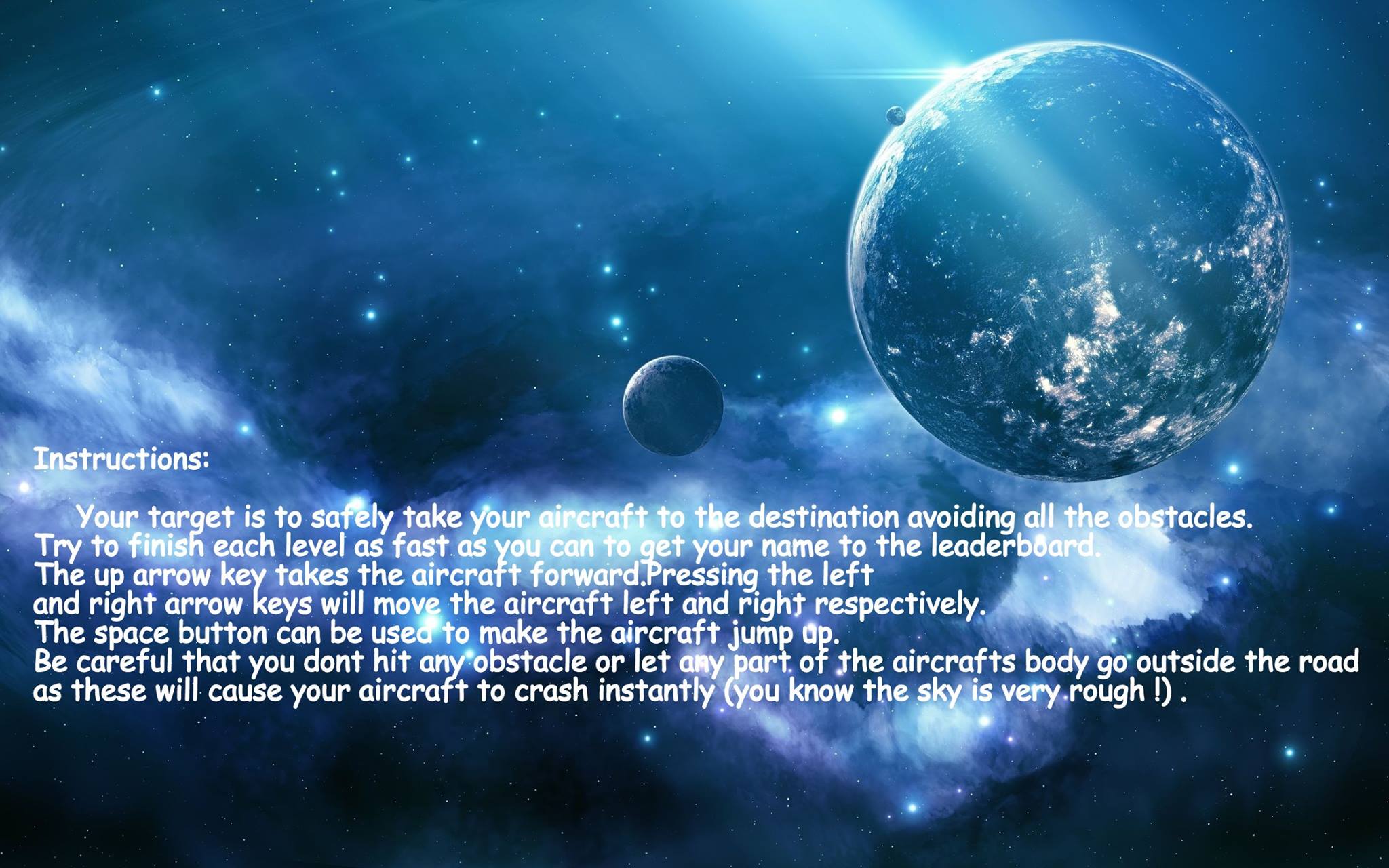
1. **Instructions:**

The game screen consists of a road and an aircraft placed on it. The aircraft has to be moved forward in order to start the game. It’s totally a keyboard controlled game. The player has to use certain keys to perform following tasks.

* To move forward, the player has to press the “UP key”.
* To move right, the player has to press the “Right key”.
* To move left, the player has to press the “Left key”.
* To jump, the player has to press the “Space bar”.

The player must any kind of collisions to complete a level. Because, no matter how far he gets , once the game is over, he has to start from the scratch in that level.

And another important fact to notice is that, if the aircraft goes too right or too left so that the ends, the game will also be over.



**Game instructions**

**A flow chart on how the game works:**

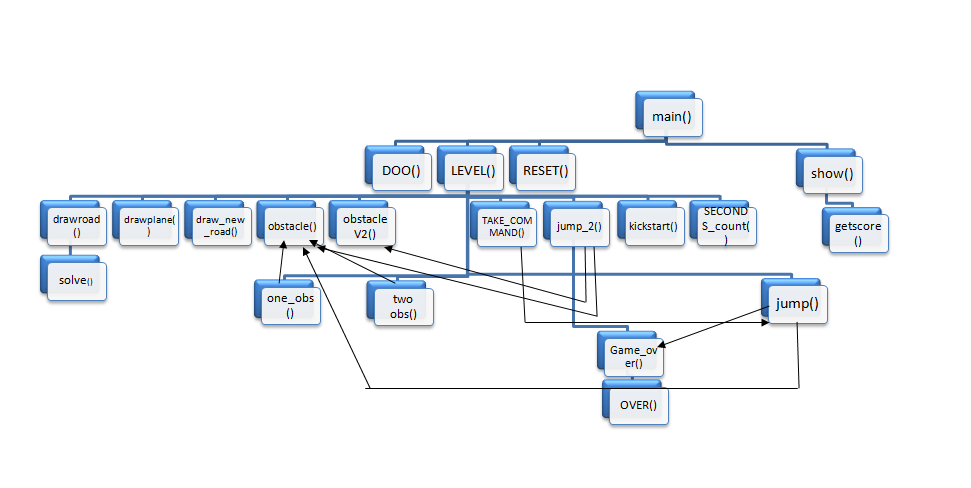
**The flowchart on how game works**

**Variables and data types:**

We had to declare a lot of global variables in order to make things easier. We tried to keep the data type mostly as integer, but character and double were also used.

**Data structures:**

We used structures, integer arrays and character arrays, vectors, pair<double,double> and some user defined data structures to store the coordinates of roads, aircrafts and obstacles.

 A flowchart on how the functions call each other

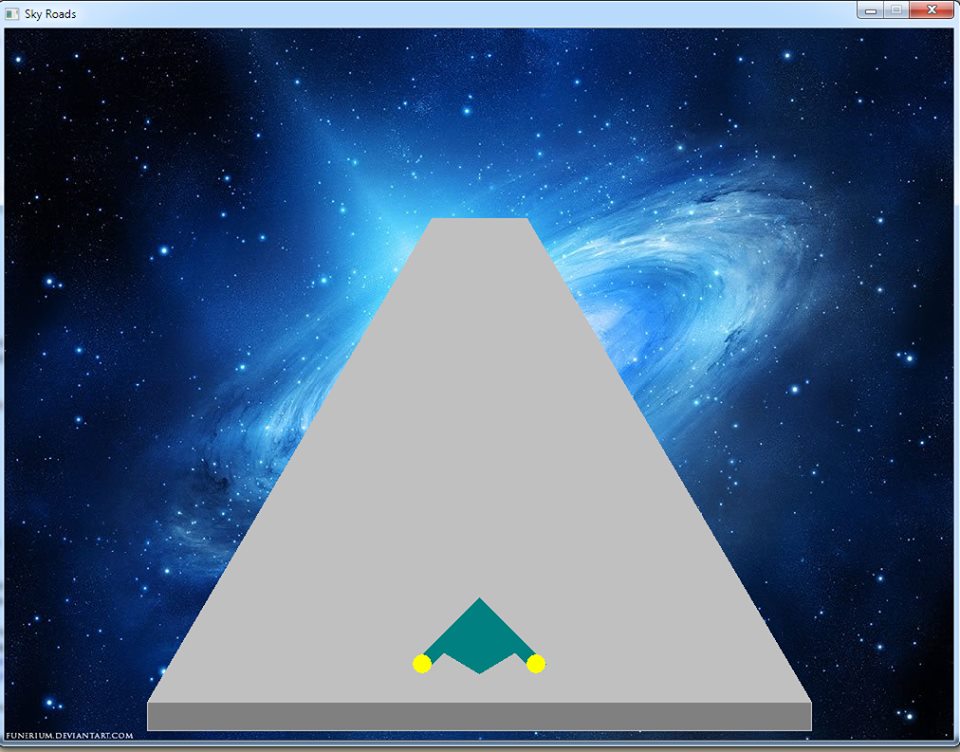
**User defined functions:**

1. void drawroad()- draws the road in an exact positions using some fixed coordinates which was declared as integer arrays.
2. void drawplane( int moved, int height)- draws the aircraft in different positions according to the command. Here the argument “moved” is used to move the plane left or right and “height” draws the plane on a definite height.
3. void draw\_new\_road(pii ltop, pii rbtm,int twidth,int bwidth,int color)- draws a new road in different positions using left top points, right bottom points, width of the top and bottom portion and color.
4. void jump()- helps the aircraft to jump across the obstacle and land on it, by drawing it in two different loops. One loop is for moving up and one is for moving down.
5. void jump\_2()-helps the aircraft to jump across the obstacle in a new pattern in same way.
6. void obstacle(int position, int height)- draws obstacles using the function fillpoly.
7. void obstacleV2(int lbtmx, int lbtmy, int lgth, int height, int color)- draws another type of obstacle using the function fillpoly.
8. void one\_obs()- brings one obstacle at a time using a loop.
9. void two\_obs()- brings two obstacle at a time using a loop.
10. void TAKE\_COMMAND()- takes command from keyboard and calls function according to it.
11. int DOO()- returns value for fixing the menu.
12. int show(int pres)- reads image file according to the return value of DOO().
13. int Game\_over()- if the conditions of game over are true, it returns 1.
14. void OVER()- if the Game\_over() function returns 1, this function will be called and it will make the Game over.
15. void getscore()- works for leader board. Appends names and scores as a string and show them on graphical window using outtextxy() function.
16. void kickstart()- detects when to start the game by counting the number if commands at the starting point.
17. void LEVEL(int lev)- detects the level and calls obstacle and jump functions
18. void RESET()- resets every variables after every level.
19. void SECONDS\_count()- counts the time for scoring.
20. SL solve()- solves a straight line equation using y=mx+c.

**Algorithm:**

As our game is not a logic-based game, we didn’t have to follow any definite algorithm.

We applied brute force, that means, our aim was to check results for all possible moves, and then do according to them. For instance: when the player makes the aircraft jump, we even checked if the jump was accurate enough to surpass the obstacle. By accuracy we mean distance from the obstacle and perfect timing.

 **The starting condition**

**Source code:**

Header file:

|  |
| --- |
| #include "graphics.h"  #include <vector>  #include <math.h>  #include <queue>  #include <stack>  #include <set>  #include <iostream>  #include <map>  #include <cstdio>  #include <string.h>  using namespace std;  int calc(string str);  void getln(char \*str);  int leader(int seconds,bool lev); |

Source code:

|  |
| --- |
| #include “leaderboard.h”  #define inf (1 << 30 – 1)  #define pii pair<double,double>  #define mp make\_pair  #define fi first  #define se second  #define xx fi  #define yy se  #define PRE\_TP 600  #define PRE\_BT 680  #define dim1 1000  #define dim2 750  #define KEY\_UP 72  #define KEY\_LEFT 75  #define KEY\_RIGHT 77  #define KEY\_DOWN 80  #define KEY\_ESCAPE 27  typedef long long int LL;  typedef double db;  using namespace std;  pii TOPL,BOTL,TOPR,BOTR,tempL,tempR;  //faka rasta  int Y1,Y2,y2,y3,tw1,tw2,bw2,tw3,tw4,hh;  double m1,c1,m2,c2;  pii lt1,lt2,rb1,rb2,rt2,lt3,rb3;  //this is the road!  Int roadbottom[8]= {150,710,150,740,850,740,850,710};  int Road[8]= {450,200,150,710,850,710,550,200};  int road[8]= {450,200,113,750,875,750,550,200};  //this is the spacecraft!  Int plbody[8]= {500,600,450,650,500,680,550,650};  int pllft[8]= {450,650,430,670,450,670,470,650};  int plrgt[8]= {530,650,550,670,570,670,550,650};  // variables  int h,I,j,k,nwh,nwx,nwb,nwbh,hb,rb,A[8],q,cnt=0,lb,ht,I;  int O2,h2;  char cmnd;  int moved=0,SECONDS;  int level=0;  //these variables are for “Game\_over” function:  int obslftx,obsrgtx,obstpy,obsbty,planetpy,planebty,rdlftx,rdrgtx,rdtpy,planergtx,planelftx;  int lightxl=440;  int lightxr=560;  int ightly=670;  int lightyr=670;  bool FINISH=0;  // VARIABLES FOR URADHURA OBS  int O1X,O1Y,L1;  int height;  int O2X,O2Y,L2;  int H2;  int O3X,O3Y,L3;  int H3;  int O4X,O4Y,L4;  int H4;  int O5X,O5Y,L5;  int H5;  struct quadrilateral  {  pii top;  pii lft;  pii bot;  pii rght;  } spcB,spcL,spcR;  void jump();  void jump\_2();  struct SL  {  db m;  db c;  } lft,rght;  SL solve(pii a,pii b)  {  SL ret;  ret.m=(a.yy-b.yy)/(a.xx-b.xx);  ret.c=a.yy-ret.m\*a.xx;  return ret;  }  void drawroad()  {  setcolor(7);  setfillstyle(SOLID\_FILL,7);  fillpoly(4,road);  }  void Drawroad()  {  setcolor(7);  setfillstyle(SOLID\_FILL,7);  fillpoly(4,Road);  setfillstyle(SOLID\_FILL,8);  fillpoly(4,roadbottom);  }  void draw\_new\_road(pii ltop ,pii rbtm,int twidth,int bwidth,int color)  {  pii lbtm, rtop;  rtop.xx=ltop.xx+twidth;  rtop.yy=ltop.yy;  lbtm.xx=rbtm.xx-bwidth;  lbtm.yy=rbtm.yy;  int newroad[8]= {ltop.xx,ltop.yy,lbtm.xx,lbtm.yy,rbtm.xx,rbtm.yy,rtop.xx,rtop.yy};  int newroadbtm[8]= {lbtm.xx,lbtm.yy,lbtm.xx,lbtm.yy+25,rbtm.xx,rbtm.yy+25,rbtm.xx,rbtm.yy};  setcolor(color);  setfillstyle(SOLID\_FILL,color);  fillpoly(4,newroad);  int btclr;  if(color==7)  btclr=8;  else if(color==10)  btclr=2;  else if(color==9)  btclr=1;  setcolor(btclr);  setfillstyle(SOLID\_FILL,btclr);  fillpoly(4,newroadbtm);  }  void drawplane(int moved,int height)  {  int m;  for(m = 0; m <= 7; m +=2)  {  plbody[m]+=moved;  pllft[m]+=moved;  plrgt[m]+=moved;  }  int h;  for(h = 1; h <= 7; h+=2)  {  plbody[h]-=height;  pllft[h]-=height;  plrgt[h]-=height;  }  setcolor(3);  setfillstyle(SOLID\_FILL,3);  fillpoly(4,plbody);  setfillstyle(SOLID\_FILL,3);  fillpoly(4,pllft);  setfillstyle(SOLID\_FILL,3);  fillpoly(4,plrgt);  setfillstyle(SOLID\_FILL,3);  setcolor(14);  setfillstyle(SOLID\_FILL,14);  lightxl+=moved;  lightxr+=moved;  ightly-=height;  lightyr-=height;  fillellipse(lightxl,ightly,10,10);  fillellipse(lightxr,lightyr,10,10);  }  void obstacle(int pos,int height)  {  TOPL.xx=450;  TOPL.yy=200;  BOTL.xx=150;  BOTL.yy=710;  TOPR.xx=550;  TOPR.yy=200;  BOTR.xx=850;  BOTR.yy=710;  lft=solve(TOPL,BOTL);  rght=solve(TOPR,BOTR);  h=1;  setcolor(RED);  tempL.yy=pos;  tempL.xx=(tempL.yy-lft.c)/lft.m;  tempR.yy=pos;  tempR.xx=(tempR.yy-rght.c)/rght.m;  nwh=h+height/2;  int A[8]= {tempL.xx,tempL.yy,tempL.xx,tempL.yy+nwh,tempR.xx,tempR.yy+nwh,tempR.xx,tempR.yy};  setfillstyle(SOLID\_FILL,RED);  fillpoly(4,A);  obslftx=0;  obsrgtx=inf;  }  void obstacleV2(int lbtmx,int lbtmy,int lgth,int height,int color)  {  int rbtmx=lbtmx+lgth;  int rbtmy=lbtmy;  int nwh=h+height/2;  int C[8]= {lbtmx,lbtmy-nwh,lbtmx,lbtmy,rbtmx,rbtmy,rbtmx,rbtmy-nwh};  setcolor(color);  setfillstyle(SOLID\_FILL,color);  fillpoly(4,C);  obstpy=lbtmy-nwh;  obslftx=lbtmx;  obsrgtx=rbtmx;  }  void SECONDS\_count()  {  int q=0;  while(!kbhit())  {  q++;  if(q==5500000)  {  SECONDS++;  q=0;  }  }  return;  }  int Game\_over()  {  //return 0;  if(level==1)  {  rdrgtx=836;  rdlftx=155;  rdtpy=0;  }  planetpy=plbody[1];  planebty=planetpy+80;  if(rdtpy-30>planetpy && rdtpy>=600 && planebty>=680)  {  return 1;  }  if(lightxr>=rdrgtx)  return 1;  if(lightxl<=rdlftx)  return 1;  if(obstpy<510)  return 0;  if(obslftx>lightxr || obsrgtx<lightxl)  return 0;  if(obstpy+100>=planetpy && obstpy+80<=planebty)  {  return 1;  }  return 0;  }  void OVER()  {  int c;  FINISH=1;  setfillstyle(SOLID\_FILL,BLACK);  fillellipse(500,375,10000,10000);  delay(300);  readimagefile(“game\_over.jpg”,0,0,1000,750);  delay(500);  while(1)  {  c=getch();  if(c==13)  return;  }  }  void TAKE\_COMMAND()  {  take\_command2:  ;  SECONDS\_count();  cmnd=getch();  if(cmnd==0)  {  cmnd=getch();  if(cmnd==KEY\_RIGHT)  drawplane(12,0);  else if (cmnd==KEY\_LEFT)  drawplane(-12,0);  else if(cmnd!=KEY\_UP)  {  obstacle(lb,ht);  drawplane(0,0);  cmnd=-1;  goto take\_command2;  }  }  else if(cmnd==’ ‘)  {  if(level==1 || level==2)  jump();  else if(level==3)  jump\_2();  }  else  {  obstacle(lb,ht);  drawplane(0,0);  cmnd=-1;  goto take\_command2;  }  if(FINISH==1)  return;  drawplane(0,0);  cmnd=-1;  }  void jump( )  {  int cnt=0;  // this for loop takes the plane upwards  for(q=-1; q>=-410; q-=45,lb+=9,ht+=6,cnt++)  {  if(cnt==15)  {  SECONDS++;  cnt=0;  }  if(lb>400)  {  lb+=1;  if(I%2==0)  {  obstacle(O2,h2);  if(lb>650)  O2+=2;  O2+=6;  h2+=4;  obstpy=O2;  }  }  if(lb<=750)  {  obstacle(lb,ht);  obstpy=lb;  }  if(FINISH==1)  return;  if(Game\_over())  {  OVER();  cout << “hello” << endl;  return;  }  cout << “asd” << endl;  if(FINISH==1)  return;  drawplane(0,30);  readimagefile(“space-sunrise.jpg”,0,0,dim1,dim2);  drawroad();  }  //this for loop is for taking the plane downwards.  // comments above are also applicable here  for(q=-410; q<=0; q+=45,ht+=6,lb+=9,cnt++)  {  if(cnt==15)  {  SECONDS++;  cnt=0;  }  if(lb>400)  {  lb+=1;  if(I%2==0)  {  obstacle(O2,h2);  if(lb>650)  O2+=2;  O2+=6;  h2+=4;  obstpy=O2;  }  }  if(lb<=750)  {  obstacle(lb,ht);  obstpy=lb;  }  if(FINISH==1)  return;  if(Game\_over())  {  OVER();  return;  }  if(FINISH==1)  return;  drawplane(0,-30);  readimagefile(“space-sunrise.jpg”,0,0,dim1,dim2);  drawroad();  }  }  void one\_obs()  {  int cnt=0;  for(lb=200,ht=1; lb<=750; lb+=6,ht+=4,cnt++)  {  drawplane(0,0);  if(cnt==25)  {  cout << SECONDS << endl;  SECONDS++;  cnt=0;  }  if(lb>350)  lb+=1;  obstacle(lb,ht);  obstpy=lb;  q=0;  TAKE\_COMMAND();  if(FINISH==1)  return;  if(Game\_over())  {  OVER();  return;  }  if(FINISH==1)  return;  drawroad();  }  }  void two\_obs()  {  int cnt=0;  for(lb=200,ht=1,O2=200,i=1,h2=1; lb<=1000; lb+=6,ht+=4,i++,cnt++)  {  drawplane(0,0);  if(cnt==25)  {  SECONDS++;  cnt=0;  }  if(lb>=400)  {  lb+=1;  obstacle(O2,h2);  if(lb>650)  O2+=1;  O2+=6;  h2+=4;  obstpy=O2;  }  if(lb<=750)  {  obstacle(lb,ht);  obstpy=lb;  }  TAKE\_COMMAND();  if(FINISH==1)  return;  if(Game\_over())  {  OVER();  return;  }  if(FINISH==1)  return;  drawroad();  }  }  void getscore(int lev)  {  readimagefile(“leaders.jpg”,0,0,dim1,dim2);  int I,ycor;  setbkcolor(WHITE);  setcolor(BLUE);  settextstyle(GOTHIC\_FONT,HORIZ\_DIR,5);  outtextxy(270,100,”LEADER BOARD”);  settextstyle(BOLD\_FONT,HORIZ\_DIR,5);  outtextxy(80,200,”NAME”);  outtextxy(740,200,”TIME”);  settextstyle(BOLD\_FONT,HORIZ\_DIR,4);  FILE \*fp;  if(lev==1)  fp=fopen(“in.txt”,”r”);  else  fp=fopen(“in2.txt”,”r”);  char name[20],a[5],score[10],c=’.’;  for(i=1,ycor=250; i<=10; i++,ycor+=50)  {  if(1!=fscanf(fp,”%s”,name))  break;  fscanf(fp,”%s”,score);  ightl(a,”%2d”,i);  outtextxy(80,ycor,a);  ightl(a,”%c”,c);  outtextxy(120,ycor,a);  outtextxy(140,ycor,name);  settextstyle(BOLD\_FONT,HORIZ\_DIR,4);  outtextxy(740,ycor,score);  }  fclose(fp);  return;  }  void show(int pres)  {  if(pres==0)  readimagefile(“play.jpg”,0,0,dim1,dim2);  if(pres==1)  readimagefile(“Instructions.jpg”,0,0,dim1,dim2);  if(pres==2)  readimagefile(“Leaderboard.jpg”,0,0,dim1,dim2);  if(pres==3)  readimagefile(“Exit.jpg”,0,0,dim1,dim2);  if(pres==5)  readimagefile(“ins.jpg”,0,0,dim1,dim2);    if(pres==7)  readimagefile(“level1.jpg”,0,0,dim1,dim2);  if(pres==8)  readimagefile(“level2.jpg”,0,0,dim1,dim2);  if(pres==9)  readimagefile(“lev1.jpg”,0,0,dim1,dim2);  if(pres==10)  readimagefile(“lev2.jpg”,0,0,dim1,dim2);  if(pres==11)  {  readimagefile(“leaders.jpg”,0,0,dim1,dim2);  getscore(1);  while(1)  {  int command=getch();  if(command==KEY\_ESCAPE)  return;  }  }  if(pres==12)  {  readimagefile(“leaders.jpg”,0,0,dim1,dim2);  getscore(2);  while(1)  {  int command=getch();  if(command==KEY\_ESCAPE)  return;  }  }  return;  }  int DOO(int pres) /// -1 means exit, -2 means go to main menu  {  int command;  if(pres==3)  return -1;  if(pres==2)  {  int present=9;  show(9);  while(1)  {  command=getch();  if(command==KEY\_ESCAPE)  return -2;  if(command==13)  {  show(present+2);  show(present);  }  if(command==0)  {  command=getch();  if(command==KEY\_UP)  {  present=max(9,present-1);  show(present);  }  else if(command==KEY\_DOWN)  {  present=min(10,present+1);  show(present);  }  }  }  }  if(pres==1)  {  show(5);  while(1)  {  command=getch();  if(command==27)  return -2;  }  }  if(pres==0)  {  int present=7;  while(1)  {  show(present);  command=getch();  if(command==13)  return present;  else if(command==27)  return -2;  else if(command==0)  {  command=getch();  if(command==KEY\_UP)  present=max(7,present-1);  else if(command==KEY\_DOWN)  present=min(8,present+1);  }  }  }  }  char str[30];  void kickstart()  {  int cnt=0;  while(1)  {  SECONDS\_count();  cmnd=getch();  if(cmnd==0)  {  cmnd=getch();  if(cmnd==KEY\_UP)  {  delay(500);  return;  }  }  }  }  void LEVEL(int lev)  {  int command;  if(lev==1)  {  readimagefile(“space-sunrise.jpg”,0,0,dim1,dim2);  Drawroad();  drawplane(0,0);  SECONDS\_count();  kickstart();  drawroad();  drawplane(0,0);  while(I<5)  {  if(FINISH==1)  break;  I++;  if(I%2==1)  one\_obs();  else  two\_obs();    }  readimagefile(“leaders.jpg”,0,0,dim1,dim2);  if(FINISH==0)  leader(SECONDS,0);  getscore(1);  while(1)  {  command=getch();  if(command==KEY\_ESCAPE)  return;  }  }  else  {  readimagefile(“space-sunrise.jpg”,0,0,dim1,dim2);  rb1.xx=875,rb1.yy=750;  SL eq1=solve(mp(450,200),mp(113,750));  m1=eq1.m;  c1=eq1.c;  SL eq2=solve(mp(550,200),mp(875,750));  m2=eq2.m;  c2=eq2.c;  Drawroad();  drawplane(0,0);  SECONDS\_count();  kickstart();  ///whole level 2  {  drawroad();  drawplane(0,0);  rdrgtx=836;  rdlftx=155;  I=2;  two\_obs();    I=1;  if(FINISH!=1)  one\_obs();    level=3;  y2=180;  lt2.yy=y2;  lt2.xx=(lt2.yy-c1)/m1;  rt2.yy=y2;  rt2.xx=(rt2.yy-c2)/m2;  O1X=lt2.xx,O1Y=lt2.yy,L1=20;  L2=20,O2X=rt2.xx-L2,O2Y=rt2.yy;  O3X=lt2.xx,O3Y=lt2.yy,L3=20;  O4X=rt2.xx-L2,O4Y=rt2.yy+15,L4=20,H4=30;  O5X=lt2.xx,O5Y=lt2.yy+15,L5=90,H5=H4;  int cnt=0;  for(Y1=200,Y2=200,y2=200,tw1=100,height=10,tw2=100,tw3=100,tw4=100,hh=10; Y1<=5000 && FINISH!=1; Y1+=10,tw1+=10,cnt++)  {  if(cnt>=10)  {  cnt=0;  SECONDS++;  }  lt1.yy=Y1;  lt1.xx=(lt1.yy-c1)/m1;  rb2.yy=lt1.yy-70;  rb2.xx=lt1.xx+tw1;  if(Y1>2600)  {  lt2.yy=Y2;  lt2.xx=(lt2.yy-c1)/m1;  rb2.yy=rb1.yy;  rb2.xx=rb1.xx-100;  draw\_new\_road(lt2,rb2,tw3,650,10);  rdtpy=lt2.yy;  tw3+=10;  Y2+=10;  lt3.yy=y2;  lt3.xx=(lt3.yy-c1)/m1;  rb3.yy=lt2.yy-70;  rb3.xx=lt2.xx+tw3;  draw\_new\_road(lt3,rb3,tw4,tw3,9);  if(rb3.xx-tw3<700)  obstacleV2(rb3.xx-tw3,rb3.yy,tw3,hh,4);  hh+=3;  rb3.yy=lt2.yy-70;  rb3.xx=lt2.xx+tw3;  if(lt2.yy>600)  rdtpy=0;  }  else  {  draw\_new\_road(lt2,rb2,tw2,tw1,10);  rdtpy=0;  }  draw\_new\_road(lt1,rb1,tw1,762,7);  drawplane(0,0);  if(Y1<=750)  rdtpy=Y1;  if(Y1>1000 && Y1<1600)  {  obstacleV2(O1X,O1Y,L1,height,5);  O1X-=5;  O1Y+=10;  L1+=5;  height+=4;  }  if(Y1>1700 && Y1<2550)  {  obstacleV2(O3X,O3Y,L3,height-270,6);  O3X-=5;  O3Y+=10;  L3+=3;  height+=2;  }  if(Y1>1500 && O2Y<700)  {  obstacleV2(O2X,O2Y,L2,height-200,12);  O2X-=1;  O2Y+=10;  L2+=4;  height+=2;  }  if(Y1>3400)  {  obstacleV2(O4X,O4Y,L4,H4,4);  O4X-=1;  O4Y+=10;  L4+=5;  H4+=3;  }  if(Y1>3900)  {  obstacleV2(O5X,O5Y,L5,H5,13);  O5X-=3;  O5Y+=10;  L5+=6;  H5+=3;  }    TAKE\_COMMAND();  if(FINISH==1)  break;  if(Game\_over())  {  OVER();  break;  }  readimagefile(“space-sunrise.jpg”,0,0,dim1,dim2);  }  }  readimagefile(“leaders.jpg”,0,0,dim1,dim2);  if(FINISH==0)  leader(SECONDS,1);  getscore(2);  while(1)  {  command=getch();  if(command==KEY\_ESCAPE)  return;  }  }  return;  }  void RESET()  {  //Resetting plbody  plbody[0]=500;  plbody[1]=600;  plbody[2]=450;  plbody[3]=650;  plbody[4]=500;  plbody[5]=680;  plbody[6]=550;  plbody[7]=650;  //Resetting pllft  pllft[0]= 450;  pllft[1]= 650;  pllft[2]= 430;  pllft[3]= 670;  pllft[4]= 450;  pllft[5]= 670;  pllft[6]= 470;  pllft[7]= 650;  //Resetting plrgt  plrgt[0]= 530;  plrgt[1]= 650;  plrgt[2]= 550;  plrgt[3]= 670;  plrgt[4]= 570;  plrgt[5]= 670;  plrgt[6]= 550;  plrgt[7]= 650;  h=0,j=0,k=0,nwh=0,hb=0,rb=0,q=0,cnt=0,lb=0,ht=0,I=0;  O2=0,h2=0;  cmnd=-1;  moved=0,SECONDS=0;  level=0;  //these variables are for “Game\_over” function:  obslftx=0,obsrgtx=0,obstpy=0,obsbty=0,planetpy=0,planebty=0,rdlftx=0,rdrgtx=0,rdtpy=0;  lightxl=440;  lightxr=560;  ightly=670;  lightyr=670;  FINISH=0;  }  void jump\_2()  {  int cnt=0;  // this for loop takes the plane upwards  for(q=-1; q>=-410; q-=45,cnt++,Y1+=20,tw1+=20)  {  drawplane(0,30);  lt1.yy=Y1;  lt1.xx=(lt1.yy-c1)/m1;  rb2.yy=lt1.yy-70;  rb2.xx=lt1.xx+tw1;  if(Y1>2600)  {  lt2.yy=Y2;  lt2.xx=(lt2.yy-c1)/m1;  rb2.yy=rb1.yy;  rb2.xx=rb1.xx-100;  draw\_new\_road(lt2,rb2,tw3,650,10);  rdtpy=lt2.yy;  tw3+=10;  Y2+=10;  lt3.yy=y2;  lt3.xx=(lt3.yy-c1)/m1;  rb3.yy=lt2.yy-70;  rb3.xx=lt2.xx+tw3;  draw\_new\_road(lt3,rb3,tw4,tw3,9);  obstacleV2(rb3.xx-tw3,rb3.yy,tw3,hh,4);  }  else  draw\_new\_road(lt2,rb2,tw2,tw1,10);  drawplane(0,0);  draw\_new\_road(lt1,rb1,tw1,762,7);  drawplane(0,0);  obstpy=-inf;  if(Y1<=750)  rdtpy=Y1;  obstpy=-inf;  if(Y1>1000 && Y1<1600)  {  obstacleV2(O1X,O1Y,L1,height,5);  O1X-=5;  O1Y+=10;  L1+=5;  height+=4;  }  if(Y1>1700)  {  obstacleV2(O3X,O3Y,L3,height-270,6);  O3X-=5;  O3Y+=10;  L3+=3;  height+=2;  }  if(Y1>1500 && O2Y<700)  {  obstacleV2(O2X,O2Y,L2,height-200,12);  O2X-=1;  O2Y+=10;  L2+=4;  height+=2;  }  if(Y1>3400)  {  obstacleV2(O4X,O4Y,L4,H4,4);  O4X-=1;  O4Y+=10;  L4+=5;  H4+=3;  }  if(Y1>3900)  {  obstacleV2(O5X,O5Y,L5,H5,13);  O5X-=3;  O5Y+=10;  L5+=6;  H5+=3;  }  drawplane(0,0);  if(Game\_over())  {  OVER();  return;  }  if(cnt==15)  {  SECONDS++;  cnt=0;  }  if(FINISH==1)  return;  if(Game\_over())  {  OVER();  return;  }  if(FINISH==1)  return;  readimagefile(“space-sunrise.jpg”,0,0,dim1,dim2);  drawplane(0,0);  }  //this for loop is for taking the plane downwards.  // comments above are also applicable here  for(q=-410; q<=0; q+=45,cnt++,Y1+=10,tw1+=10)  {  drawplane(0,-30);  lt1.yy=Y1;  lt1.xx=(lt1.yy-c1)/m1;  rb2.yy=lt1.yy-70;  rb2.xx=lt1.xx+tw1;  if(Y1>2600)  {  lt2.yy=Y2;  lt2.xx=(lt2.yy-c1)/m1;  rb2.yy=rb1.yy;  rb2.xx=rb1.xx-100;  draw\_new\_road(lt2,rb2,tw3,650,10);  tw3+=10;  Y2+=10;  lt3.yy=y2;  lt3.xx=(lt3.yy-c1)/m1;  rb3.yy=lt2.yy-70;  rb3.xx=lt2.xx+tw3;  draw\_new\_road(lt3,rb3,tw4,tw3,9);  obstacleV2(rb3.xx-tw3,rb3.yy,tw3,hh,4);  }  else  {  draw\_new\_road(lt2,rb2,tw2,tw1,10);  rdtpy=0;  }  drawplane(0,0);  draw\_new\_road(lt1,rb1,tw1,762,7);  drawplane(0,0);  if(Y1<=750)  rdtpy=Y1;  else  rdtpy=0;  obstpy=-inf;  if(Y1>1000 && Y1<1600)  {  obstacleV2(O1X,O1Y,L1,height,5);  O1X-=5;  O1Y+=10;  L1+=5;  height+=4;  }  if(Y1>1700)  {  obstacleV2(O3X,O3Y,L3,height-270,6);  O3X-=5;  O3Y+=10;  L3+=3;  height+=2;  }  if(Y1>1500 && O2Y<700)  {  obstacleV2(O2X,O2Y,L2,height-200,12);  O2X-=1;  O2Y+=10;  L2+=4;  height+=2;  }  if(Y1>3400)  {  obstacleV2(O4X,O4Y,L4,H4,4);  O4X-=1;  O4Y+=10;  L4+=5;  H4+=3;  }  if(Y1>3900)  {  obstacleV2(O5X,O5Y,L5,H5,13);  O5X-=3;  O5Y+=10;  L5+=6;  H5+=3;  }  drawplane(0,0);  if(Game\_over())  {  OVER();  return;  }  if(cnt==15)  {  SECONDS++;  cnt=0;  }  if(FINISH==1)  return;  readimagefile(“space-sunrise.jpg”,0,0,dim1,dim2);  drawplane(0,0);  }  }  int main()  {  int command,present=0,temporary;  initwindow(dim1, dim2, “First Sample”);  /// MAIN MENU  readimagefile(“play.jpg”,0,0,dim1,dim2);  while(1)  {  RESET();  present=0;  show(present);  while(1)  {  command=getch();  if(command==13)  {  temporary=DOO(present);  if(temporary==-1)  return 0;  else if(temporary==-2)  {  present=0;  show(present);  continue;  }  break;  }  if(command!=0)  continue;  command=getch();  if(command==KEY\_UP)  present=max(present-1,0);  else if(command==KEY\_DOWN)  present=min(3,present+1);  else  continue;  show(present);  }  if(temporary==7)  {  level=1;  LEVEL(1);  }  else  {  level=2;  LEVEL(2);  }  }  return 0;  } |

Score board:

|  |
| --- |
| #include <iostream>  #include <vector>  #include "graphics.h"  #include <cstdio>  #include <string.h>  using namespace std;  int calc(string str)  {  int ret = 0, into = 1, len = str.size(), i;  for(i = len-1; i >= 0; i--)  {  ret = ret + (str[i]-'0')\*into;  into = into\*10;  }  return ret;  }  void getln(char \*str)  {  int i=1;  char ch, s[2] = "";  strcpy(str, "");  while(i<=10)  {  i++;  ch = getch();  while(!ch)  {  ch = getch();  ch = getch();  }  if(ch == 13 || ch== ' ')  {  break;  }  if(ch==8)  {  str[strlen(str)-1]=0;  setbkcolor(BLUE);  setcolor(WHITE);  settextstyle(GOTHIC\_FONT,HORIZ\_DIR,4);  outtextxy(300, 300, str);  i-=2;  continue;  }  s[0] = ch;  strcat(str, s);  setbkcolor(BLUE);  setcolor(WHITE);  settextstyle(GOTHIC\_FONT,HORIZ\_DIR,4);  outtextxy(500, 300, str);  }  return;  }  int leader(int seconds,bool lev)  {  int k;  FILE \*fp;  if(lev==0)  fp = fopen("in.txt", "r");  else  fp = fopen("in2.txt", "r");  vector <string> names[15];  string scr,ST;  char str[100];  char np[100], sp[100]; // np -> name of the player, sp -> score of the player  int nw = 1, val, pos = -1, i, j, int\_scr;  sprintf(sp,"%d",seconds);  scr=sp;  int\_scr = calc(scr);  sprintf(sp,"Your timing is %d seconds",seconds);  setbkcolor(BLUE);  setcolor(WHITE);  settextstyle(GOTHIC\_FONT,HORIZ\_DIR,4);  outtextxy(300, 200, sp);    i=0;  while(EOF!= fscanf(fp, "%s", str) && nw <= 10 )  {  i++;  names[nw].push\_back(str);  if(i%2==0)  {  val = calc(str);  if(val > int\_scr && pos == -1)  {  pos = nw;  names[nw+1] = names[nw];  names[nw].clear();  nw++;  }  nw++;  }  }  fclose(fp);    if(nw <= 10 && pos == -1) pos = nw;  cout <<"pos=" << pos << endl;  if(pos == -1)  {  delay(500);  getch();  getch();  return 0;  }  delay(500);  outtextxy(100, 300, "Enter Your Name: ");  getln(np);  ST = np;  names[pos].push\_back(ST);  names[pos].push\_back(scr);  if(lev==0)  fp=fopen("in.txt", "w");  else  fp = fopen("in2.txt", "w");  for(i = 1; i <= min(nw,10); i++)  {  for(j = 0; j <= 1; j++)  {  for(k=0;k<(int)names[i][j].size();k++)  {  fprintf(fp,"%c",names[i][j][k]);  }  fprintf(fp," ");  }  fprintf(fp,"\n");  }  fclose(fp);  return 0;  } |

**Constraints:**

The game is only for DOS platforms. Windows or any other GUI were not used in this game. So that the game is not so good looking. We only learned the BGI functions to develop this game. As we know BGI is not standard graphical interface and it has certain limitations. So our game has some unwanted bugs and some keyboard buffering issues.

**Augmentation:**

In future, when we get to work on this project more, we will increase the number of levels with more difficulties. We will also improve the graphical representations and make the game more attractive and try to eliminate the bugs present now.

**Conclusion:**

The one thing that will make our project popular to others is that, there is hardly any person who hasn’t played this game in their childhood. Also it is an interesting game that will give the players a taste of futuristic racing game. The most interesting fact about our game is it will make you feel nostalgic and feel like you tried something new at the same time. To sum up, we are hopeful that if we learn more work on this project, we can make it better.

**References:**

1. Wikipedia.
2. [www.bluemoon.ee/history/**skyroads**/](http://www.bluemoon.ee/history/skyroads/)
3. [www.cs.colorado.edu/~main/**bgi**/doc/](http://www.cs.colorado.edu/~main/bgi/doc/)
4. www.cplusplus.com.