VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgaum-590018

A Computer Graphics Mini Project Report
on

"JUMPING DINOSAUR GAME"

Submitted in Partial fulfillment of the Requirements for the VI Semester of the Degree of

Bachelor of Engineering
In
Computer Science & Engineering
By

C M ARJUN
(1CR16CS035)

R SHIV NARAYAN REDDY
(1CR16CS122)

Under the Guidance of

Kartheek G C R
Assistant Professor, Dept. of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI,

BANGALORE-560037

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI,

BANGALORE-560037

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Computer Graphics Project work entitled **“Jumping Dinosaur Game”** has been carried out by **R SHIV NARAYAN REDDY(1CR16CS122)** and **C M ARJUN (1CR16CS035)** bonafide students of CMR Institute of Technology in partial fulfillment for the award of **Bachelor of Engineering** in **Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year **2018-2019**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. This Computer Graphics Project Report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

-----------------------	**----------------------**
Signature of Guide	**Signature of HOD**
Kartheek G C R **Assistant Professor** **Dept. of CSE, CMRIT** **CMRIT**	**Mrs. Jhansi Rani** **Professor, Head** **Dept. of CSE,**

External Viva

Name of the examiners	Signature with date
1.	
2.	

ABSTRACT

OpenGL provides a set of commands to render a three dimensional scene. That means you provide them in an Open GL-useable form and Open GL will show this data on the screen(render it). It is developed by many companies and it is free to use. You can develop Open GL- applications without licensing.

Open GL is a hardware- and system dependent interface. An Open GL-application will work on every platform, as long as there is an installed implementation, because it is system independent, there are no functions to create windows etc., but there are helper functions for each platform. A very useful thing is GLUT.

We recreated the popular Android game, Jumping Dinosaur, for the partial fulfillment of credits towards our Computer Graphics Laboratory. The game is however a desktop version. Our primary aim is to implement the entire game using OpenGL libraries and functions, and build the game from scratch. At last, it would be a comprehensible achievement, if this project reaches the hands of others who can not only play this game, but also make modifications and take it to greater heights.

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany a successful completion of any task would be incomplete without the mention of people who made it possible, success is the epitome of hard work and perseverance, but steadfast of all is encouraging guidance.

So with gratitude I acknowledge all those whose guidance and encouragement served as beacon of light and crowned our effort with success.

I would like to thank **Dr. Sanjay Jain,** Principal, CMRIT, Bangalore for providing excellent academic environment in the college and his never-ending support for the B.E program.

I would also like to thank **Mrs. Jhansi Rani,** Professor and HOD, Department of Computer Science and Engineering, CMRIT, Bangalore who shared her opinions and experiences through which I received the required information crucial for the project.

I consider it a privilege and honour to express my sincere gratitude to my internal guide **Shivraj V B,** Asst. Professor, Department of Computer Science & Engineering, CMRIT, Bangalore for their valuable guidance throughout the tenure of this project work.

I would also like to thank all the faculty members who have always been very Co-operative and generous. Conclusively, I also thank all the non-teaching staff and all others who have done immense help directly or indirectly during our project.

C M ARJUN
R SHIV NARAYAN REDDY

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The main Agenda of this project was to recreate the popular Android game, Jumping Dinosaur. We created the project from scratch using OpenGL libraries and implemented the graphics required by the game. The game's algorithm was however implemented using C++.

Jumping Dinosaur, is our version of the game for PCs. The game consists of subtle yet interesting graphics. This is a single player game (offline).

The main idea of the game is however very simple, it is to taking control of the dinosaur and jumping over the obstacles that appears on the screen. A player presses the UP arrow key and grants the dinosaur to jump thereby avoiding the obstacles.

1.1 Introduction to 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 so natural means of communicating with the computer that they have become widespread.
- Interactive graphics is 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.

1.2 Areas of Application of Computer Graphics

- ➢ User interfaces and Process control
- ➢ Cartography
- ➢ Office automation and Desktop publishing
- ➢ Plotting of graphs and charts
- ➢ Computer aided Drafting and designs
- ➢Simulation and Animation

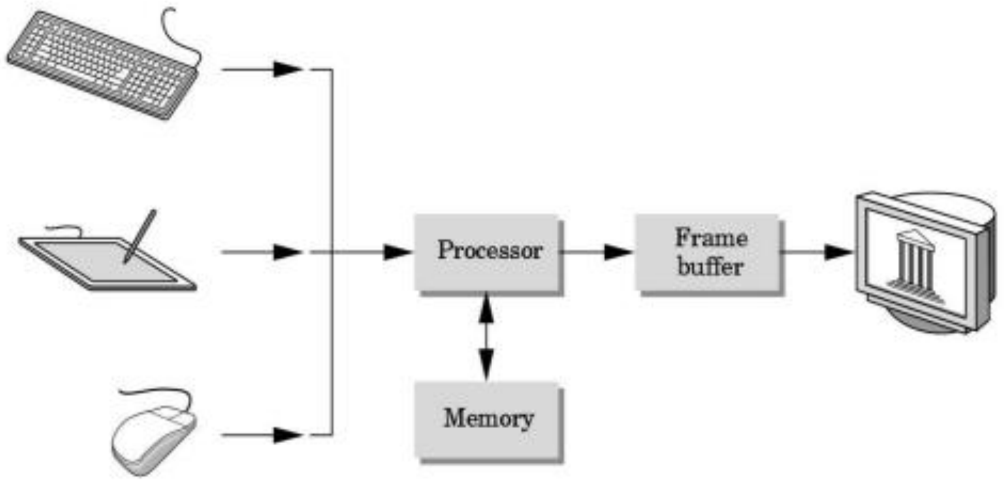


Fig 1.1: Graphic System

1.3 Introduction to OpenGL

OpenGL is the premier environment for developing portable, interactive 2D and 3D graphics applications. Since its introduction in 1992, OpenGL has become the industry's most widely used and supported 2D and 3D graphics application programming interface (API), bringing thousands of applications to a wide variety of computer platforms.

OpenGL fosters innovation and speeds application development by incorporating a broad set of rendering, texture mapping, special effects, and other powerful visualization functions. Developers can leverage the power of OpenGL across all popular desktop and workstation platforms, ensuring wide application deployment.

OpenGL available Everywhere: Supported on all UNIX® workstations, and shipped standard with every Windows 95/98/2000/NT and MacOS PC, no other graphics API operates on a wider range of hardware platforms and software environments.

OpenGL runs on every major operating system including Mac OS, OS/2, UNIX, Windows 95/98, Windows 2000, Windows NT, Linux, Open Step, and BeOS; it also works with every major windowing system, including Win32, MacOS, Presentation Manager, and X-Window System. OpenGL is callable from Ada, C, C++, Fortran, Python, Perl and Java and offers complete independence from network protocols and topologies.

Our application will be designed to access OpenGL directly through functions in three libraries namely: gl, glu, glut.

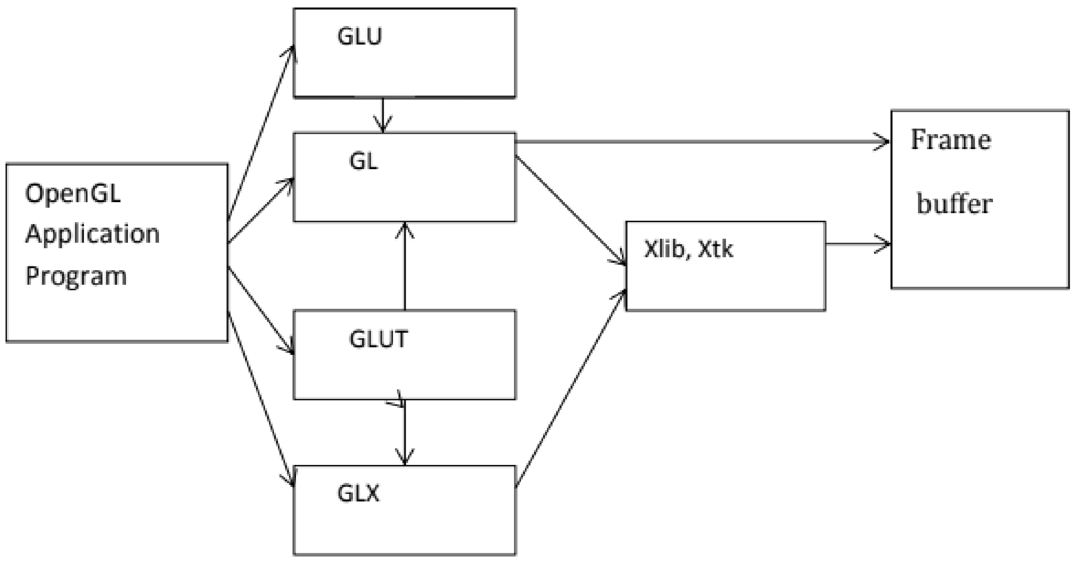


Fig 1.2: Library organization of OpenGL

1.3.1 Graphics Functions

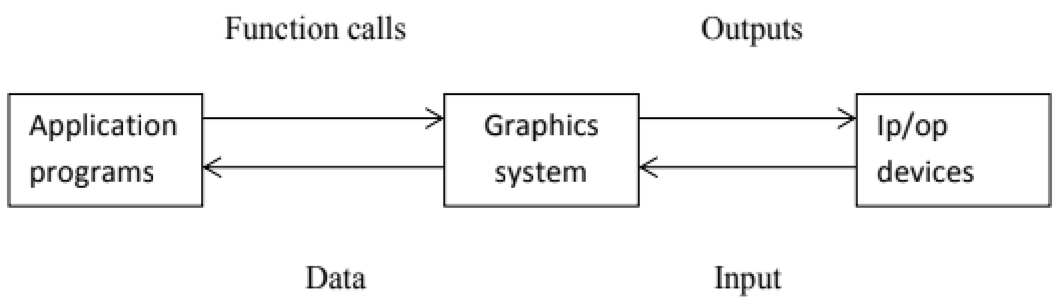


Fig 1.3: Graphics system as a black box

Our basic model of a graphics package is a black box, a term that engineers use to denote a system whose properties are described only by its inputs and outputs. We describe an API through the functions in its library. Some of the functions are:

- The primitive functions define the low-level objects or atomic entities that our system can display.
- Attribute functions allow us to perform operations ranging from choosing the color

with which we display a line segment, to picking a pattern with which to fill the inside of a polygon, to selecting a typeface for the titles of a graph.

- Transformation function allows carrying out transformations of objects, such as rotation, translation, and scaling.
- A set of input functions allow us to deal with the diverse forms of input that

characterize modern graphics systems.

- The control functions enable us to communicate with the window systems, to initialize our programs, and to deal with any errors that take place during the execution of programs.

CHAPTER 2

System Requirements

2.1 Hardware Requirements

Processors: Intel i3,i5,i7

Processor Speed: 3.00 GHZ

RAM: 4 GB

Storage: 500 GB

Monitor: 15 inches

Keyboard: Standard 102 keys

Mouse: Standard 3 buttons

2.2 Software Requirements

-- Operating System can be either Windows/Linux. The project is compatible with either of them.

-- OpenGL libraries and GLUT installed

-- C++ compiler

CHAPTER 3

DESIGN

3.1 Dinosaur Model

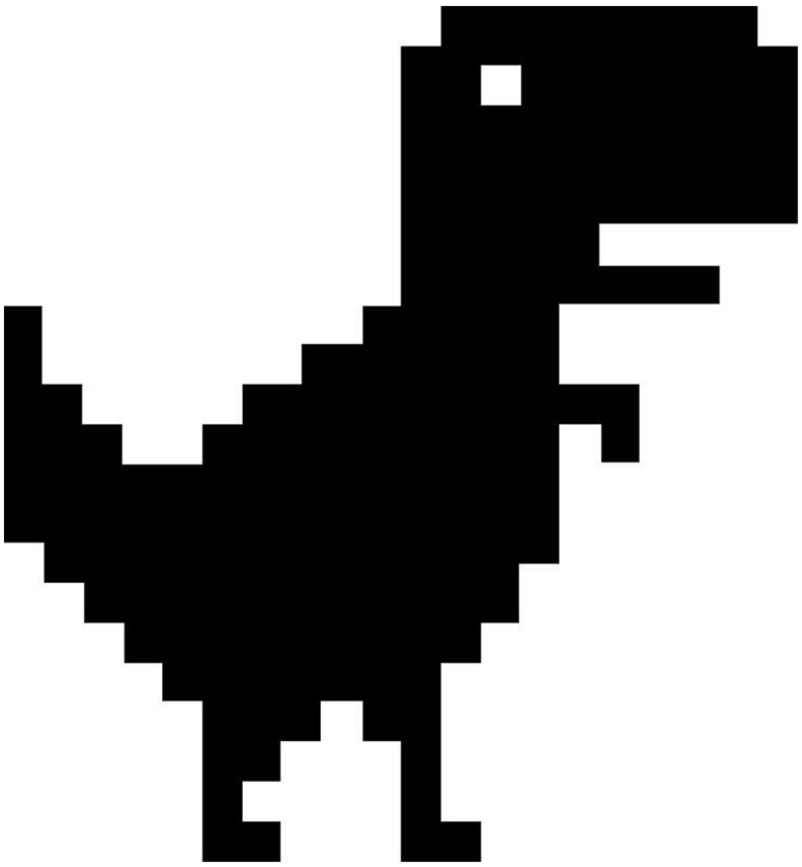


Fig 3.1: Dinosaur Model

The coordinates points for the above shown model of the dinosaur was plotted using the OpenGL primitive GL_LINES which was then scaled using OpenGL function glScaled() to display it on the screen.

CHAPTER 4

IMPLEMENTATION

We used Visual Studio for as our IDE and used OpenGL to implement the graphics needed for the game. This enabled us to make the streamlined yet subtle graphics for is appealing yet simplistic in form. The entire algorithm and implementation is written in C++.

Lastly, for interaction with the game, the user will however have to use the keyboard.

4.1 OpenGL Function Details

- **glutInitDisplayMode** — sets the initial display mode.
    - Declaration: void glutInitDisplayMode (unsigned int mode);
    - Remarks: The initial display mode is used when creating top-level windows, sub windows, and overlays to determine the OpenGL display mode for the to-be-created window or overlay.

- **glutInitWindowPosition** --- set the initial window position.
    - Declaration: void glutInitWindowPosition(int x, int y);

      x: Window X location in pixels.

      y: Window Y location in pixels.

- **glutInitWindowSize** --- set the initial window size.
    - Declaration: void glutInitWindowSize(int width,int height);

      width: Width in pixels

      height: Height in pixels.

- **glutCreateWindow** --- set the title to graphics window.
    - Declaration: Int glutCreateWindow(char *title);
    - Remarks: This function creates a window on the display. The string title can be used to label the window.The integer value returned can be used to set the current window when multiple windows are created.

- **glutDisplayFunc**
  - Declaration: void glutDisplayFunc(void(*func)void)); Remarks: This function registers the display function that is executed when the window needs to be redrawn.

- **glClear:**
  - Declaration: void glClear();
  - Remarks: This function clears the particular buffer.

- **glClearColor:**
  - Declaration:
    void glClearColor(GLfloat red, GLfloat green, Glfloat blue, Glfloat alpha);
  - Remarks: This function sets the color value that is used when clearing the color buffer.

- **glEnd**
  - Declaration: void glEnd();
  - Remarks: This function is used in conjunction with glBegin to delimit the vertices of an opengl primitive.

- **glMatrixMode**
  - Declaration: void glMatrixMode(GLenum mode);
  - Remarks: This function specifies which matrix will be affected by subsequent transformations mode can be GL_MODELVIEW,GL_PROJECTION or GL_TEXTURE..

- **gluOrtho2D**
  - Declaration:
    void glOrtho(GLdouble left, GLdouble right, GLdouble bottom,GLdouble top);
  - Remarks: It defines an orthographic viewing volume with all parameters measured from the center of the projection plane.

- **glutInit**
  - Declaration:

    void glutInit(int *argc, char **argv);

  - Remarks: To start thru graphics system, we must first call glutInit (),glutInit will initialize the GLUT library and negotiate a session with the window system. During this process, glutInit may cause the termination of the GLUT program with an error message to the user if GLUT cannot be properly initialized.

- **glutSolidSphere**
    - Declaration: void glutSolidSphere (GLdouble radius , GLint slices, GLint stacks);
    - Renders a sphere centered at the modeling coordinates origin of the specified radius. The sphere is subdivided aroundthe Z axis into slices and along the Z axis into stacks.
- **glutTimerFunc**
    - void glutTimerFunc(unsigned int msecs, void(*func), int value);
    - glutTimerFunc registers the timer callback func to be triggered in at least msecs milliseconds. The value parameter to the timer callback will be the value of the value parameter to glutTimerFunc.

4.2 Source Code

```
#include <iostream>
#include <stdio.h>
 #include <stdlib.h>
 #include <time.h>
 #include <cmath>
 #include <GL/gl.h>
 #include <GL/glut.h>
 #include <GL/glu.h>

static int animationPeriod = 4;
static int isAnimate = 0;

const int fact = 3;
const int x = 80;
const double DEG2RAD = 3.1415926535897932384/180;

static double w = 200;
static int flag = 0;
static int walk = 0;
static int x_ = 2500;

using namespace std;

void animate(int value){

    if(isAnimate){

        glutPostRedisplay();
        glutTimerFunc(animationPeriod, animate, 1);
    }
}

void keyInput(unsigned char key , int x, int y){

    switch(key){
    case 27:
        exit(0);
    case ' ':
        if(isAnimate) isAnimate = 0;
        else{
            isAnimate = 1;
            animate(1);
        }
        break;
    }
}

bool collision(double len){
    if(abs(157 + x - (x_ + x + 50)) <= 100 + x){
        if(5 * fact + w <= 650 * len)return 1;
        return 0;
    }
    return 0;
}



void specialKeyInput(int key , int x , int y ){

    if( key == GLUT_KEY_UP && flag==0 && w <= 200.0){
        flag  = 1;
    }
    glutPostRedisplay();
}

void draw_circle(double theta, double inner_radius, double outer_radius, int x, int y, int
sin_sign = 1, int cos_sign = 1){

   glBegin(GL_POINTS);
   glColor3f(0.0/ 255.0, 0.0/ 255.0, 0.0/ 255.0);
   for(double r = outer_radius; r >= inner_radius; r -= 3.0){
        for(double i = 0; i < theta ; i++){
          double degInRad = i * DEG2RAD;
          glVertex2f( cos_sign * cos(degInRad) * r + x , sin_sign * sin(degInRad) * r + y  );
       }
   }
   glEnd();
}
```

```
void generate_tree(int x_, double len){

    int x = 30;
    glColor3f((0) / 255.0, (0) / 255.0, (0) / 255.0);
    glBegin(GL_POLYGON);
        glVertex2f(x_, 250 * len);
        glVertex2f(x_ + x, 250 * len);

        glVertex2f(x_ + x, 650 * len);
        glVertex2f(x_, 650 * len);
    glEnd();

    draw_circle(180.0, 0.0, x / 2, x_ + x / 2, 650 * len);

    glBegin(GL_POLYGON);
        glVertex2f(x_ + x + 25, 400 * len);
        glVertex2f(x_ + x + 50, 400 * len);
        glVertex2f(x_ + x + 50, 600 * len);
        glVertex2f(x_ + x + 25, 600 * len);
    glEnd();

    draw_circle(180.0, 0.0, 25.0 / 2, x_ + x + 75.0 / 2, 600 * len);

    glBegin(GL_POLYGON);
        glVertex2f(x_ - 25, 400 * len);
        glVertex2f(x_ - 50, 400 * len);
        glVertex2f(x_ - 50, 600 * len);
        glVertex2f(x_ - 25, 600 * len);
    glEnd();
    draw_circle(180.0, 0.0, 25.0 / 2, x_ - 75.0 / 2, 600 * len);
    draw_circle(90.0, 25, 50, x_ + x, 400 * len, -1);
    draw_circle(90.0, 25, 50, x_, 400 * len, -1, -1);
}

void reset(){

    w = 200;
    flag = 0;
    walk = 0;
    x_ = 2500;
    animationPeriod = 4;
    isAnimate = 0;
}

void render( void ){

    glClear(GL_COLOR_BUFFER_BIT);
    glPointSize(2);
    glBegin(GL_POINTS);
        glColor3f((0) / 255.0, (0) / 255.0, (0) / 255.0);

        for(int i = 0; i < 100; i++){
            glVertex2f(rand() % 2000, 200);
            glVertex2f((rand() + 31) % 2000, 150);
        }
    glEnd();

    generate_tree(x_, 1.0);

    if(x_>= 0)
        x_ -= 5;
    else{
        x_ = 2000 + rand()%400;
    }

    glLineWidth(2);
    glBegin(GL_LINES);
        glColor3f((0) / 255.0, (0) / 255.0, (0) / 255.0);
        glVertex2f(0, 250);
        glVertex2f(2000, 250);
    glEnd();
    glLineWidth(10);
     glBegin(GL_LINES);
        glColor3f(0 / 255.0, 0 / 255.0, 0 / 255.0);
        glVertex2f(10 + x, 75 * fact + w);
        glVertex2f(10 + x, 45 * fact + w);
        glVertex2f(15 + x, 65 * fact + w);
        glVertex2f(15 + x, 40 * fact + w);
        glVertex2f(20 + x, 60 * fact + w);
        glVertex2f(20 + x, 35 * fact + w)
        glVertex2f(25 + x, 55 * fact + w);
        glVertex2f(25 + x, 35 * fact + w);
        glVertex2f(30 + x, 55 * fact + w);
        glVertex2f(30 + x, 35 * fact + w);
        glVertex2f(35 + x, 55 * fact + w);
        glVertex2f(35 + x, 25 * fact + w);
        glVertex2f(40 + x, 60 * fact + w);
        glVertex2f(40 + x, 5 * fact + w-walk);
        glVertex2f(45 + x, 65 * fact + w);
        glVertex2f(45 + x, 15 * fact + w);
        glVertex2f(45 + x, 10 * fact + w-walk)
```

```
glVertex2f(45 + x, 5 * fact + w-walk);
glVertex2f(50 + x, 10 * fact + w-walk);
glVertex2f(50 + x, 5 * fact + w-walk);
glVertex2f(55 + x, 10 * fact + w-walk);
glVertex2f(55 + x, 5 * fact + w-walk);
glVertex2f(50 + x, 65 * fact + w);
glVertex2f(50 + x, 20 * fact + w);
glVertex2f(55 + x, 70 * fact + w);
glVertex2f(55 + x, 25 * fact + w);
glVertex2f(63 + x, 75 * fact + w);
glVertex2f(63 + x, 20 * fact + w)
glVertex2f(70 + x, 115 * fact + w);
glVertex2f(70 + x, 5 * fact + w+walk);
glVertex2f(78 + x, 120 * fact + w);
glVertex2f(78 + x, 25 * fact + w);
glVertex2f(78 + x, 10 * fact + w+walk);
glVertex2f(78 + x, 5 * fact + w+walk);
glVertex2f(85 + x, 10 * fact + w+walk);
glVertex2f(85 + x, 5 * fact + w+walk);
glVertex2f(87 + x, 120 * fact + w);
glVertex2f(87 + x, 115 * fact + w);
glVertex2f(87 + x, 110 * fact + w);
glVertex2f(87 + x, 30 * fact + w);
glVertex2f(95 + x, 120 * fact + w);
glVertex2f(95 + x, 35 * fact + w);
glVertex2f(103 + x, 120 * fact + w);
glVertex2f(103 + x, 75 * fact + w);
glVertex2f(103 + x, 65 * fact + w)
glVertex2f(103 + x, 60 * fact + w);
glVertex2f(110 + x, 65 * fact + w);
glVertex2f(110 + x, 60 * fact + w);
```

```
    glVertex2f(118 + x, 65 * fact + w);
    glVertex2f(118 + x, 55 * fact + w);
    glVertex2f(112 + x, 120 * fact + w);
    glVertex2f(112 + x, 85 * fact + w);
    glVertex2f(112 + x, 80 * fact + w);
    glVertex2f(112 + x, 75 * fact + w);
    glVertex2f(120 + x, 120 * fact + w);
    glVertex2f(120 + x, 85 * fact + w);
    glVertex2f(120 + x, 80 * fact + w);
    glVertex2f(120 + x, 75 * fact + w);
    glVertex2f(126 + x, 120 * fact + w);
    glVertex2f(126 + x, 85 * fact + w);
    glVertex2f(126 + x, 80 * fact + w);
    glVertex2f(126 + x, 75 * fact + w);
    glVertex2f(135 + x, 120 * fact + w);
    glVertex2f(135 + x, 85 * fact + w);
    glVertex2f(135 + x, 80 * fact + w);
    glVertex2f(135 + x, 75 * fact + w);
    glVertex2f(142 + x, 120 * fact + w);
    glVertex2f(142 + x, 85 * fact + w);
    glVertex2f(150 + x, 120 * fact + w);
    glVertex2f(150 + x, 85 * fact + w);
    glVertex2f(157 + x, 115 * fact + w);
    glVertex2f(157 + x, 85 * fact + w);
glEnd();

if(collision(1.0)){
    reset();
}
if( w <=200){
    if(walk==-20 )
```

```
            walk = 20;
        else{
            walk = -20;
        }
    }
    else{
        walk = 0;
    }



    if(flag==1){

        if(w<=1000 ){
            w = w + 8
        }
        else {
            flag = 0;
        }
    }

    else if(w >= 200 )
        w = w - 8;

    glFlush();
}

void setup(void){

            glClearColor(1.0, 1.0, 1.0, 0.0);
           glMatrixMode(GL_PROJECTION);
        gluOrtho2D(0.0, 2000, 0.0, 2000);

}

int main( int argc , char** argv ){

    srand(time(NULL));
    glutInit( &argc, argv );



    glutInitDisplayMode( GLUT_SINGLE | GLUT_RGBA );
    glutInitWindowSize( 1230, 650 );
    glutInitWindowPosition( 50 , 0 );
    glutCreateWindow("Dinosaur!!");
    setup();
    glutDisplayFunc(render);
    glutKeyboardFunc(keyInput)
    glutSpecialFunc(specialKeyInput);

    glutMainLoop();

}
```

CHAPTER 5

SCREENSHOTS

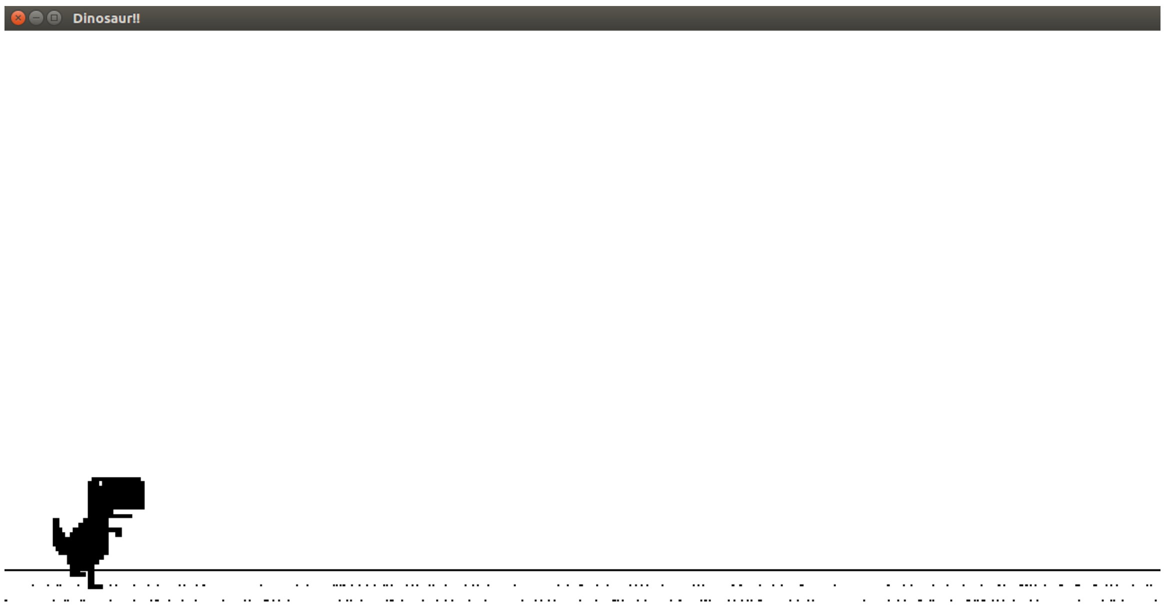


Fig 5.1: OPENING SCREEN

This is the initial screen when the game is loaded

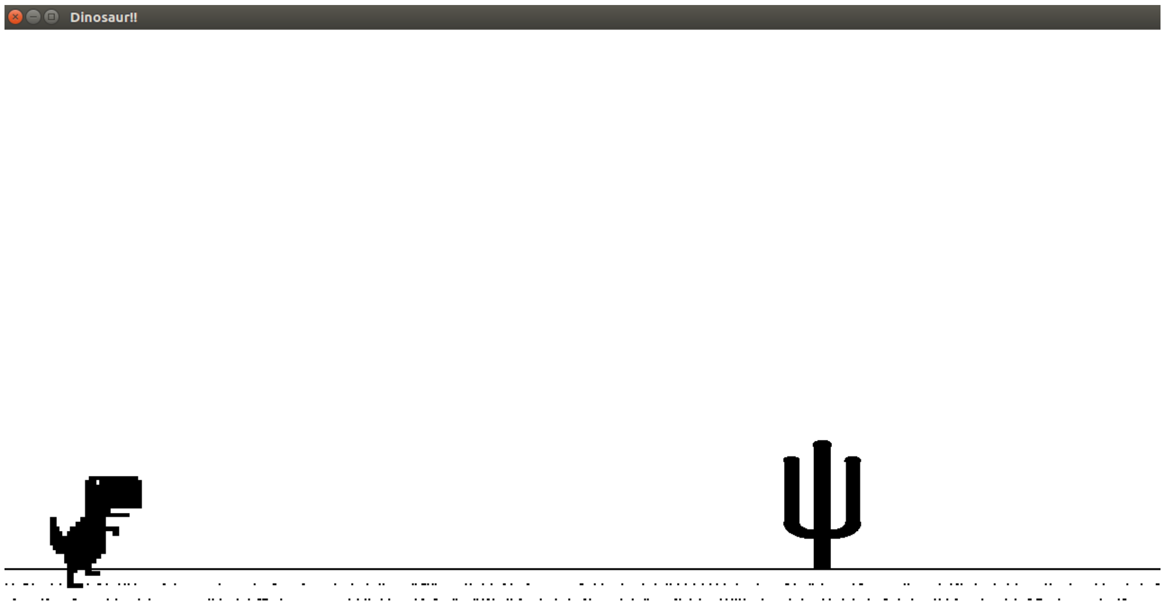


Fig 5.2: PLAY AREA

Game is started by clicking the Spacebar enabling the dinosaur to start moving forward

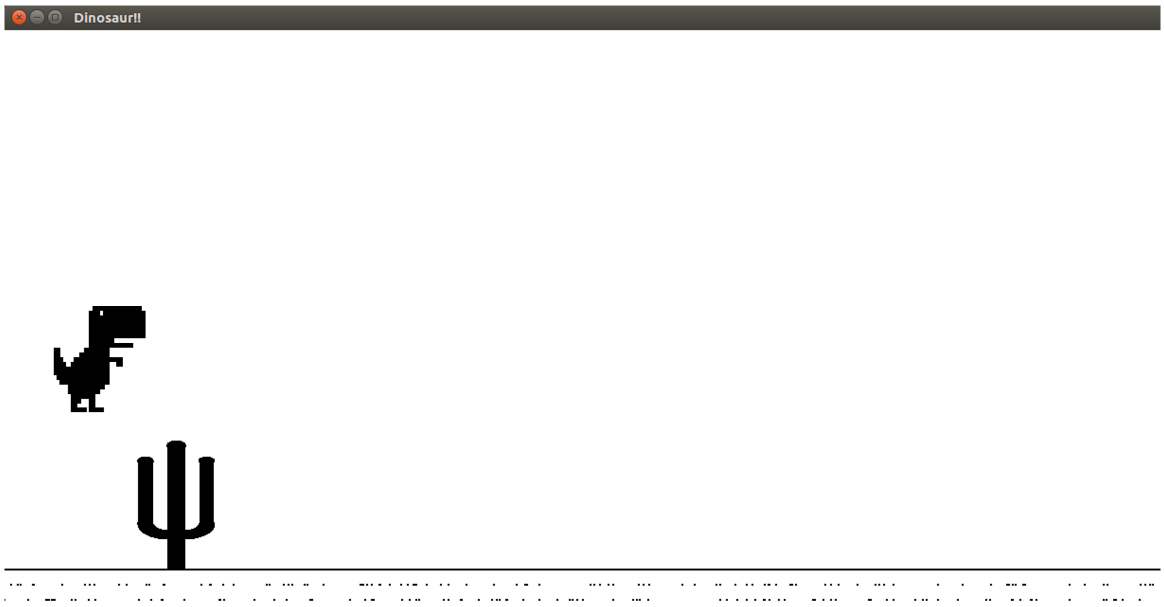


Fig 5.3: AVOIDING THE OBSTACLE

The Up arrow is used to avoid the obstacles by jumping over it

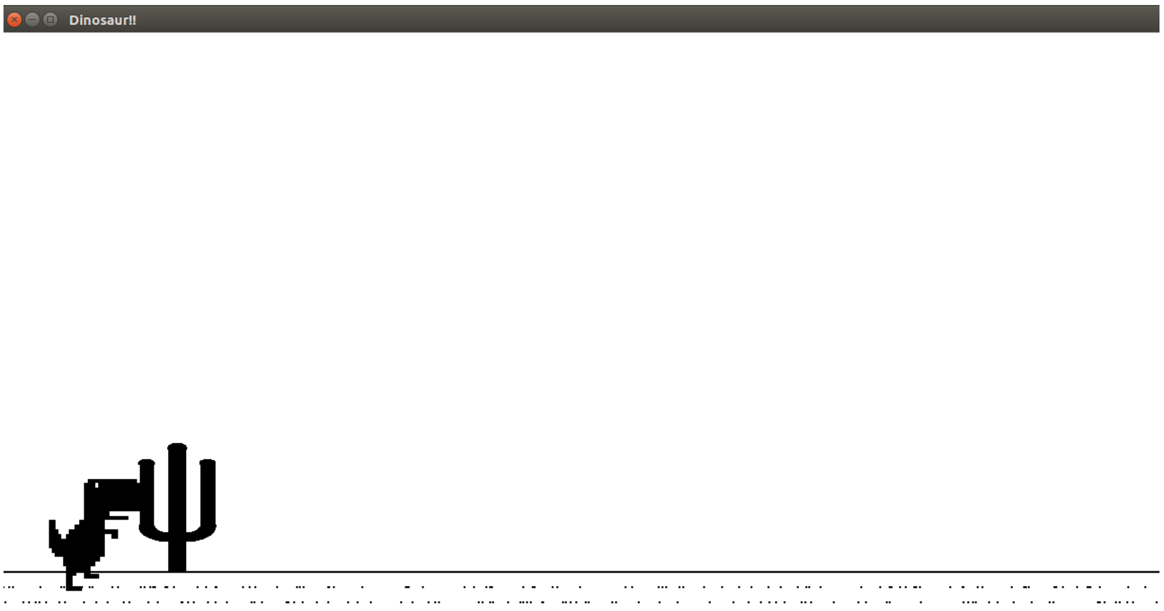
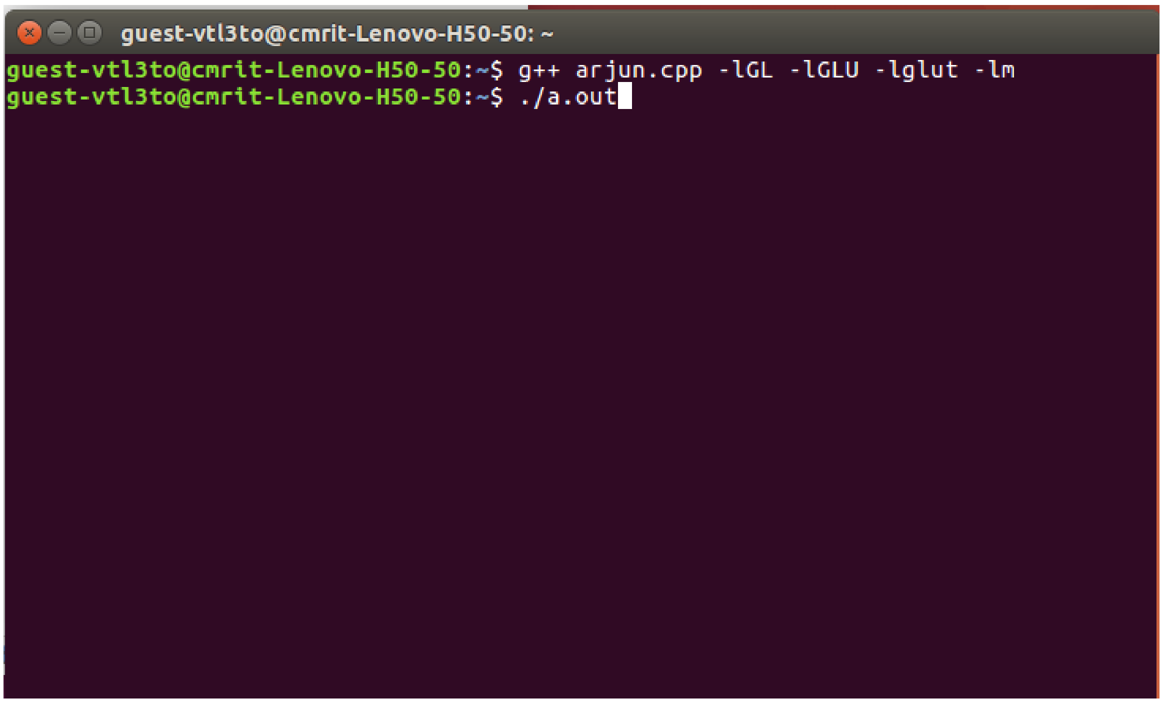


Fig 5.4 : END GAME SCREEN

The dinosaur and hence the game stops when the dinosaur has hit the obstacle



```
guest-vtl3to@cmrit-Lenovo-H50-50: ~
guest-vtl3to@cmrit-Lenovo-H50-50:~$ g++ arjun.cpp -lGL -lGLU -lglut -lm
guest-vtl3to@cmrit-Lenovo-H50-50:~$ ./a.out
```

Fig 5.5 : RUNNING COMMAND SCREEN

We used the C++ compiler to compile our .cpp file and execute it using ./a.out cmd

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

This is to conclude that the project we undertook has been worked upon with sincere effort and has been completed successfully. The requirements and goals of the project have been achieved. The desktop application has been thoroughly tested and can now players can play in the real world. By this project I hope to bring satisfaction to the users and meet their expectations.

BIBLIOGRAPHY

[1] https://www.youtube.com/watch?v=2apVwq-pX9E

[2] https://stackoverflow.com/search?q=jumping+dinosaur+game

[3] Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version,3rd / 4th Edition, Pearson Education,2011

[4] Edward Angel: Interactive Computer Graphics- A Top Down approach with OpenGL, 5 th edition. Pearson Education, 2008