VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELGAUM-590014

MINI PROJECT ENTITLED

"SUPER MARIO GAME"

For the Academic Year 2021-2022

Submitted by:

PABBATHI JAHNAVI 1MV19CS080
RAYANNAGARI SREEHARI 1MV19CS091

Project carried out at

Sir M Visvesvaraya Institute of Technology

Bangalore-562157

Under the guidance of:
Mr Suraj Kumar B P
Assistant Professor, Department of CSE
Sir M. Visvesvaraya Institute of Technology, Bengaluru

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SIR M. VISVESVARAYA INSTITUTE OF TECHNOLOGY
HUNASAMARANAHALLI, BENGALURU-56215

SIR M. VISVESVARAYA INSTITUTE OF TECHNOLOGY

Krishnadevaraya Nagar, International Airport Road,
Hunasamaranahalli, Bengaluru – 562157

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

It is certified that the **Computer Graphics Mini Project work** entitled " **SUPER MARIO GAME**" is carried out by PABBATHI JAHNAVI (1MV19CS080), RAYANNAGARI SREEHARI (1MV19CS091) bonafide students of **Sir M Visvesvaraya Institute of Technology** in partial fulfilment for the 6th semester for the award of the Degree of Bachelor of Engineering in Computer Science and Engineering of the **Visvesvaraya Technological University, Belagavi** during the academic year **2021-2022**.It is certified that all corrections and suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the course of Bachelor of Engineering.

Name & Signature	Name & Signature	Name & Signature
Mr Suraj Kumar B P	Dr. G. C. Bhanu Prakash	Dr. V.R. Manjunath
Assistant Professor	HOD, Dept. Of CSE,	Principal,
Dept. Of CSE, Sir MVIT	Sir MVIT	Sir MVIT
Bengaluru - 562157	Bengaluru - 562157	Bengaluru – 562157

External Examination:

Name of Examiner Signature with Date

1)

2)

DECLARATION

We hereby declare that the entire project work embodied in this dissertation has been carried out by us and no part has been submitted for any degree or diploma of any institution previously.

Place: Bengaluru

Date:

Signature of Students:

PABBATHI JAHNAVI	RAYANNAGARI SREEHARI
(1MV19CS080)	(1MV19CS091)

ACKNOWLEDGMENT

It gives us immense pleasure to express our sincere gratitude to the management of **Sir M. Visvesvaraya Institute of Technology,** Bengaluru for providing the opportunity and the resources to accomplish our project work in their premises. On the path of learning, the presence of an experienced guide is indispensable and we would like to thank our guide **Mr Suraj Kumar B P, Assistant Professor**, Dept. of CSE, for his invaluable help and guidance. Heartfelt and sincere thanks to **Dr. G. C. Bhanu Prakash, HOD**, Dept. of CSE, for his suggestions, constant support and encouragement. We would also like to convey our regards to **Dr. V.R. Manjunath, Principal**, Sir. MVIT for providing us with the infrastructure and facilities needed to develop our project. We would also like to thank the staff of Department of Computer Science and Engineering and lab-in-charges for their co-operation and suggestions. Finally, we would like to thank all our friends for their help and suggestions without which completing this project would not have been possible.

ABSTRACT

This project is written in C and used OpenGL (Open Graphics Library). Open Graphics Library is a cross-language, cross-platform application programming interface for rendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit, to achieve hardware-accelerated rendering.

Computer games are **programs that enable a player to interact with a virtual game environment for entertainment and fun**. There are many types of computer games available, ranging from traditional card games to more advanced video games such as role-playing games and adventure games.

This project is one of such implementations of computer graphics game named Super Mario. This project demonstrates how a Mario crosses all the obstacles to reach a finish line. Mario gains 10 points on crossing each obstacle

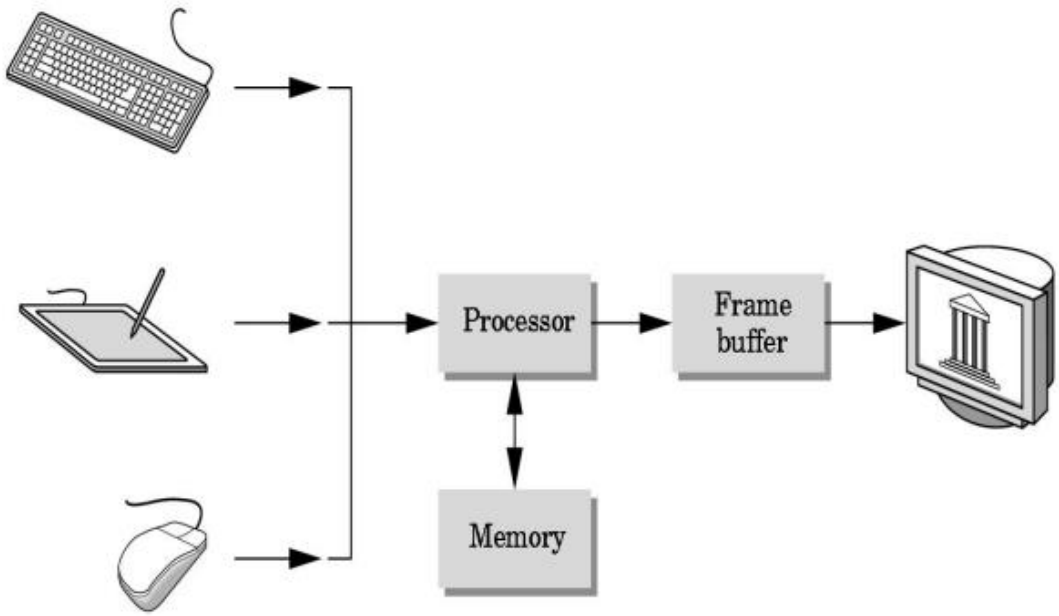
TABLE OF CONTENTS

LIST OF FIGURES

1.INTRODUCTION

1.1 Computer Graphics:

- Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D Or 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly.
- Computers have become a powerful medium for the rapid and economical production of pictures. Graphics provide a natural means of communicating with the computer that they have become widespread.
- Interactive graphics have been the most important means of producing pictures since the invention of photography and television.
- We can make pictures of not only the real-world objects but also of abstract objects such as mathematical surfaces on 4D and of data that have no inherent geometry.
- A computer graphics system is a computer system with all the components of the general-purpose computer system.
- There are five major elements in system: input devices, processor, memory, frame buffer, output devices.



Computer Graphics system

1.2 Aim:

The aim of this project is to implement Super Mario game. When the user starts the game, Mario starts running in forward direction in a path where few obstacles are placed. User has to interact with the keyboard and help Mario in crossing the obstacles and reach the final line. The interface should be user friendly and should use keyboard interface for the interaction with the user. The main goal is to implement an openGL game using c++.

1.3 Introduction to OpenGL:

OpenGL (**Open Graphics Library**) is a cross language, cross platform Application programming Interface (API) for rendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit (GPU), to achieve hardware-accelerated rendering.

OpenGL was introduced by Silicon Graphics Inc in 1992 and was used in various fields that required complex visualization of 2D and 3D images such as CAD, virtual Reality, flight simulators, and scientific visualization. It was also very obvious to enthusiasts that OpenGL could be used to render graphics in video games as well.

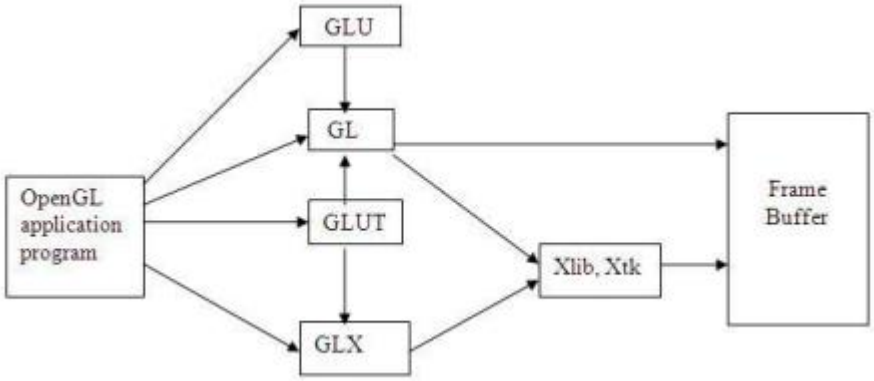
OpenGL is an open specification for an applications program interface for defining 2D and 3D objects. The specification is cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. It renders 3D objects to the screen, providing the same set of instructions on different computers and graphics adapters. Thus it allows us to write an application that can create the same effects in any operating system using any OpenGL-adhering graphics adapter.

Computer graphics, a 3-dimensional primitive can be anything from a single point to an n sided polygon. From the software standpoint, primitives utilize the basic 3dimensional rasterization algorithms such as Bresenham's line drawing algorithm, polygon scan line fill, texture mapping and so forth. OpenGL's basic operation is to accept primitives such as points, lines and polygons, and convert them into pixels. This is done by a graphics pipeline known as the OpenGL state machine. Most OpenGL commands either issue primitives to the graphics pipeline, or configure how the pipeline processes these primitives.

OpenGL is a low-level, procedural API, requiring the programmer to dictate the exact steps required to render a scene. OpenGL's low-level design requires programmers to have a good knowledge of the graphics pipeline, but also gives a certain amount of freedom to implement novel rendering algorithms.

GLUT:

GLUT, short for OpenGL Utility Toolkit, is a set of support libraries available on every major platform. OpenGL does not directly support any form of windowing, menus, or input. That's where GLUT comes in. It provides basic functionality in all of those areas, while remaining platform independent, so that you can easily move GLUT-based applications from, for example, Windows to UNIX with few, if any, changes.



OpenGL Interface

1.4 Project Related Concepts:

In the original *Super Mario Bros.*, solo players pilot Mario, and an additional player can play as Luigi. The game is based on a series of side-scrolling levels, each filled with enemies ranging from mushroom like Goombas to evil turtles known as Koopa Troopas. The levels take place in different settings, some in dungeons and some above ground, with fights against Bowser impersonators at the end of castle levels. Once the imposter is defeated, a Mushroom Kingdom resident informs Mario or Luigi that the princess is in another castle. The game is completed with the defeat of the true Bowser and the rescue of Princess Toadstool.

1.5 Interfaces:

Keyboard Interface:

Two functionalities are implemented using the keyboard function

- Game starts if the user presses the key ‘s’.
- Mario moves up if the user presses the key ‘u’.
- Mario relives again after pressing the key ‘l’.
- Mario stops the game if user presses the key ‘b’.

2. REQUIREMENT SPECIFICATION

Code::Blocks is a free, open-source cross-platform IDE that supports multiple compilers including GCC, Clang and Visual C++. It is developed in C++ using wxWidgets as the GUI toolkit. Using a plugin architecture, its capabilities and features are defined by the provided plugins.

2.1 Purpose of the requirements document

The software requirement specification is the official statement of what is required for development of a particular project. It includes both user requirements and system requirements. This requirement document is utilized by a variety of users starting from the project manager who gives the project to the engineer responsible for development of the project. It should give details of how to maintain, test, verify and what all the actions to be carried out through the life cycle of the project.

2.2 Specific Requirements:

HARDWARE REQUIREMENTS

The hardware requirements are very minimal and the software can run on most of the machines.

- Disk Space : 32 GB or more
- Processor : 1.4 GHz 64 bit or Pentium , Pentium iv , core i3 and above
- Memory : 512 MB,4GB Ram,10 GB HDD Free Space
- Display : (800*600)Capable video adapter and monitor

SYSTEM REQUIREMENTS

- Operating System : windows
- Programming Language : C/C++ using OpenGL
- Compiler : GCC Compiler
- Graphics Library : GL /glut.h

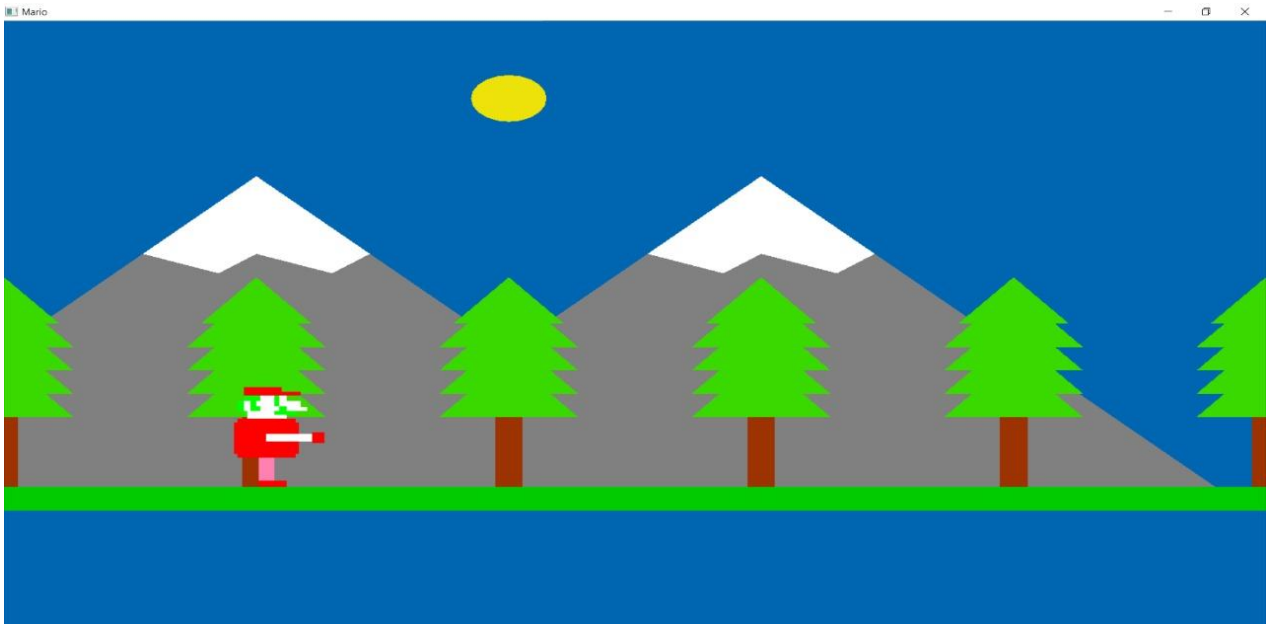
3.SYSTEM DESIGN

3.1 Window Design:

Super Mario game uses only one window, that is Mario(main window). This window contains 2 mountains, Mario, sun , cloud, trees , grass, Rock(obstacles). This is the window used for all events and functions used in this project. All the window events are also triggered in this window. Score secured by the user is also displayed on this window only.

Game display:

As soon as the user clicks the key ‘S’ Mario starts moving. When the user clicks the key ‘U’ Mario moves up.



3.2 Implementation:

Functions:

GL_POLYGON :

Draws a single, convex polygon. Vertices 1 through N define this polygon. Only a subset of GL commands can be used between glBegin and glEnd.

GL_LINE_STRIP :

The adjacent vertices are considered lines. Thus, if you pass *n* vertices, you will get *n*-1 lines. If the user only specifies 1 vertex, the drawing command is ignored.

GL_TRIANGLE_FAN :

The first vertex is always held fixed. From there on, every group of 2 adjacent vertices form a triangle with the first. So with a vertex stream, you get a list of triangles like so: (0, 1, 2) (0, 2, 3), (0, 3, 4), etc. A vertex stream of n length will generate n-2 triangles.

Basic Functions:

glRasterPos2f:

It is used to position pixel and bitmap write operations. specify the raster position for pixel operations

SYNTAX: void glRasterPos2f(GLfloat x,GLfloat y);

PARAMETERS: Object coordinates for the current raster position.

Pushmatrix():

glPushMatrix pushes the current matrix stack down by one, duplicating the current matrix. That is, after a glPushMatrix call, the matrix on top of the stack is identical to the one below it.

SYNTAX: void glPushMatrix(void);

Popmatrix():

glPopMatrix pops the current matrix stack, replacing the current matrix with the one below it on the stack.

SYNTAX: void glPopMatrix(void);

glutInitWindowSize() & glutInitWindowPosition():

glutInitWindowPosition and glutInitWindowSize set the *initial window position* and *size* respectively.

SYNTAX: void glutInitWindowSize(int width, int height);

void glutInitWindowPosition(int x, int y);

glutCreateWindow():

Description glutCreateWindow creates a top-level window. The name will be provided to the window system as the window's name. The intent is that the window system will label the window with the name. Implicitly, the current window is set to the newly created window.

glutBitmapCharacter:

It will adjust the current raster position based on the width of the character.

SYNTAX: void glutBitmapCharacter(void *font, int character);

PARAMETERS:font - Bitmap font to use., character-Character to render (not confined to 8 bits).

glBegin & glEnd Function:

The glBegin and glEnd functions delimit the vertices of a primitive or a group of like primitives.

SYNTAX: void glBegin, glEnd(GLenum mode);

PARAMETERS: mode - The primitive or primitives that will be created from vertices presented between glBegin and the subsequent glEnd. The following are accepted symbolic constants and their meanings.

Transformation Functions:

glTranslate Function:

The glTranslated and glTranslatef functions multiply the current matrix by a translation matrix.

SYNTAX: void glTranslate(x, y, z);

PARAMETERS: x, y, z - The x, y, and z coordinates of a translation vector.

glRotate Function:

glRotate produces a rotation of *angle* degrees around the vector x y z .

SYNTAX : void glRotatef(GLfloat angle,GLfloat x,GLfloat y,GLfloat z);

PARAMETERS: x, y, z - The x, y, and z coordinates of a Rotation vector.

Functions used to display:

glMatrixMode Function:

The glMatrixMode function specifies which matrix is the current matrix.

SYNTAX: void glMatrixMode(GLenum mode);

PARAMETERS: mode - The matrix stack that is the target for subsequent matrix operations. The mode parameter can assume one of three values:

Value	Meaning
GL_PROJECTION	Applies subsequent matrix operations to the projection matrix stack.

glLoadIdentity Function:

The glLoadIdentity function replaces the current matrix with the identity matrix.

SYNTAX: void glLoadIdentity(void);

Functions used to set the viewing volume:

gluOrtho2D:

This function defines orthographic viewing volume with all parameters measured from the center of projection. multiply the current matrix by a perspective matrix.

SYNTAX: void gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top) .

PARAMETERS: left, right - Specify the coordinates for the left and right vertical clipping planes. Bottom, top - Specify the coordinates for the bottom and top horizontal clipping planes.

Call back functions:

glutDisplayFunc Function:

glutDisplayFunc sets the display callback for the current window.

SYNTAX: void glutDisplayFunc(void (*func)(void));

glutKeyboardFunc(args):

glutKeyboardFunc sets the keyboard callback for the current window. When a user types into the window, each key press generating an ASCII character will generate a keyboard callback.

glutTimerFunc(args):

glutTimerFunc registers a timer callback to be triggered in a specified number of milliseconds.

SYNTAX: void glutTimerFunc(unsigned int msecs, void (*func)(int value), value);

glutMainLoop Function:

glutMainLoop enters the GLUT event processing loop.

SYNTAX: void glutMainLoop(void);

glutPostRedisplay():

glutPostRedisplay may be called within a window's display or overlay display callback to re-mark that window for redisplay.

SYNTAX : void glutPostRedisplay(void);

main function:

glutInit Function:

glutInit is used to initialize the GLUT library.

SYNTAX: glutInit(int *argcp, char **argv);

PARAMETERS:

- argcp - A pointer to the program's unmodified argc variable from main. Upon return, the value pointed to by argcp will be updated, because glutInit extracts any command line options intended for the GLUT library.
- Argv - The program's unmodified argv variable from main. Like argcp, the data for argv will be updated because glutInit extracts any command line options understood by the GLUT library. glutInit(&argc,argv);

glutInitDisplayMode Function:

glutInitDisplayMode sets the initial display mode.

SYNTAX: void glutInitDisplayMode(unsigned int mode);

PARAMETERS:

- mode - Display mode, normally the bitwise OR-ing of GLUT display mode bit masks. See values below:
  GLUT_RGB: An alias for GLUT_RGBA.

  GLUT_DOUBLE:Bit mask to select a double buffered window. This overrides GLUT_SINGLE.

  GLUT_DEPTH: Bit mask to select a window with a depth buffer.

4.SOURCE CODE

```
#include<cstdio>
#include <GL/gl.h>
#include <GL/glut.h>
#include <math.h>
#include<windows.h>
#include<time.h>
float upD=700,upS=0;
bool stat=true;
float Rep2 = 0.0f;
float x = 0.0f;
float Rep = 0.0f;
GLfloat ll=0,gol=0,jumping=0;
int flag=1,flagm=0,game=5,flagj=0,flagjj=1;
int count1= 0;
void displayRasterText(float x ,float y ,float z ,char *stringToDisplay) {
        glRasterPos3f(x, y, z);
        for(char* c = stringToDisplay; *c != '\0'; c++){
                glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24 , *c);
        }
}
void poly(int xl,int xr,int yu,int yd){
glBegin(GL_POLYGON);
glVertex2i(xl,yd);
glVertex2i(xl,yu);
glVertex2i(xr,yu);
glVertex2i(xr,yd);
glEnd();
}
//TREE--->pulok
void tree2(){
glTranslatef(-500.0f, -430.0f, 0.0f);
```

```
glScalef(220.0, 300.0, 0.0);

glBegin(GL_POLYGON);
glColor3f(0.2267f,0.85f, 0.0f);
glVertex2f(0.0f,1.0f);
glVertex2f(-0.2f,0.8f);
glVertex2f(0.2f,0.8f);
 glEnd();

glBegin(GL_POLYGON);
glColor3f(0.2267f,0.85f, 0.0f);
glVertex2f(-0.15f,0.8f);
glVertex2f(-0.25f,0.7f);
glVertex2f(0.25f,0.7f);
glVertex2f(0.15f,0.8f);
 glEnd();

 glBegin(GL_POLYGON);
glColor3f(0.2267f,0.85f, 0.0f);
glVertex2f(-0.15f,0.7f);
glVertex2f(-0.25f,0.6f);
glVertex2f(0.25f,0.6f);
glVertex2f(0.15f,0.7f);
 glEnd();

 glBegin(GL_POLYGON);
glColor3f(0.2267f,0.85f, 0.0f);
glVertex2f(-0.15f,0.6f);
glVertex2f(-0.25f,0.5f);
glVertex2f(0.25f,0.5f);
glVertex2f(0.15f,0.6f);
 glEnd();

 glBegin(GL_POLYGON);
```

```
glColor3f(0.2267f,0.85f, 0.0f);
glVertex2f(-0.15f,0.5f);
glVertex2f(-0.25f,0.4f);
glVertex2f(0.25f,0.4f);
glVertex2f(0.15f,0.5f);
 glEnd();

 glBegin(GL_POLYGON);
glColor3f(0.61f, 0.20f, 0.0f);//Brown
glVertex2f(-0.05f,0.4f);
glVertex2f(0.05f,0.4f);
glVertex2f(0.05f,0.0f);
glVertex2f(-0.05f,0.0f);
 glEnd();
}
//Mountain --->pulok
void mountain(){
glTranslatef(-500.0f, -400.0f, 0.0f);
glScalef(600.0, 500.0, 0.0);
glBegin(GL_POLYGON);
glColor3f(0.5f, 0.5f, 0.5f);//Brown
glVertex2f(-0.6f,0.0f);
glVertex2f(0.0f,0.8f);
glVertex2f(0.6f,0.0f);
glEnd();
glBegin(GL_POLYGON);
glColor3f(1.0f, 1.0f, 1.0f);//Brown
glVertex2f(0.0f,0.8f);
glVertex2f(-0.15f,0.6f);
glVertex2f(-0.05f,0.55f);
glVertex2f(0.0f,0.6f);
glVertex2f(0.1f,0.55f);
glVertex2f(0.15f,0.6f);
 glEnd();
```

```
}
//LOSE --->pulok
void LoseSign(){
if(count1==1){
   glColor3f(1.0, 0.5, 0.0);
   displayRasterText(250, 350, 0.0,"sorry you lost,your score is 10");
}
if(count1==2){
   glColor3f(1.0, 0.5, 0.0);
   displayRasterText(250, 350, 0.0,"sorry you lost,your score is 20");
}
if(count1==3){
   glColor3f(1.0, 0.5, 0.0);
   displayRasterText(250, 350, 0.0,"sorry you lost,your score is 30");
}
if(count1==4){
   glColor3f(1.0, 0.5, 0.0);
   displayRasterText(250, 350, 0.0,"sorry you lost,your score is 40");
}
}
void treeSet(){
float x;
while(x<5000){
glPushMatrix();
glTranslatef(-200.0f+x, 200.0f, 0.0f);
tree2();

glPopMatrix();
x+=200.0f;
}
}
void mario(){
glPushMatrix();
glTranslatef(0, jumping, 0);
```

```
glRotatef(gol, 0.0f, 0.0f, 1.0f);
glTranslatef(0, 13, 0);
glColor3f(1.0f, 0.0f, 0.0f);//hat
poly(-310,-280,-85,-95);
poly(-280,-265,-90,-95);
glColor3f(1.0f, 1.0f, 1.0f);//face
poly(-307,-276,-95,-125);
poly(-310,-265,-102,-115);
poly(-265,-260,-110,-115);
glColor3f(0.0f, 1.0f, 0.0f);//hair
poly(-310,-297,-95,-102);
poly(-305,-300,-102,-115);
poly(-300,-297,-109,-115);
poly(-315,-310,-102,-115);
poly(-310,-307,-115,-120);
poly(-286,-282,-95,-109);
poly(-282,-276,-109,-115);
poly(-286,-268,-115,-120);
int x=-292,y=-165;
glPushMatrix();
glTranslatef(x, y, 0);
glRotatef(ll, 0.0f, 0.0f, 1.0f);
glTranslatef(-x, -y, 0);
glColor3f(1.0f, 1.0f, 1.0f);//leg
poly((x-6),(x+6),(y),(y-40));glColor3f(1.0f, 0.0f, 0.0f);poly((x-6),(x+16),(y-40),(y-50));
glPopMatrix();
glPushMatrix();
glTranslatef(x, y, 0);
glRotatef(-ll, 0.0f, 0.0f, 1.0f);
glTranslatef(-x, -y, 0);
glColor3f(1.0f, 0.50f, 0.7f);//leg
poly((x-6),(x+6),(y),(y-40));glColor3f(1.0f, 0.0f, 0.0f);poly((x-6),(x+16),(y-40),(y-50));
glPopMatrix();
```

```
glColor3f(1.0f, 0.0f, 0.0f);//body
poly(-315,-269,-125,-130);
poly(-318,-266,-130,-170);
poly(-315,-269,-170,-175);
int xh=-292,yh=-155;
glPushMatrix();
glTranslatef(xh, yh, 0);
glRotatef(ll, 0.0f, 0.0f, 1.0f);
glTranslatef(-xh, -yh, 0);
glColor3f(1.0f, 1.0f, 1.0f);//hand
poly((xh),(xh+36),(yh),(yh+10));glColor3f(1.0f, 0.0f, 0.0f);poly((xh+36),(xh+46),(yh-2),(yh+12));
glPopMatrix();
glPopMatrix();
}
void WinSign(){
glColor3f(1.0, 0.0, 0.0);
displayRasterText(250, 350, 0.0,"Congratulations,well done!!!!");
displayRasterText(100, 350, 0.0,"your score is 50");
glBegin(GL_LINE_STRIP);//W
glColor3f(0.30f, 0.0f, 0.70f);
glVertex2f(-180.0f ,100.0f);
glVertex2f(-150.0f ,-100.0f);
glVertex2f(-100.0f ,100.0f);
glVertex2f(-50.0f ,-100.0f);
glVertex2f(0.0f ,100.0f);
glEnd();

glBegin(GL_LINE_STRIP);//I
glColor3f(0.30f, 0.0f, 0.70f);
glVertex2f(50.0f ,100.0f);
glVertex2f(50.0f ,-100.0f);
glEnd();

glBegin(GL_LINE_STRIP);//N
```

```
glColor3f(0.30f, 0.0f, 0.70f);
glVertex2f(110.0f ,-100.0f);
glVertex2f(110.0f ,100.0f);
glVertex2f(200.0f ,-100.0f);
glVertex2f(200.0f ,100.0f);
glEnd();
}

void FinishLine(){
glTranslatef(650.0f, 0.0f, 0.0f);
glBegin(GL_POLYGON);
glColor3f(0.80f, 0.20f, 0.20f);
glVertex2f(400.0f ,-200.0f);
glVertex2f(410.0f ,-200.0f);
glVertex2f(410.0f ,200.0f);
glVertex2f(400.0f ,200.0f);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.20f, 0.80f, 0.20f);
glVertex2f(410.0f ,180.0f);
glVertex2f(410.0f ,120.0f);
glVertex2f(490.0f ,150.0f);
glEnd();
}

void Repeater(float position){
glTranslatef(position,0.0f, 0.0f);
glColor3f(0.0f, 0.5f, 1.0f);

 glBegin(GL_POLYGON);// grass surface layer
glColor3f(0.0f, 0.80f, 0.0f);
glVertex2f(-500.0f ,-230.0f);
glVertex2f( -200.0f, -230.0f);
```

```
glVertex2f( -200.0f, -200.0f);
glVertex2f( -500.0f, -200.0f);
glEnd();
}
void Rock(float x)
{
glBegin(GL_POLYGON);// big rock
glColor3f(0.8f, 0.8f, 0.8f);
glVertex2f(100.0f+x ,-200.0f);
glVertex2f(210.0f+x, -200.0f);
glVertex2f(210.0f+x, -80.0f);
glVertex2f(180.0f+x , -80.0f);
glVertex2f(120.0f+x, -150.0f);
glEnd();

 glBegin(GL_POLYGON);//shade
glColor3f(0.6f, 0.6f, 0.6f);
glVertex2f(170.0f+x ,-200.0f);
glVertex2f(210.0f+x, -200.0f);
glVertex2f(210.0f+x, -80.0f);
glVertex2f(200.0f+x , -100.0f);
glVertex2f(180.0f+x, -150.0f);
glEnd();

 glBegin(GL_POLYGON);//small rock
glColor3f(0.8f, 0.8f, 0.8f);
glVertex2f(200.0f+x ,-200.0f);
glVertex2f(230.0f+x, -200.0f);
glVertex2f(230.0f+x, -180.0f);
glVertex2f(220.0f+x, -170.0f);
glVertex2f(200.0f+x , -180.0f);
glEnd();
}
void delay(unsigned int ms){
```

```
clock_t goal= ms + clock();
while(goal>clock());
stat=!stat;
}
void drawFilledCircle(GLfloat x, GLfloat y, GLfloat radius){
int i;
int triangleAmount = 20; //# of triangles used to draw circle
 //GLfloat radius = 0.8f; //radius
GLfloat twicePi = 2.0f * 3.1416;
 glBegin(GL_TRIANGLE_FAN);
glVertex2f(x, y); // center of circle
for(i = 0; i <= triangleAmount;i++) {
glVertex2f(
x + (radius * cos(i * twicePi / triangleAmount)),
y + (radius * sin(i * twicePi / triangleAmount))
);
}
glEnd();
}
void cloud(){
glTranslatef(50.0f, 0.0f, 0.0f);
glColor3ub(255, 255, 255);
drawFilledCircle(600,270,40);
drawFilledCircle(640,270,30);
drawFilledCircle(570,290,30);
drawFilledCircle(570,250,30);
drawFilledCircle(550,270,30);
drawFilledCircle(520,260,40);
drawFilledCircle(1200,270,40);
drawFilledCircle(1240,270,30);
drawFilledCircle(1170,290,30);
drawFilledCircle(1170,250,30);
drawFilledCircle(1150,270,30);
drawFilledCircle(1120,260,40);
```

```
}
void display(){
glClearColor(0.0f, 0.4f, 0.7f, 0.0f);
glClear(GL_COLOR_BUFFER_BIT);
glPushMatrix();
glTranslatef(0,upS,0);
glColor3ub(237, 226, 9);
drawFilledCircle(-100,300,30);
glPopMatrix();
glPushMatrix();
glTranslatef(0,upD,0);
glColor3ub(255, 255, 255);
drawFilledCircle(100,300,40);
glPopMatrix();
glPushMatrix();
glTranslatef(600.0f, 200.0f, 0.0f);
mountain();
glPopMatrix();
glPushMatrix();
glTranslatef(200.0f, 200.0f, 0.0f);
mountain();
glPopMatrix();
glPushMatrix();
glPushMatrix();
glTranslatef(Rep,0.0f, 0.0f);
treeSet();
glPopMatrix();
mario();
glTranslatef(Rep,0.0f, 0.0f);
glPushMatrix();
if(count1==4){
FinishLine();
}
glPopMatrix();
```

```
Rock(600);
Repeater(0.0f);
for(int i=0; i<10;i++)
        Repeater(300.0f);
glPopMatrix();
glPushMatrix();
glTranslatef(Rep2, 0.0f, 0.0);
cloud();
glPopMatrix();
if(count1==5){
 WinSign();
}
if(game>=1 && game !=5){
LoseSign();
}
glFlush();
}
int flag2=0;
void update(int value){
if(count1<5){
if(game==0){
if(upS==0&&upD==200){delay(2000);}
else if(upS==-800&&upD==0){
delay(3000);
}
if(upS>-800&&flag2==0&&stat){
if(upS==-799){flag2=1;}
upS--;
}
if(upD>0&&flag2==1&&stat){
if(upD==1){flag2=2;}
upD--;
}
if(upD<200&&flag2==2&&!stat){
```

```
if(upD==199){flag2=3;}
upD++;
}
if(upS<0&&flag2==3&&!stat){
if(upS==-1){flag2=0;}
upS++;
}
// for mario movement
if(ll==30){flag=-1;}
else if(ll==-30){flag=1;}
if (flag==-1){ll=ll-5;}
else if (flag==1){ll=ll+5;}
if(flagj==1){
if(jumping==160){flagjj=-1;}
else if(jumping==0){flagjj=1;}
if (flagjj==-1){jumping=jumping-10;
if(jumping==0){flagj=0;}
else if (flagjj==1){jumping=jumping+20;}
}
// for ground movement
if(Rep <-1300.0f)
Rep = 0.0f;
if(Rep == 0.0f){
count1++;
}
Rep -= 20;
if(Rep==-980&&flagj==0){game=1;}
if(Rep>=-1100&&Rep<=-1080&&jumping<120&&flagj==1){game=1;}
if(Rep2 <-1800.0f)
Rep2=0.0f;
Rep2 -= 10;
glutPostRedisplay();
}
}
```

```
glutTimerFunc(30, update, 0); //speed
}
void init(void){
glClearColor (1.0, 1.0, 1.0, 0.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(-500.0, 500.0, -400.0, 400.0, -1.0, 1.0);
}
void handleKeypress(unsigned char key, int x, int y){
switch (key){
case 'u':
flagj=1;
break;
 case 's':
game = 0;
break;
 case 'b':
game = 5;
break;
 }
}
int main(int argc, char** argv){
glutInit(&argc, argv);
glutInitWindowSize(640, 560);
glutInitWindowPosition(0, 0);
glutCreateWindow("Mario");
init();
glLineWidth(50);
glutDisplayFunc(display);
glutTimerFunc(10, update, 0);
glutKeyboardFunc(handleKeypress);
glutMainLoop();
return 0;
}
```

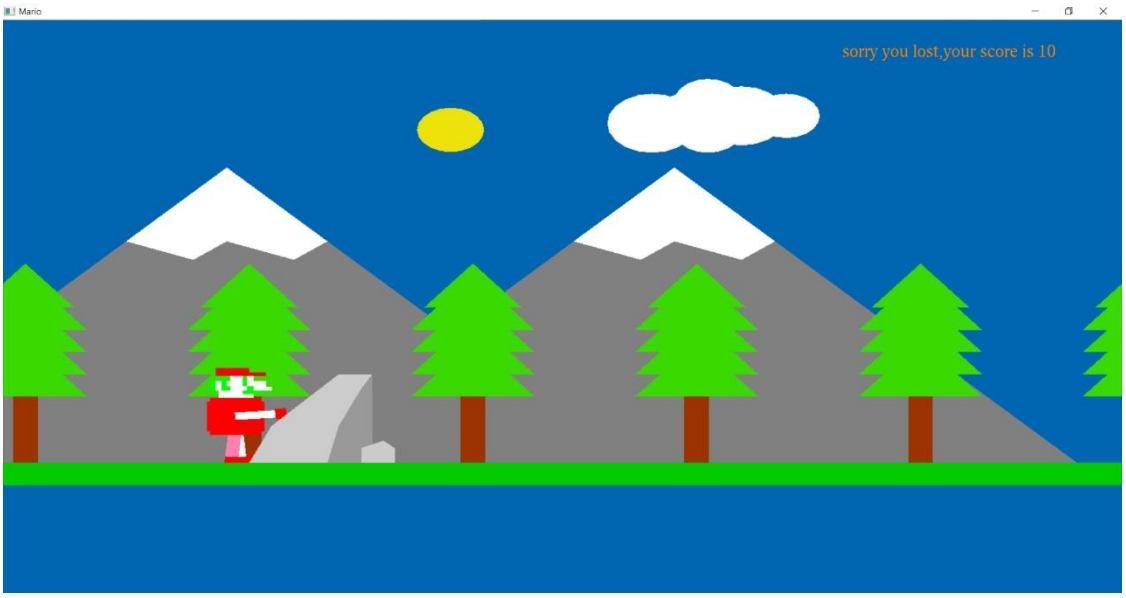
5.TESTING

S.NO	Functionality	Comments	Remarks
1	Simulate	When key ‘s’ is pressed, Mario starts Moving.	pass
2	Stop	When key ‘b’ is pressed, Mario stops moving.	pass
3	Jump upwards	When key ‘u’ is pressed, Mario moves upwards i.e., Mario jumps.	pass
4	Loose	When Mario hits an obstacle, game is lost.	pass
5	Win	If Mario crosses all the obstacles, then the player wins.	pass
6	Lives	When key ‘l’ is pressed, Mario lives again even if it hits an obstacle.	pass

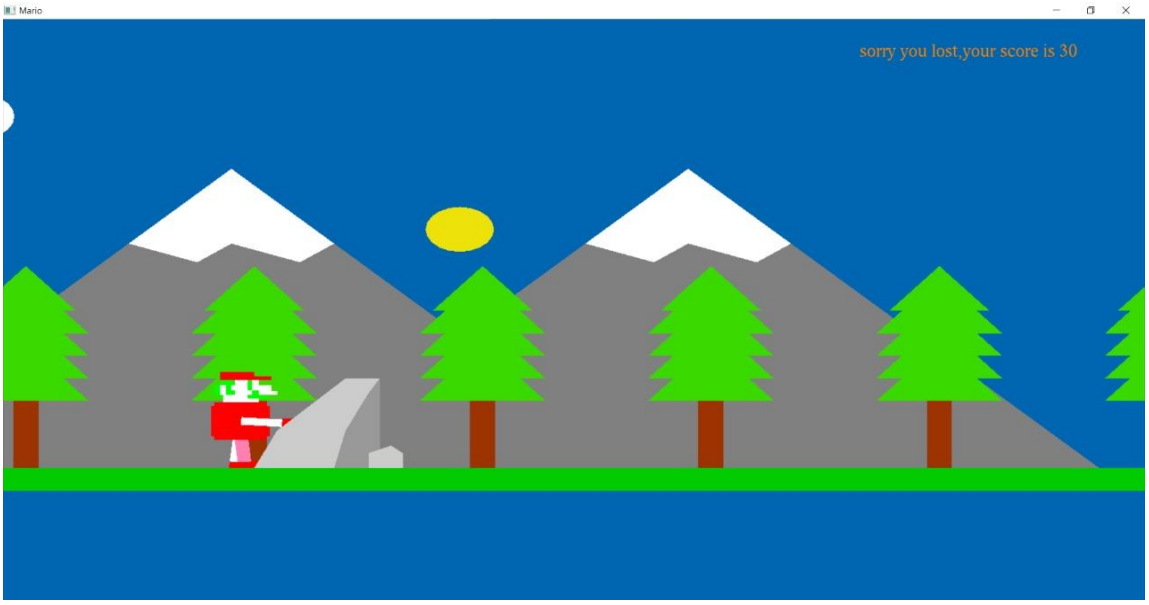
Table: Test Cases for keyboard Interface

6. SNAPSHOTS


Mario


Mario
sorry you lost,your score is 10


Mario
sorry you lost,your score is 20


Mario
sorry you lost,your score is 30


Mario
sorry you lost,your score is 40


Mario
your score is 50
Congratulations,well done!!!!
WIN

6.CONCLUSION

The project started with the designing phase in which we figured the requirements needed, the layout design, then came the detailed designing of each function after which, was the testing and debugging stage. We have tried to implement the project making it as user-friendly and error free as possible. We regret any errors that may have inadvertently crept in.

7.FUTURE ENHANCEMENTS

The following are some features that are planned to be supported in the future versions of SUPER MARIO GAME:

- Adding more levels with increased difficulty
- Adding different kind of Obstacles.
- Adding more outfits to Super Mario.

8.REFERENCES

- Interactive Computer Graphics: A Top-Down Approach with OpenGL -Edward Angel, 5th Edition, Addison-Wesley, 2008.
- Computer Graphics Using OpenGL – F.S. Hill Jr. 2nd Edition, Pearson Education, 2001.
- Computer Graphics – James D Foley, Andries Van Dam, Steven K Feiner, John F Hughes, Addison-Wesley 1997.
- www.google.com
- www.wikepedia.com
- www.vtucs.com
- www.github.com