VISVESVARAYA TECHNOLOGICAL UNIVERSITY
“Jnana Sangama”, Belagavi – 590 014

COMPUTER GRAPHICS LABORATORY WITH MINI PROJECT
REPORT (18CSL67) ON

“2D CHROME DINOSAUR”

Submitted in partial fulfilment of the requirements for the award of the Degree of

Bachelor of Engineering
In
Computer Science and Engineering
By

Mohammed Thanish Ali	4BP19CS042
Mushfiq K	4BP19CS046
Shaheer	4BP19CS056

Under the guidance of:

Mr. Afsar Baig M
Head of Department of CSE, BIT

Academic Year: 2021 -22

Department of Computer Science and Engineering,
Bearys Institute of Technology,

MANGALURU - 574 153

BEARYS INSTITUTE OF TECHNOLOGY

(*Affiliated to Visvesvaraya Technological University & Recognized by AICTE*)
INNOLI, MANGALURU – 574153, KARNATAKA

DEPARTMENTOF
COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the mini project work entitled "**2D Chrome Dinosaur"** carried out by **Mohammed Thanish Ali (4BP19CS042), Mushfiq K (4BP19CS046),** and **Shaheer (4BP19CS056)** in partial fulfilments of the requirements of Computer Graphics Laboratory with Mini Project (18CSL67) prescribed by the Visvesvaraya Technological University,Belagavi for the VI Semester B.E (Computer Science and Engineering) Degreecourse during the academic year of 2021-2022.

Signature of the H.O.D
Mr. Afsar Baig M
Head of Department
Department of CSE, BIT

DECLARATION

We, the students of the sixth semester of Computer Science and Engineering, Bearys Institute of Technology, Mangalore, declare that the work entitled “**2D CHROME DINOSAUR**” has been successfully completed under the guidance of Mr. Afsar Baig M, Computer Science and Engineering Department, Bearys Institute of Technology. This dissertation work is submitted to Visvesvaraya Technological University in partial fulfilment of the requirements for the award of Degree of Bachelor of Engineering in Computer Science during the academic year 2021 - 2022. Further, the matter embodied in the project report has not been submitted previously by anyone for the award of any degree or diploma to any university.

Team members:

1. Mohammed Thanish Ali (4BP19CS042)
2. Mushfiq K (4BP19CS046)
3. Shaheer (4BP19CS056)

Place:

Date:

ACKNOWLEDGEMENT

The knowledge and satisfaction that accompanies a successful completion of a project is hard to describe. Behind any successful project there are wise people guiding throughout. We thank them for guiding us, correcting our mistakes, and providing valuable feedback. We would consider it as our privilege to express our gratitude and respect to all those who guided and encouragement.

We extend our heartfelt gratitude to our chairman, **MR. SAYYED MUHAMMAD BEARY** and also to our beloved principal, **DR. S.I MANJUR BASHA** for the success of this project.

We are grateful to **MR. AFSAR BAIG M**, Professor & Head of CSE Department, Bearys Institute of Technology, for guidance and encouragement at all times needed.

ABSTRACT

A 2d graphic Chrome Dinosaur is a great start for a student who starts learning computer graphics & visualization**.** The development of this project has large scope to learn computer graphics from scratch. We used OpenGL utility toolkit to implement the algorithm, written it in C++ language.

There is still scope left in the development of project like, adding the welcome screen which displays our name, score in the game database saving the game and resuming the game from our previous progress. In future we hope we would implement itin source code for better experience of playing this game.

Finally, we could say by developing this project we have learnt the basics of computer graphics and in future by developing it further we shall learn more. It will be our pleasure if we could develop in 3d graphics package.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 About Computer Graphics

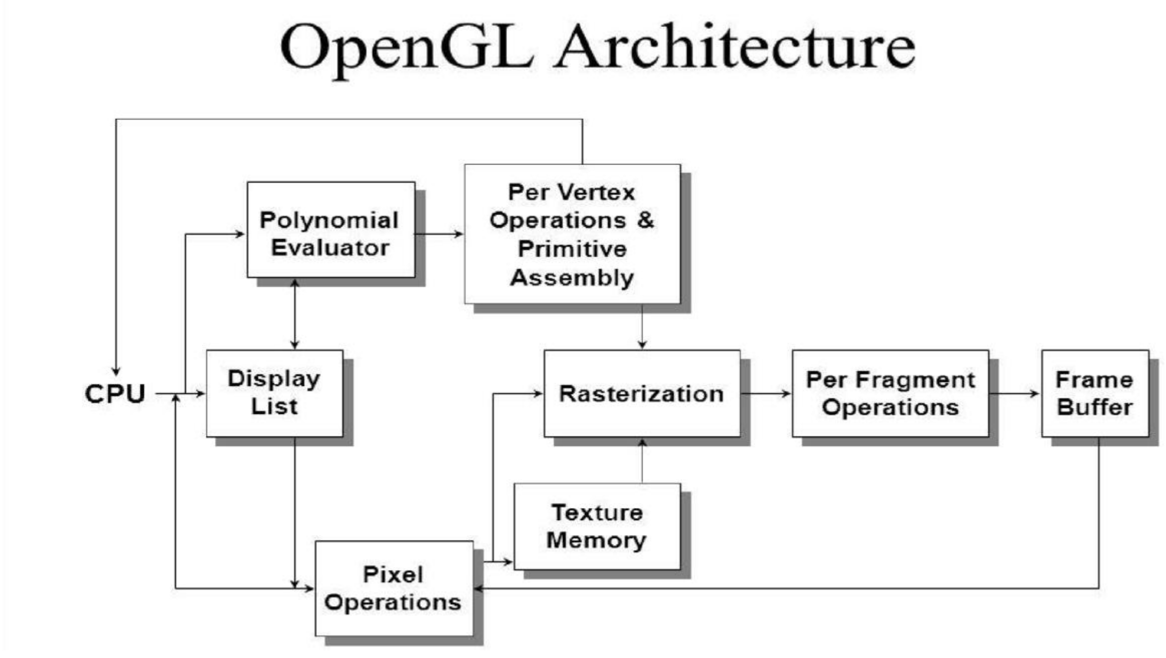
Computer graphics is an art of drawing pictures on computer screens with the help of programming. It involves computations, creation, and manipulation of data. In other words, we can say that computer graphics is a rendering tool for the generation and manipulation of images. In today's world, It is one of the most widely used powerful and interesting features of the computer which gives us the power to handle the graphical data very efficiently and also process them rapidly and effectively. Computer graphics started with a display of data on hard copy plotters and cathode ray tube screens soon after the introduction of computers themselves. The development of computer graphics has been driven both by the needs of the user community and by advances in hardware and software. Computer graphics today is largely interactive. The user controls the contents, structures and appearances of the object and of their displayed images by using input devices, such as keyboard, mouse or touch-screen.

1.2 About OpenGL

OpenGL is a software interface to graphics hardware. This interface consists of about 150distinct commands that you use to specify the objects and operations needed to produce interactive three-dimensional applications. It is designed as a streamlined, hardware independent interface to be implemented on many different hardware platforms.

1.2.1 OpenGL Architecture

This is a diagram representing the flow of graphical information, as it is processed from C.P.U to the frame buffr , in OpenGL. The upper pipeline is for the geometric, vertex-based primitives. The lower pipeline is for pixel-based, image primitives.



Texturing combines the two types of primitives together.

There are several APIs related to OpenGL:

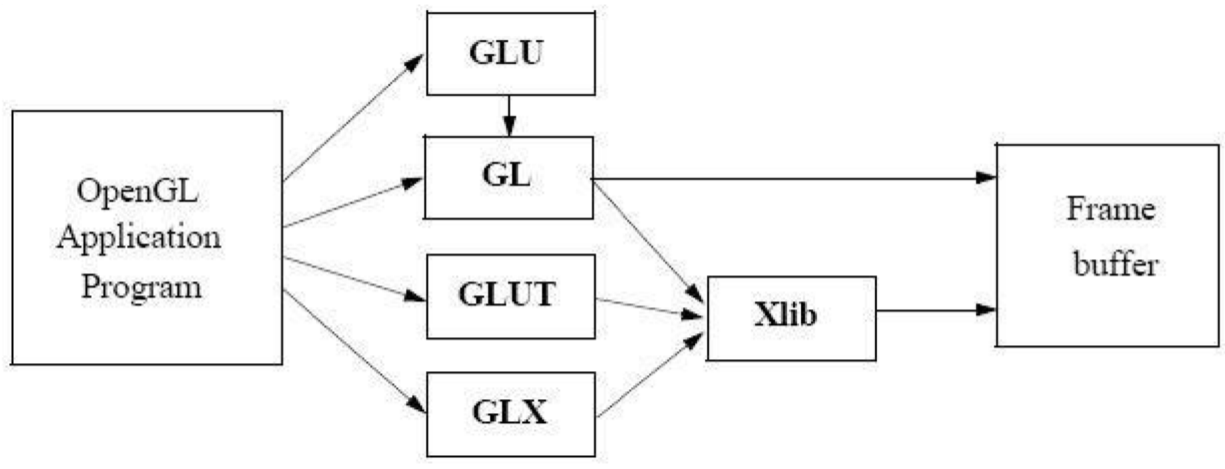
- AGL, WGL, GLX: OpenGL & windowing systems interfaces
  - GLU: 2D & 3D Geometry, tessellations, etc.
- GLUT: portable windowing API

1.3 OpenGL Library Organization

Support provided by a graphics API:

1. Primitive functions: points, line segments, polygons, text, curves, surfaces.
2. Attribute functions: colour, fill, type face.
3. Viewing functions: attributes of the synthetic camera.
4. Transformation functions: rotation, translation, scaling. Matrix computations.
5. Input functions: keyboard, pointing device.
6. System communication: window system, OS, other workstations, other users

OpenGL Library Organization



1.4 OpenGL in the Project

In the project, we use a particular graphics software system, OpenGL. The applications are designed to access OpenGL directly through functions in three libraries. The first is the GLlibrary in which the function name begins with gl. The second is the OpenGL utility library(GLU) which uses only GL functions but contains code for creating common objects and simplify viewing and these functions that begin with glu. To interface with the windows system and to get input from the external devices into our programs we use the third library, the Opengl Utility Toolkit (GLUT) whose functions are prefixed as glut.

1.5 OpenGL Functions

1.5.1 GLUT Callbacks

The OpenGL Utility Library (GLU) provides many of the modeling features. It has a wide variety of functions in GL, GLU, GLUT, and GLE. GLUT uses a callback mechanism to do its event processing. Callbacks simplify an event processing for the application developer. Given below are some of the functions used inthis project:

- glutDisplayFunc()- called when pixels in the window need to be refreshed.
- glutMouseFunc()- called when the user presses a mouse button on the mouse.
- glutKeyboardFunc()- called when user enters a value (input) though Keyboard.
- glutIdleFunc()- called when nothing else is going on.
- glutReshapeFunc()- sets the reshape callback for the current window.
- glutTimerFunc()- registers a timer callback to be triggered in a specified number of milliseconds.

1.5.2 OpenGL Command

OpenGL commands use the prefix **gl** and initial capital letters for each word making up thecommand name such as glBegin(). Similarly, OpenGL defined constants begin with **GL_**, use all capital letters, and use underscores to separate words such as **GL_COLOR_BUFFER_BIT**.

Some OpenGL command names have a number and one, two, or three letters at the end to denote the number and type of parameters to the command. The first character indicates thenumber of values of the indicated type that must be presented to the command. The secondcharacter or character pair indicates the specific type of the arguments: 8-bit integer, 16-bitinteger, 32-bit integer, single-precision floating-point, or double-precision floating-point. The final character, if present, is v, indicating that the command takes a pointer to an array (a vector) of values rather than a series of individual arguments.

SYSTEM REQUIREMENT

The package is designed such that users with a computer having minimum configuration can also use it, which does not require complex graphics packages.

The package requires simple in-built functions found in the header file along with a few users defined functions.

2.1 Software Requirements

- ✦ Operating System - Ubuntu 14.04
- ✦ Language Tool - C Compiler - GNU C Compiler
- ✦ Libraries - freeglut3
- ✦ Documentation Tool - MS-Word

2.2 Hardware Requirements

- ✦ Processor - Intel Pentium onwards Compatible Hardware
- ✦ RAM - 256Mb RAM (minimum)
- ✦ Hard Disk - 3 GB (minimum)
- ✦ Monitor – VGA Compatible
- ✦ Keyboard - Standard 101 keys Keyboard

DESIGN

The objective of the project is to run a Dinosaur with restricted upward and downward motion using keyboard, meanwhile Cactus will move towards Dinosaur and player have to avoid a collision between them.
A 2D Dinosaur is an object standing on pitch, if we press space key in the keyboard Dinosaur will start to move against the Cactus. If collision happens between Dinosaur and Cactus, the Dinosaur gets to its original start position
To avoid collision between Dinosaur and Cactus, Player has to use Arrow-Up key on keyboard. Using Arrow-Up key Dinosaur jumps over the Cactus and avoids collision.

The components used in this project are:

1. Dinosaur
2. Cactus
3. Floor

CHAPTER 4

IMPLEMENTATION

4.1 2D Chrome dinosaur

This mini project under Computer Graphics & Visualization Laboratory is an implementation of a kind popular helicopter game using the OpenGL GraphicsLibrary and GLUT Toolkit.
The objective of the project is to run a Dinosaur restricted upward and downward motion using keyboard,meanwhile Cactus will move towards Dinosaur and player have to avoid a collision between them. The Dinosaur will come to it’s start position if a collision occurrs.

Keyboard function: -
Pressing UP key move Dinosaur up.

4.2 OpenGl Functions

4.2.1 Init()

4.2.2 DisplayMode

This function call requests a display with the properties that are specified in mode.

Code: **glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);**

4.2.3 WindowSize

This function call specifies the initial height and width of window in pixels.

Code: **glutInitWindowSize(1230,650);**

4.2.4 WindowPosition

This function call specifies the position of the window in the specified window.

Code: **glutinitWindowPosition(50,0)**

4.2.5 Create Window

This function is used to give name to the window.

Code: glutCreateWindow

4.2.6 Display Function

This function is used to call the display function which is designed by the programmer.

Code: glutDisplayFunc

4.2.7 Keyboard Function

This function is used to connect the keys function to the keyboard.

Code: **glutKeybordFunc**(keys)

4.2.8 Flush

This function call forces any buffered openGL command to execute. Code: **Void glFlush();**

4.2.9 Init

This function used to initialize the GLUT library.

Code: **Void glFlush();**

4.2.10 Draw Circle

This function used to draw a circle.

Code: **Void draw_circle();**

4.2.11 Generate Tree

This function used to draw a tree.

Code: **Void generate_tree();**

4.2.12 Reset

This function used to reset the matrix.

Code: **Void reset();**

4.2.13 Color

This Function is used to colour the displaying objects

Code: **glColor3f (1.0,0.0,0.0);**

4.2.14 Projection Matrix

This function is used for the projection of matrix.

Code: **glMatrixMode()**

4.3 Standard Header and Library Functions

4.3.1 Stdio.h

This header file is used for standard input and output and stream manipulation function.

4.3.2 Stdlib.h

Stdlib.h is the header of the general-purpose standard library of C programming language whichinclude functions involving memory allocation, process control, conversions and others.

4.3.3 GL/glut.h

Library utility in OpenGL program.

4.3.4 String.h

It contains macro definitions, constants and declaration of function and types not only for stringhandling but also various memory handling functions.

4.3.5 Time.h

This header file contains definitions of functions to get and manipulate date and time information

4.3.6 Cmath

This header file is used to implement basic mathematical functions

5. Souce 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; //start point of dino

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(0.0,1.0,0.0);//tree color (outer part)
   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(0.0,1.0,0.0);//tree color(rectangle)
    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(0.54,0.27,0.07);//path color
    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(4);
  glBegin(GL_LINES);

glColor3f(0.54,0.27,0.07);//path line color
     glVertex2f(0, 350);                    //path way
     glVertex2f(2000, 350);
      glVertex2f(0, 50);
     glVertex2f(2000, 50);
  glEnd();

  glLineWidth(10);
  glBegin(GL_LINES);
     glColor3f(1.0,0.0,0.0);

     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(0.0, 0.0, 0.0, 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 6

RESULTS

6.1 Overview

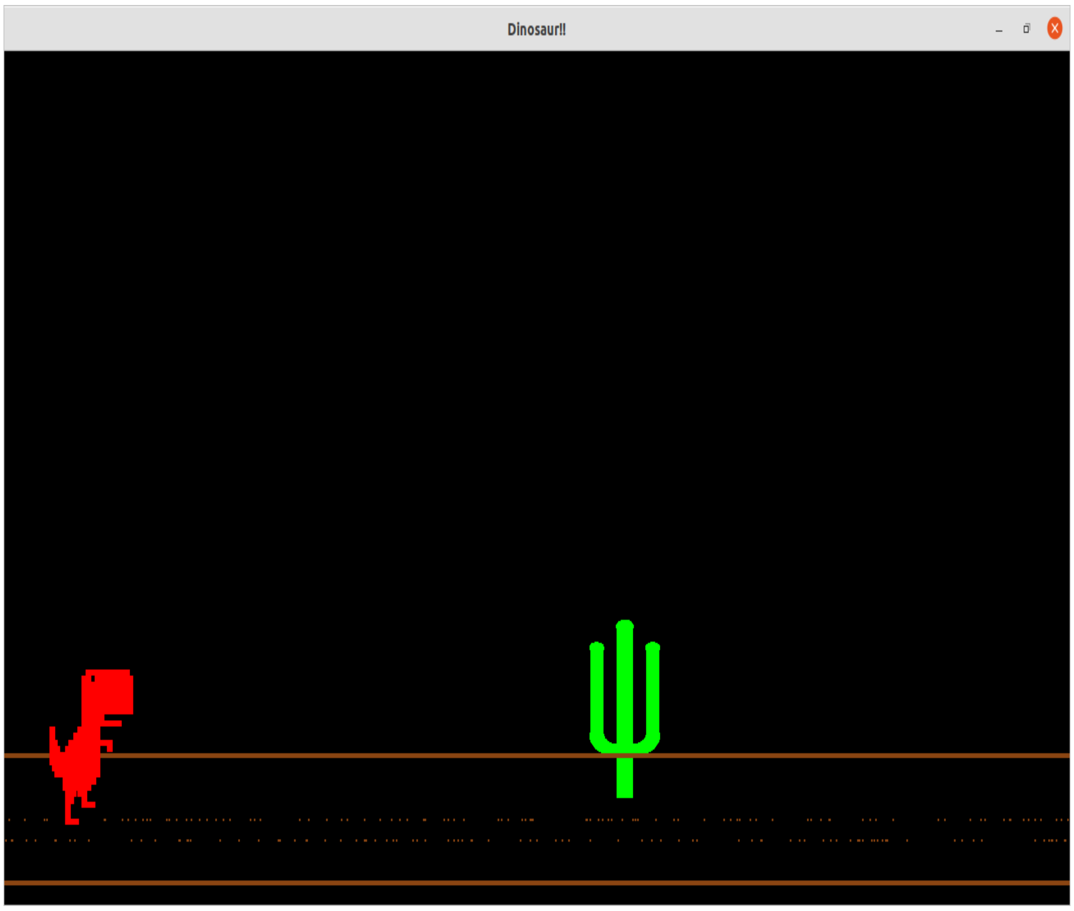


Figure 6.1: welcome window screen

6.2 Game Running Window

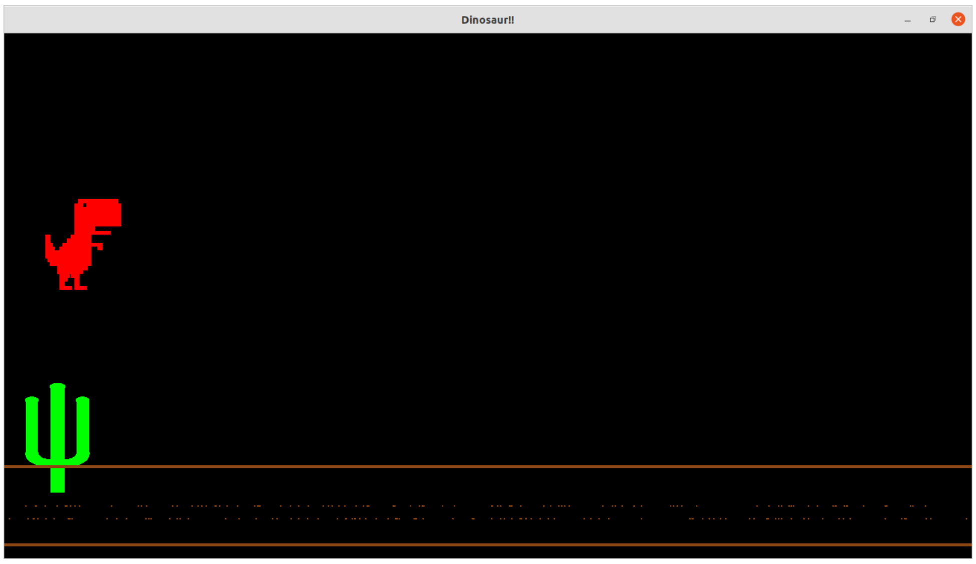
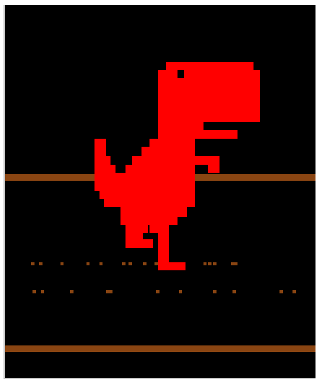
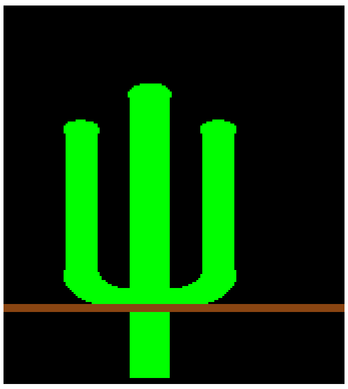


Figure 6.2: game running window

6.3 Objects


6.3.1 Dino


6.3.2 Cactus

CONCLUSION

We have attempted to design and implement “2D Dinosaur”. OpenGl supports enormous flexibility in the design and the use of OpenGl graphics programs. The presence of many built in classes methods take care of much functionality and reduce the job of coding as well as makes the implementation simpler.

The project was started with the designing phase in which we figured the requirements needed, the layout design, then comes the detail 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.

REFERENCES

✦ Interactive computer graphics a top-down approach with OpenGL- Edward Angel, 5th Edition , Addition -Wesley 2009

✦ Computer graphics using Open GL–F.S.Hill, Jr.2nd edition, Pearson education 2001

✦ OpenGL shading language -Randi J. Rost, John M. Kessenich - 2006 ✦ OpenGL game programming -Kevin Hawkins, Dave Astle - 2001.

✦ Computer graphics with OpenGL -Donald Hearn, M. Pauline Baker - 2004