VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi – 590 018

A

Mini Project Report

on

"SIMULATION OF STORY THIRSTY CROW"

Submitted in partial fulfillment of Computer Graphics Laboratory with Mini project

18CSL67 in Computer Science and Engineering for the Academic Year 2020-2021

Submitted by

PAVITHRA K TANTRY
1GA18CS189

Under the Guidance of

Mrs. Reshma S
Assistant Professor

GLOBAL ACADEMY OF TECHNOLOGY

Department of Computer Science and Engineering

(Accredited by NBA 2019-2022)

Raja Rajeshwari Nagar, Bengaluru – 560 098

2020-2021

GLOBAL ACADEMY OF TECHNOLOGY

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the VI Semester Mini Project in Computer Graphics Laboratory entitled **“Simulation of story thirsty crow”** carried out by **Ms. Pavithra K Tantry** , bearing **USN 1GA18CS189** is submitted in partial fulfillment for the award of the **Bachelor of Engineering** in Computer Science and Engineering from **Visvesvaraya Technological University, Belagavi** during the year 2020-2021.The Computer Graphics with Mini project report has been approved as it satisfies the academic requirements in respect of the mini project work prescribed for the said degree.

--------------------	-----------------------
Mrs. Reshma S Assistant Professor, Dept of CSE, GAT, Bengaluru.	**Dr. Bhagyashri R Hanji** Professor & Head, Dept of CSE, GAT, Bengaluru.

Name of the Examiner　　　　　　　　Signature with date

1.________________________　　　　________________________

2.________________________　　　　________________________

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant encouragement and guidance crowned our efforts with success.

I consider myself proud, to be part of **Global Academy of Technology** family, the institution which stood by our way in endeavors.

I express my deep and sincere thanks to our principal **Dr. N. Rana Pratap Reddy** for his support.

I would be grateful to **Dr . Bhagyashri R Hanji. ,** Professor and HOD, Dept of CSE who is source of inspiration and of invaluable help in channelizing my efforts in right direction.

I wish to thank my internal guide **Prof. Reshma S**, Assistant Professor, Dept of CSE for guiding and correcting various documents of mine with attention and care. They have taken lot of pain to go through the document and make necessary corrections as and when needed.

I would like to thank the faculty members and supporting staff of the Department of CSE, GAT for providing all the support for completing the Project work.

Finally, I am grateful to my parents and friends for their unconditional support and help during the course of my Project work.

PAVITHRA K TANTRY

1GA18CS189

ABSTRACT

The project uses the OpenGL and C/C++ library functions to simulate the moral story **‘Thirsty Crow’**.

The project has been implemented by efficiently using the data structures to obtain the optimized results and also various functions and features that are made available by the OpenGL software package have been utilized effectively. The different phases of the story are displayed by using the keyboard.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO COMPUTER GRAPHICS

Computer Graphics is concerned with all aspects of producing pictures or images using a computer. Graphics provides one of the most natural means of communicating within a computer, since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and effectively. Interactive computer graphics is the most important means of producing pictures since the invention of photography and television.

Applications of Computer Graphics

1. Display of information
2. Design
3. Simulation and animation
4. User interfaces

The Graphics Architecture

Graphics Architecture can be made up of seven components:

1. Display processors
2. Pipeline architectures
3. The graphics pipeline
4. Vertex processing
5. Clipping and primitive assembly
6. Rasterization
7. Fragment processing

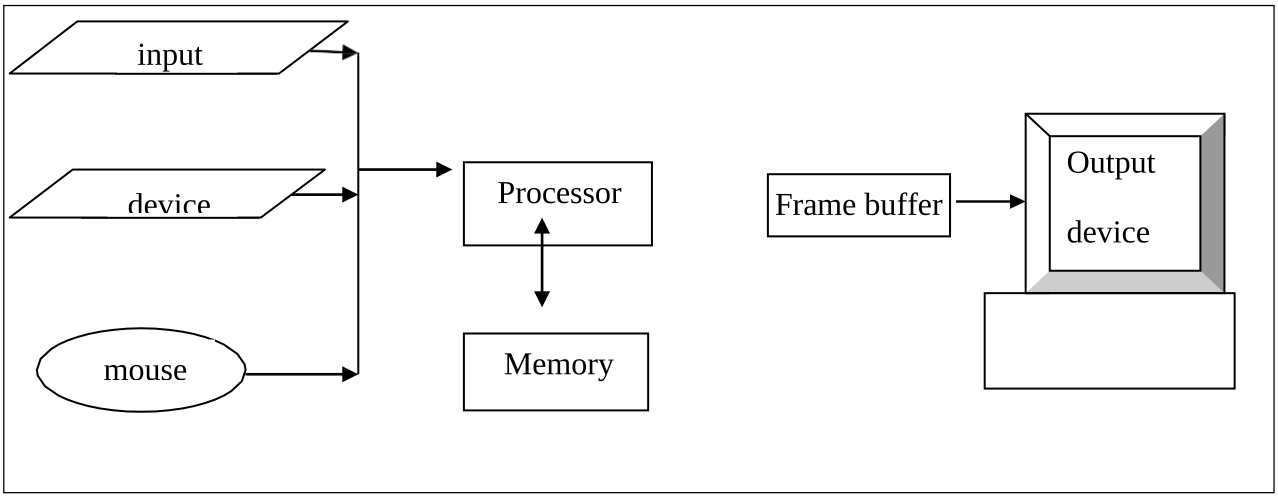


Figure 1.1: Components of Graphics Architecture and their working

1.2 INTRODUCTION TO OPENGL

OpenGL is software used to implement computer graphics. The structure of OpenGL is similar to that of most modern APIs including Java 3D and DirectX. OpenGL is easy to learn, compared with other.

APIs are nevertheless powerful. It supports the simple 2D and 3D programs. It also supports the advanced rendering techniques. OpenGL API explains following 3 components

1. Graphics functions
2. Graphics pipeline and state machines
3. The OpenGL interfaces

There are so many polygon types in OpenGL like triangles, quadrilaterals, strips and fans. There are 2 control functions, which will explain OpenGL through,

1. Interaction with window system
2. Aspect ratio and view ports

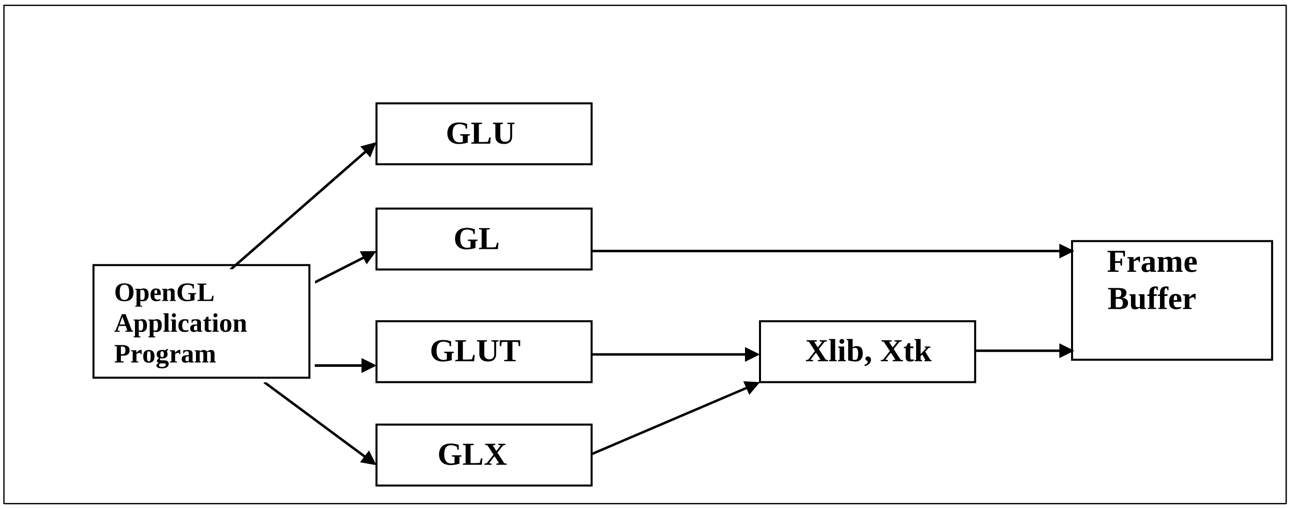


Figure 1.2: OpenGL Library organization

Most implementations of OpenGL have a similar order of operations, a series of processing stages called the OpenGL rendering pipeline. This ordering, as shown in Figure 1.2, is not a strict rule of how OpenGL is implemented but provides a reliable guide for predicting what OpenGL will do. The following diagram shows the assembly line approach, which OpenGL takes to process data. Geometric data (vertices, lines, and polygons) follow the path through the row of boxes that includes evaluators and per-vertex operations, while pixel data (pixels, images, and bitmaps) are treated differently for part of the process. Both types of data undergo the same final steps before the final pixel data is written into the frame buffer.

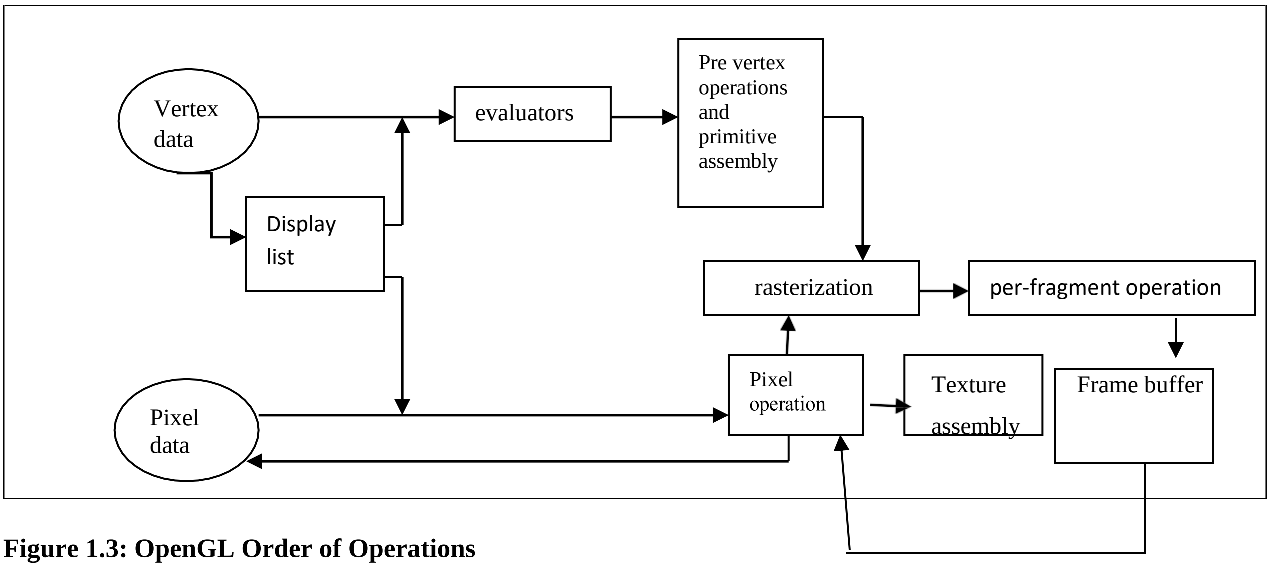


Figure 1.3: OpenGL Order of Operations

CHAPTER 2

REQUIREMENTS SPECIFICATION

2.1 SOFTWARE REQUIREMENTS

- Operating system – Windows 10
- Visual Studio 2019
- OPENGL library files – GL, GLU, GLUT
- Language used is C/C++

2.2 HARDWARE REQUIREMENTS

- Processor – Intel i3 7th Gen
- Memory – 8GB RAM
- 1TB Hard Disk Drive
- Keyboard
- Display device

CHAPTER 3

SYSTEM DEFINITION

3.1 PROJECT DESCRIPTION

OpenGL is software which provides a graphical interface. It is an interface between the application program and the graphics hardware.

‘Thirsty Crow’ is one of the most popular and well-known fables. The story is about a thirsty crow that comes upon a pitcher with very little water, beyond the reach of its beak. After failing, the bird drops pebbles one by one until the water rises to the top of the pitcher, allowing the crow to drink.

This story teaches us patience, presence of mind. Therefore, the moral of this story is that “Where there is a will there is a way!”

Simulation of the story thirsty crow involves different phases of the story and its implementation.

3.2 USER DEFINED FUNCTIONS

- **void Display1() :** This function is used to create the basic view of the forest.
- **void Display2() :** This function is used to show the crow in the forest.
- **void Display3() :** This function shows crow searching for water.
- **void Display4() :** This function shows crow finding a pot.
- **void Display5() :** This function shows crow trying to drink water from the pot.
- **void Display6() :** This function shows crow throwing pebbles one by one into the pot.
- **void Display7() :** This function shows water has risen, crow drinks water and flies away.
- **void Display8() :** This function displays the moral of the story.

CHAPTER 4

IMPLEMENTATION

4.1 SOURCE CODE

```
#include <windows.h>
#include <glut.h>
#include<iostream>
# define PI 3.1416

GLfloat position = -1.2f, xpos = -1.0f, ypos = 0.45f, yup, ydown, speed = 0.08f, crowspeed = 0.1f;
int  dsix = 0, waterflag = 0, pebbleflag = 0, stonereturnflag = 0, flyaway = 0;

void Display1();
void Display2();
void Display3();
void Display4();
void Display5();
void Display6();
void Display7();
void Display8();

void move(int value) {
  if (position > 1.3)
    {
        position = -1.2f;
    }
    position += speed;     if (!waterflag)
    {
        xpos += crowspeed;
        ypos -= crowspeed;
    }
    if (stonereturnflag)
    {
        xpos += crowspeed;
        yup += crowspeed;
```

```
   }
    ydown -= speed;
    glutPostRedisplay();

    glutTimerFunc(100, move, 0);
}

void output(GLfloat x, GLfloat y, const char* text)
{
    const char* p;
    glPushMatrix();
    glTranslatef(x, y, 0);
    glScaled(0.035, 0.035, 0);
    for (p = text; *p; p++)
        glutStrokeCharacter(GLUT_STROKE_ROMAN, *p);
    glPopMatrix();
}

void output1(GLfloat x, GLfloat y, const char* text)
{
    const char* p;
    glPushMatrix();
    glTranslatef(x, y, 0);
    glScaled(0.0005, 0.0005, 0);
    for (p = text; *p; p++)
        glutStrokeCharacter(GLUT_STROKE_ROMAN, *p);
    glPopMatrix();
}

void StartingText()
{
    glColor3f(1.0, 1.0, 1.0);
    output(-40.0, 50.0, "GLOBAL ACADEMY OF TECHNOLOGY");
    output(-20.0, 40.0, "DEPT OF CSE");
    output(-30, 25.0, "NAME: PAVITHRA K TANTRY");
    output(-30.0, 15.0, "USN: 1GA18CS189");
    output(-60.0, 5.0, "PROJECT: SIMULATION OF STORY THIRSTY CROW");
    output(-45.0, -5.0, "UNDER THE GUIDANCE OF: MRS. RESHMA S");
    output(-55.0, -40.0, "TO SEE THE STORY PRESS 1, 2, 3, 4, 5, 6, 7, 8");
    glFlush();
}

void circle(GLfloat x, GLfloat y, GLfloat radius, GLfloat colorx, GLfloat colory, GLfloat colorz)
{
    int triangleAmount = 20;
    GLfloat twicePi = 2.0f * PI;
    glBegin(GL_TRIANGLE_FAN);
    glColor3f(colorx, colory, colorz);
    glVertex2f(x, y); // center of circle
    for (int i = 0; i <= triangleAmount; i++) {
        glVertex2f(
            x + (radius * cos(i * twicePi / triangleAmount)),
            y + (radius * sin(i * twicePi / triangleAmount))
        );
    }
    glEnd();

}

void sun()
{
    circle(-.7f, .8f, .10f, 1.0, .25, 0.0);
}

void Pebble()
{
    circle(0.6f, 0.0f, 0.02f, 0.2f, 0.2f, 0.2f);
}

void Bird()
{
    glBegin(GL_POLYGON);
    glColor3f(0.0, 0.0, 0.0);
    glVertex2f(-0.5, 0.15);
    glVertex2f(-0.3, 0.25);
    glVertex2f(-0.1, 0.15);
    glVertex2f(-0.3, 0.09);
    glEnd();

    //WING ONE
    glBegin(GL_POLYGON);
    glVertex2f(-0.4, 0.2);
    glVertex2f(-0.4, 0.25);
    glVertex2f(-0.25, 0.35);
    glVertex2f(-0.3, 0.28);
    glVertex2f(-0.35, 0.2);
    glEnd();

    //WING TWO
    glBegin(GL_POLYGON);
    glVertex2f(-0.4, 0.20);
    glVertex2f(-0.2, 0.31);
    glVertex2f(-0.1, 0.22);
    glEnd();

    //BEAK
    glBegin(GL_LINES);
    glColor3f(0.0, 0.0, 0.0);
    glVertex2f(-0.55, 0.1);
    glVertex2f(-0.49, 0.15);
    glVertex2f(-0.55, 0.1);
    glVertex2f(-0.48, 0.14);
    glVertex2f(-0.48, 0.14);
    glVertex2f(-0.5, 0.1);
    glVertex2f(-0.5, 0.1);
    glVertex2f(-0.45, 0.15);
    glEnd();

    //EYE
    glPointSize(25.0);
    glTranslatef(-0.45f, 0.18f, 0);
    glBegin(GL_POINTS);
    glColor3f(0.0, 0.0, 0.0);
    glVertex2f(-0.0f, -0.0f);
    glEnd();
    glPointSize(5.0);
    glTranslatef(-0.01f, 0.0f, 0);
    glBegin(GL_POINTS);
    glColor3f(1.0, 0.0, 0.0);
    glVertex2f(-0.0f, -0.0f);
    glEnd();
    if (pebbleflag)
    {
        circle(-0.075f, -0.075f, 0.02f, 0.2f, 0.2f, 0.2f);
    }
}

void mountain()
{
    glColor3f(0.1f, 0.0f, 0.0f);
    glBegin(GL_POLYGON);
    glVertex2f(-0.93, 0.1);
    glVertex2f(-0.0, 0.5);
    glVertex2f(0.93, 0.1);
    glEnd();
}

void sky()
{
    glBegin(GL_POLYGON);
    glColor3f(0.4f, 0.7f, 0.9f);
    glVertex2f(-1.0f, 1.0f);
    glVertex2f(1.0f, 1.0f);
    glVertex2f(1.0f, 0.1f);
    glVertex2f(-1.0f, 0.1f);
    glEnd();
}

void pot()
{
    int triangleAmount = 20;
    GLfloat k = .4f; GLfloat l = -.7f;
    GLfloat radius = .13f;
    GLfloat Pi = PI;
    circle(0.4f, -0.7f, 0.13f, 0.439216f, 0.858824f, 0.576471f);
    glBegin(GL_TRIANGLE_FAN);
    glColor3f(0.558824f, 0.558824f, 0.558824f);
    glVertex2f(k, l); // center of circle
    for (int i = 0; i <= triangleAmount; i++) {
        glVertex2f(
            k + (radius * cos(i * Pi / triangleAmount)),
            l + (radius * sin(i * Pi / triangleAmount))
        );
    }
    glEnd();
    glBegin(GL_POLYGON);
    glColor3f(0.558824f, 0.558824f, 0.558824f);
    glVertex2f(0.47f, -0.65f);
    glVertex2f(0.32f, -0.65f);
    glVertex2f(0.32f, -0.55f);
    glVertex2f(0.47f, -0.55f);
    glEnd();
    glBegin(GL_POLYGON);
    glColor3f(0.7, 0.7, 0.7);
    glVertex2f(0.47f, -0.55f);
    glVertex2f(0.32f, -0.55f);
    glVertex2f(0.30f, -0.52f);
    glVertex2f(0.50f, -0.52f);
    glEnd();
}

void waterdrop()
{
    glBegin(GL_POLYGON);
    glColor3f(0.439216f, 0.858824f, 0.576471f);
    glVertex2f(0.16f, -0.72f);
    glVertex2f(0.11f, -0.70f);
    glVertex2f(.08f, -0.67f);
    glVertex2f(0.11f, -0.64f);
    glVertex2f(0.13f, -0.64f);
    glVertex2f(0.15f, -0.67f);
    glVertex2f(0.16f, -0.70f);
    glVertex2f(0.17f, -0.72f);
    glEnd();

    glBegin(GL_POLYGON);
    glColor3f(0.439216f, 0.858824f, 0.576471f);
    glVertex2f(0.20f, -0.62f);
    glVertex2f(0.11f, -0.60f);
    glVertex2f(.08f, -0.57f);
    glVertex2f(0.11f, -0.54f);
    glVertex2f(0.16f, -0.54f);
    glVertex2f(0.18f, -0.57f);
    glVertex2f(0.18f, -0.60f);
    glEnd();
    glLoadIdentity();
    glBegin(GL_POLYGON);
    glColor3f(0.439216f, 0.858824f, 0.576471f);
    glVertex2f(0.47f, -0.58f);
    glVertex2f(0.32f, -0.58f);
    glVertex2f(0.32f, -0.55f);
    glVertex2f(0.47f, -0.55f);
    glEnd();
    circle(0.4f, -0.7f, 0.13f, 0.439216f, 0.858824f, 0.576471f);
}
void stone()
{
    circle(-.7f, -.7f, .02f, 0.2f, 0.2f, 0.2f);
    circle(-.67f, -.75f, .02f, 0.2f, 0.2f, 0.2f);
    circle(-.67f, -.67f, .02f, 0.2f, 0.2f, 0.2f);
    circle(-.63f, -.7f, .02f, 0.2f, 0.2f, 0.2f);
    circle(-.66f, -.70f, .02f, 0.2f, 0.2f, 0.2f);
}

void grass()
{
    glBegin(GL_POLYGON);
    glColor4f(0.0, 1.0, 0.0, 1.0);
    glVertex2f(-1.0f, 0.1f);
    glVertex2f(1.0f, 0.1f);
    glVertex2f(1.0f, -1.0f);
    glVertex2f(-1.0f, -1.0f);
    glEnd();
}

void cloud1()
{
    circle(.5f, .8f, .05f, 1.0, 1.0, 1.0);
    circle(.55f, .78f, .05f, 1.0, 1.0, 1.0);
    circle(.45f, .78f, .05f, 1.0, 1.0, 1.0);
    circle(.52f, .75f, .05f, 1.0, 1.0, 1.0);
    circle(.6f, .77f, .05f, 1.0, 1.0, 1.0);
    glFlush();
}

void cloud2()
{
    circle(-.5f, .8f, .05f, 1.0, 1.0, 1.0);
    circle(-.55f, .78f, .05f, 1.0, 1.0, 1.0);
    circle(-.45f, .78f, .05f, 1.0, 1.0, 1.0);
    circle(-.52f, .75f, .05f, 1.0, 1.0, 1.0);
    circle(-.6f, .77f, .05f, 1.0, 1.0, 1.0);
    glFlush();
}

void tree()
{
    glBegin(GL_POLYGON);
    glColor3ub(83, 53, 10);
    glVertex2f(-0.62f, -0.24f);
    glVertex2f(-0.58f, -0.24f);
    glVertex2f(-0.58f, -0.8f);
    glVertex2f(-0.62f, -0.8f);
    glEnd();
    circle(-.67f, -.11f, .15f, 0.0, 0.3, 0.0);
    circle(-.7f, .1f, .15f, 0.0, 0.3, 0.0);
    circle(-.59f, .23f, .15f, 0.0, 0.3, 0.0);
    circle(-.5f, .05f, .18f, 0.0, 0.3, 0.0);
    circle(-.53f, -.12f, .15f, 0.0, 0.3, 0.0);
}

void reshape(int w, int h)
{
    float aspectRatio = (float)w / (float)h;
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(145, aspectRatio, 1.0, 100.0);
    glMatrixMode(GL_MODELVIEW);
}

void Display(void)
{
    glClearColor(0.0f, 0.0f, 0.0f, 0.0f); //Set background color
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    glTranslatef(0, 0, -20);
    StartingText();
    glFlush();
}

void Display1()
{
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    sky();
    grass();
    cloud1();
    cloud2();
    glColor3f(0.0, 0.0, 0.0);
    output1(-0.75, -0.9, "Once, it was a hot sunny day...");
    glLoadIdentity();
    glTranslatef(0.0, 0.0, 0.0);
    tree();
    glLoadIdentity();
    glTranslatef(0.6, 0.0, 0.0);
    tree();
    glLoadIdentity();
    glTranslatef(1.2, 0.0, 0.0);
    tree();
    glLoadIdentity();
    glTranslatef(-0.15, 0.0, 0.0);
    sun();
    glLoadIdentity();
    glFlush();
}

void Display2()
{
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    sky();
    grass();
    cloud1();
    cloud2();
    glColor3f(0.0, 0.0, 0.0);
    output1(-0.75, -0.9, "A crow was very thirsty..");
    glLoadIdentity();
    glTranslatef(0.0, 0.0, 0.0);
    tree();
    glLoadIdentity();
    glTranslatef(1.2, 0.0, 0.0);
    tree();
    glLoadIdentity();
    glTranslatef(-0.17, 0.0, 0.0);
    sun();
    glTranslatef(position, 0.4f, 0.0f);
    glRotatef(180, 0, 1, 0);
    Bird();
    glLoadIdentity();
    glFlush();
}

void Display3()
{
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    sky();
    grass();
    mountain();
    cloud1();
    cloud2();
    glColor3f(0.0, 0.0, 0.0);
    output1(-0.75, -0.9, "It looked for water here and there..");
    glLoadIdentity();
    glTranslatef(0.0, 0.0, 0.0);
    tree();
    glLoadIdentity();
    glTranslatef(-0.17, 0.0, 0.0);
    sun();
    glTranslatef(position, 0.4f, 0.0f);
    glRotatef(180, 0, 1, 0);
    Bird();
    glLoadIdentity();
    glFlush();
}

void Display4()
{
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    sky();
    grass();
    pot();
    cloud1();
    cloud2();
    glColor3f(0.0, 0.0, 0.0);
    output1(-0.75, -0.9, "At last it found a pot...");
    glLoadIdentity();
    glTranslatef(1.5, 0.0, 0.0);
    tree();
    glLoadIdentity();
    glTranslatef(-0.17, 0.0, 0.0);
    sun();
    glLoadIdentity();
    stone();
    glPushMatrix();
    glTranslatef(-1.0, 0.45f, 0.0f);
    glRotatef(180, 0, 1, 0);
    Bird();
    glPopMatrix();
    glFlush();
}

void Display5()
{
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    sky();
    grass();
    pot();
    cloud1();
    cloud2();
    glColor3f(0.0, 0.0, 0.0);
    output1(-0.85, -0.9, "But there was very little water & it couldn't reach...");
    glLoadIdentity();
    glTranslatef(1.5, 0.0, 0.0);
    tree();
    glLoadIdentity();
    glTranslatef(-0.17, 0.0, 0.0);
    sun();
    glLoadIdentity();
    stone();
    glPushMatrix();
    glTranslatef(xpos, ypos, 0.0f);
    if (xpos >= -0.2f && ypos <= -0.55f)
    {
        waterflag = 1;
    }
    glRotatef(180, 0, 1, 0);
    if (waterflag == 1)
    {
        Bird();
    }
    else
        Bird();
    glPopMatrix();
    glFlush();
}
```

```
void Display6()
{
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    sky();
    grass();
    pot();
    cloud1();
    cloud2();
    glColor3f(0.0, 0.0, 0.0);
    output1(-1.0, -0.9, "So it picked the pebbles one by one & put it in the pot...");
    glLoadIdentity();
    glTranslatef(1.5, 0.0, 0.0);
    tree();
    glLoadIdentity();
    glTranslatef(-0.17, 0.0, 0.0);
    sun();
    glLoadIdentity();
    stone();
    glPushMatrix();
    glTranslatef(xpos, yup, 0.0f);
    glRotatef(180, 0, 1, 0);
    Bird();
    glPopMatrix();
    if (!pebbleflag)
    {
        glPushMatrix();
        glTranslatef(-0.2, ydown, 0.0f);
        Pebble();
        glPopMatrix();
        if (ydown < -0.7)  //stone into pot
        {
            ydown = 0.5;
        }
    }
    glFlush();
    if (xpos >= -0.2 && yup >= -0.25 && dsix == 0)
    {
        waterflag = 1;
        stonereturnflag = 0;
        dsix = 1;
        pebbleflag = 0;
        ydown = 0.2;
    }
}

void Display7()
{
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    sky();
    grass();
    pot();
    cloud1();
    cloud2();
    glColor3f(0.0, 0.0, 0.0);
    output1(-0.95, -0.9, "Water rised to the top of the pot,");
    output1(-0.95, -0.98, "it drank water and flew away...");
    glTranslatef(0.1f, 0.0f, 0.0f);
    waterdrop();
    glLoadIdentity();
    glTranslatef(1.5, 0.0, 0.0);
    tree();
    glLoadIdentity();
    glTranslatef(-0.17, 0.0, 0.0);
    sun();
    glLoadIdentity();
    stone();
    glPushMatrix();
    if (!flyaway)
    {
        glTranslatef(-0.2f, ydown, 0.0f);
        glRotatef(180, 0, 1, 0);
        Bird();
    }
    else
    {
        glTranslatef(xpos - 0.2, yup - 0.4, 0.0f);
        glRotatef(180, 0, 1, 0);
        Bird();
    }
    glPopMatrix();
    glFlush();
    if (ydown < -0.6)
    {
        flyaway = 1;
        stonereturnflag = 1;
    }
}

void Display8()
{
    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 1.0, 1.0);
    output1(-0.40, 0.5, "MORAL OF THE STORY ");
    output1(-0.80, -0.0, "WHERE THERE IS A WILL THERE IS  A WAY ");
    glFlush();
}
```

```
void Keypress(unsigned char key, int x, int y) {
    switch (key) {
    case '1':
        printf("1 displayed\n");
        glutDestroyWindow(1);
        glutInitWindowSize(1240, 680);
        glutInitWindowPosition(100, 100);
        glutCreateWindow("THIRSTY CROW");
        glutKeyboardFunc(Keypress);
        glutDisplayFunc(Display1);
        break;
    case '2':
        printf("\n2 Displayed\n");
        position = -1.0f;
        glutDisplayFunc(Display2);
        break;
    case '3':
        printf("\n3 Displayed\n");
        position = -1.0f;
        glutDisplayFunc(Display3);
        break;
    case '4':
        printf("\n4 Displayed\n");
        position = -0.9f;
        glutDisplayFunc(Display4);
        break;
    case '5':
        printf("\n5 Displayed\n");
        position = -0.9f;
        xpos = -1.0;
        ypos = 0.45;
        glutDisplayFunc(Display5);
        break;
    case '6':
        printf("\n6 Displayed\n");
        position = -0.9f;
        xpos = -1.4;
        ypos = -0.85;
        yup = ypos;
        pebbleflag = 1;
        stonereturnflag = 1;
        glutDisplayFunc(Display6);
        break;
    case '7':
        printf("\n7 Displayed\n");
        position = -1.0f;
        glutDisplayFunc(Display7);
        break;
    case '8':
        printf("\nMoral of the Story");
        glutDestroyWindow(1);
        glutInitWindowSize(1240, 680);
        glutInitWindowPosition(100, 100);
        glutCreateWindow("Moral");
        glutKeyboardFunc(Keypress);
        glutDisplayFunc(Display8);
        break;
    }
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitWindowSize(1240, 680);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("THIRSTY CROW");
    glutReshapeFunc(reshape);
    glutDisplayFunc(Display);
    glutKeyboardFunc(Keypress);
    glutTimerFunc(100, move, 0);
    glutMainLoop();
    return 0;
}
```

CHAPTER 5
TESTING AND RESULTS

5.1 DIFFERENT TYPES OF TESTING

1. Unit Testing

Individual components are tested to ensure that they operate correctly. Each component is tested independently, without other system components.

2. Module Testing

A module is a collection of dependent components such as an object class, an abstract Data type or some looser collection of procedures and functions. A module related Components, so can be tested without other system modules.

3. System Testing

This is concerned with finding errors that result from unanticipated interaction between Sub-system interface problems.

4. Acceptance Testing

The system is tested with data supplied by the system customer rather than simulated test data.

5.2 TEST CASES

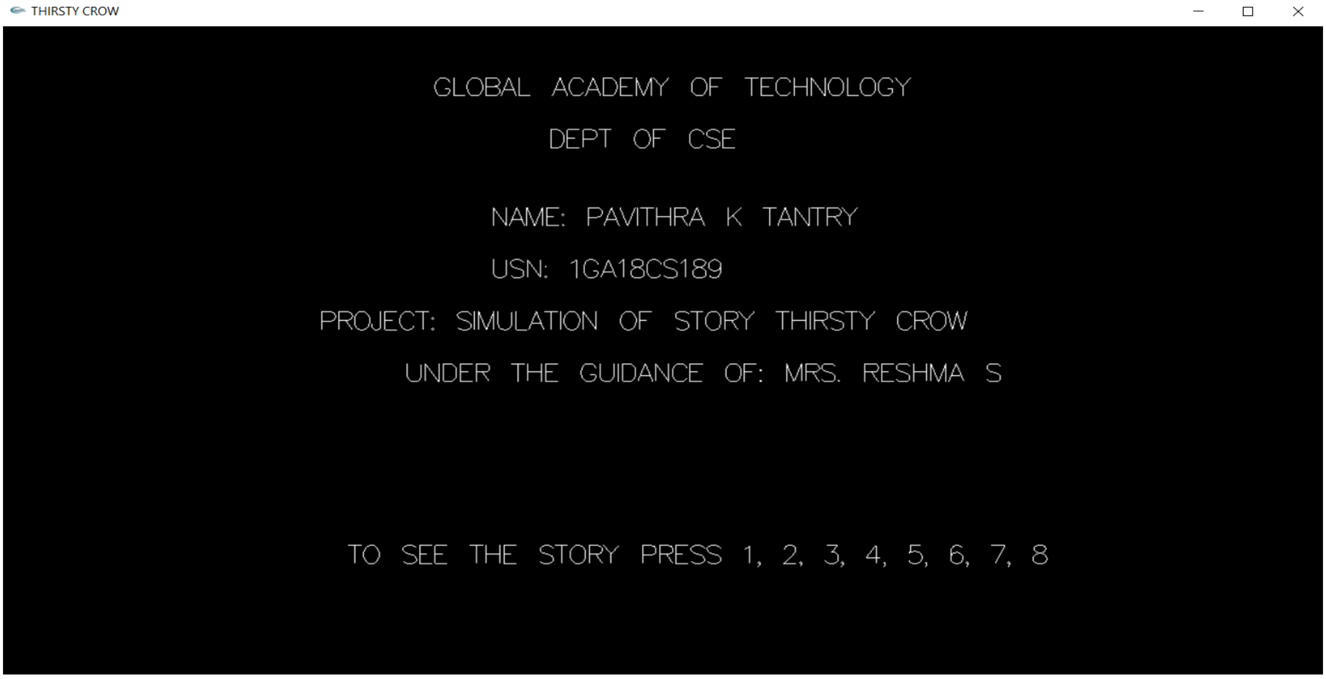
The test cases provided here test the most important features of the project.

Table 5.2.1: Test Case

Sl No	Test Input	Expected Results	Observed Results	Remarks
1	Press ‘1’	Display 1 to be displayed	Display 1 displayed	Pass
2	Press ‘2’	Display 2 to be displayed	Display 2 displayed	Pass
3	Press ‘3’	Display 3 to be displayed	Display 3 displayed	Pass
4	Press ‘4’	Display 4 to be displayed	Display 4 displayed	Pass
5	Press ‘5’	Display 5 to be displayed	Display 5 displayed	Pass
6	Press ‘6’	Display 6 to be displayed	Display 6 displayed	Pass
7	Press ‘7’	Display 7 to be displayed	Display 7 displayed	Pass
8	Press ‘8’	Moral of the story to be displayed	Moral of the story displayed	Pass

CHAPTER 6

SNAPSHOTS



GLOBAL ACADEMY OF TECHNOLOGY

DEPT OF CSE

NAME: PAVITHRA K TANTRY

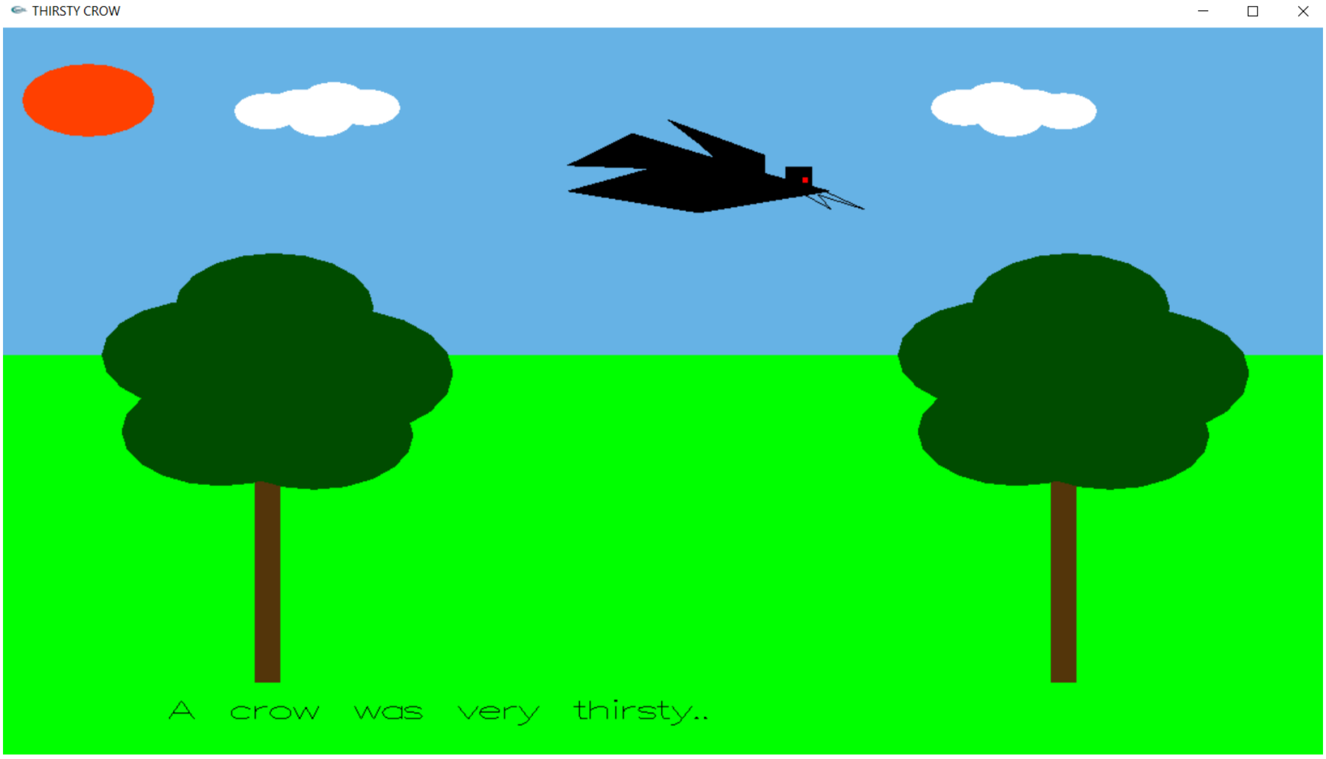
USN: 1GA18CS189

PROJECT: SIMULATION OF STORY THIRSTY CROW

UNDER THE GUIDANCE OF: MRS. RESHMA S

TO SEE THE STORY PRESS 1, 2, 3, 4, 5, 6, 7, 8

Figure 6.1: Introduction Screen



A crow was very thirsty..

Figure 6.2: Crow searching for water

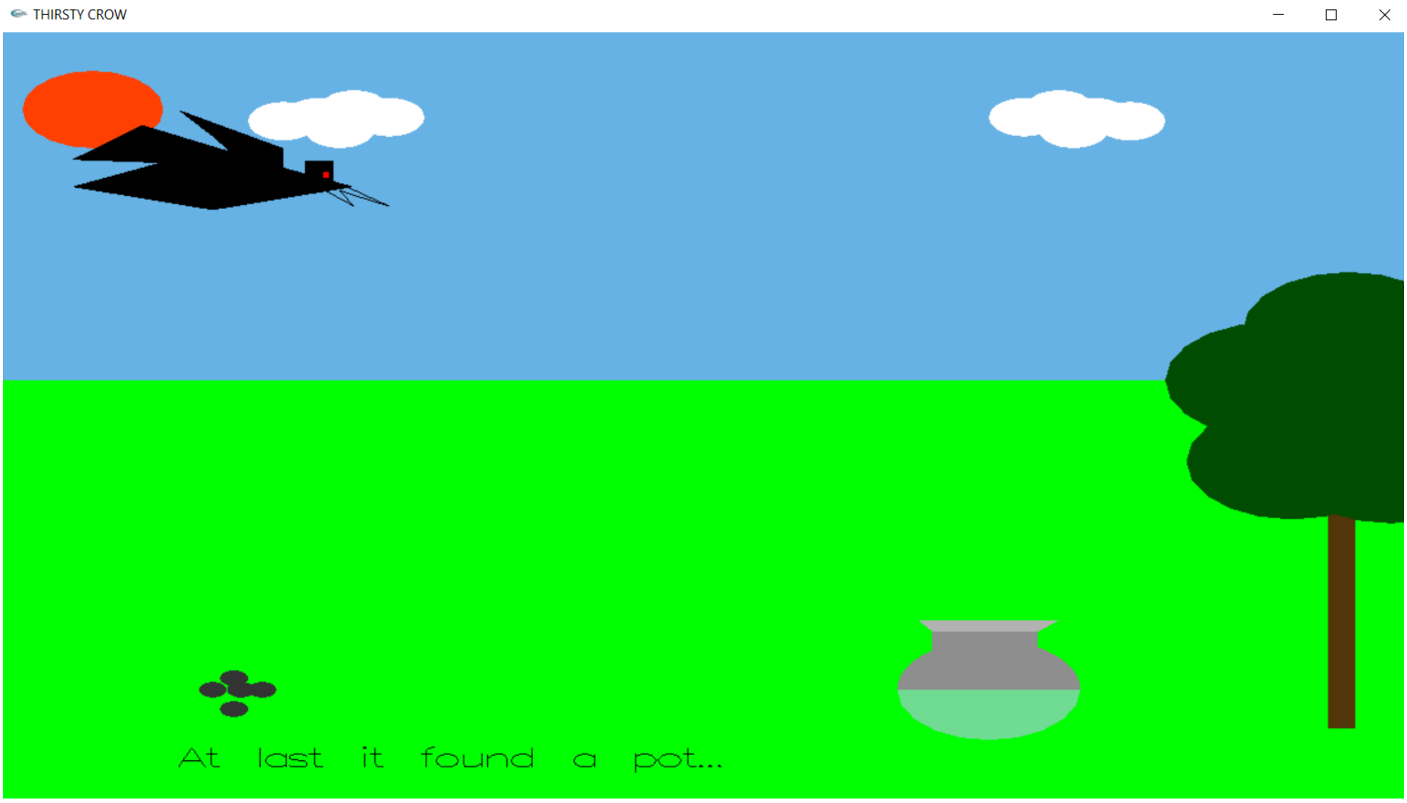


Figure 6.3: Crow finds pot of water

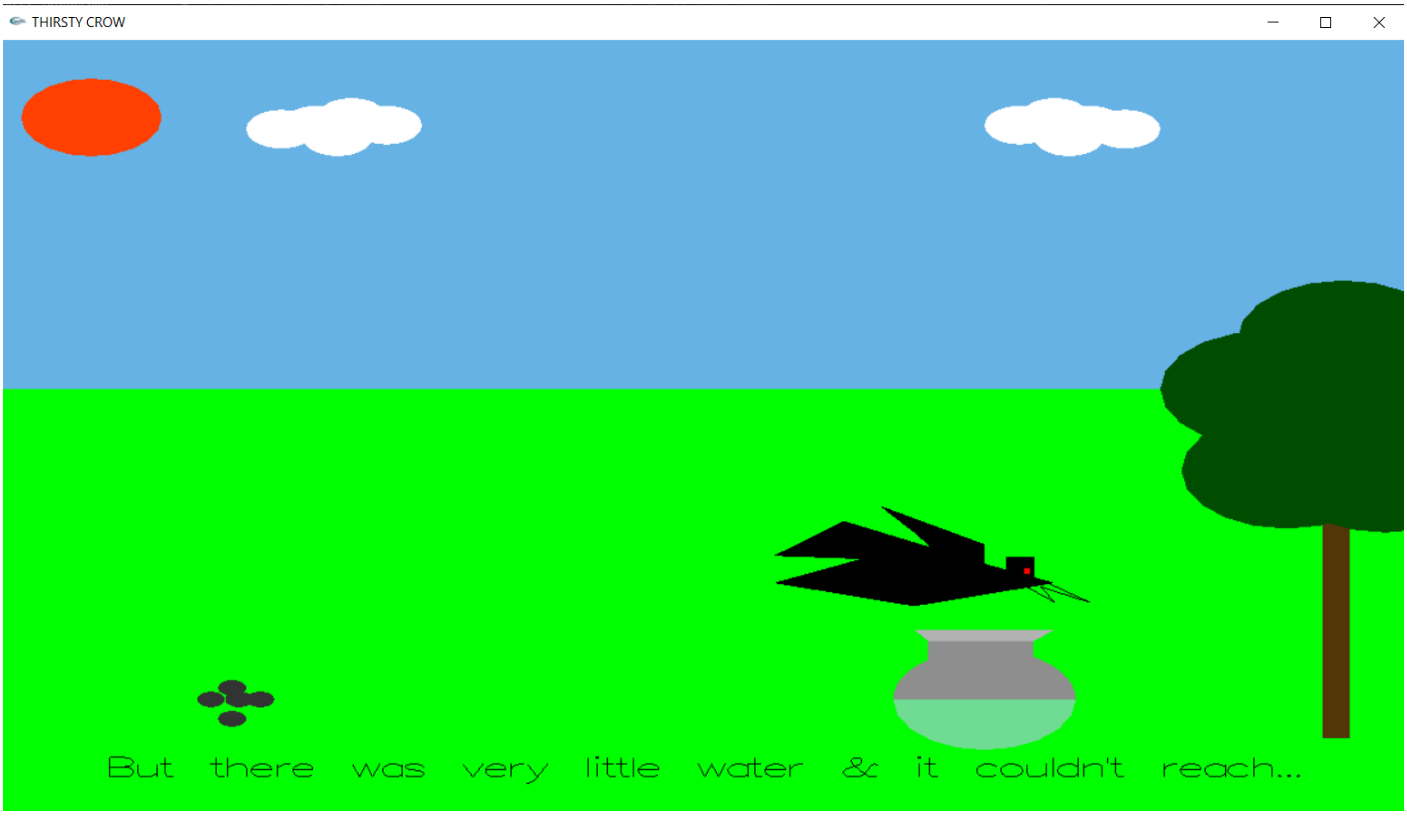
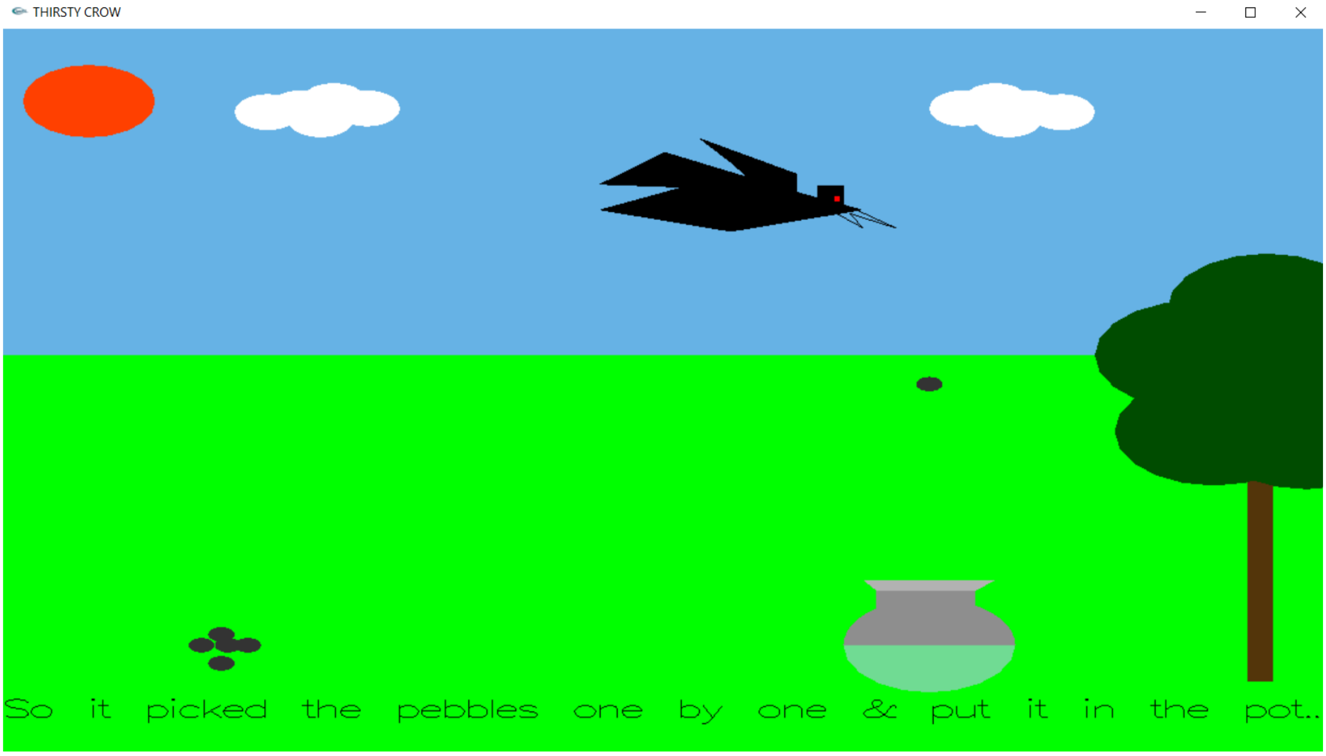
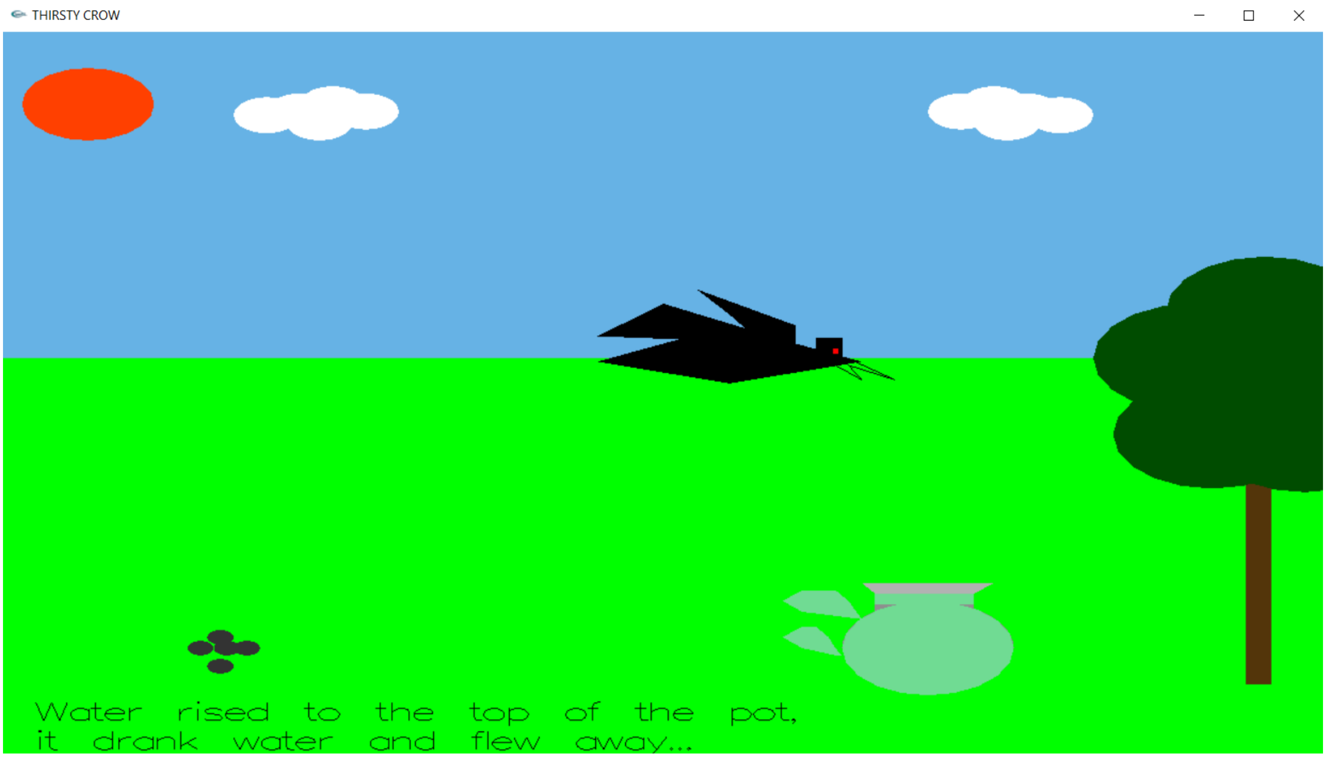


Figure 6.4: Crow can't reach water



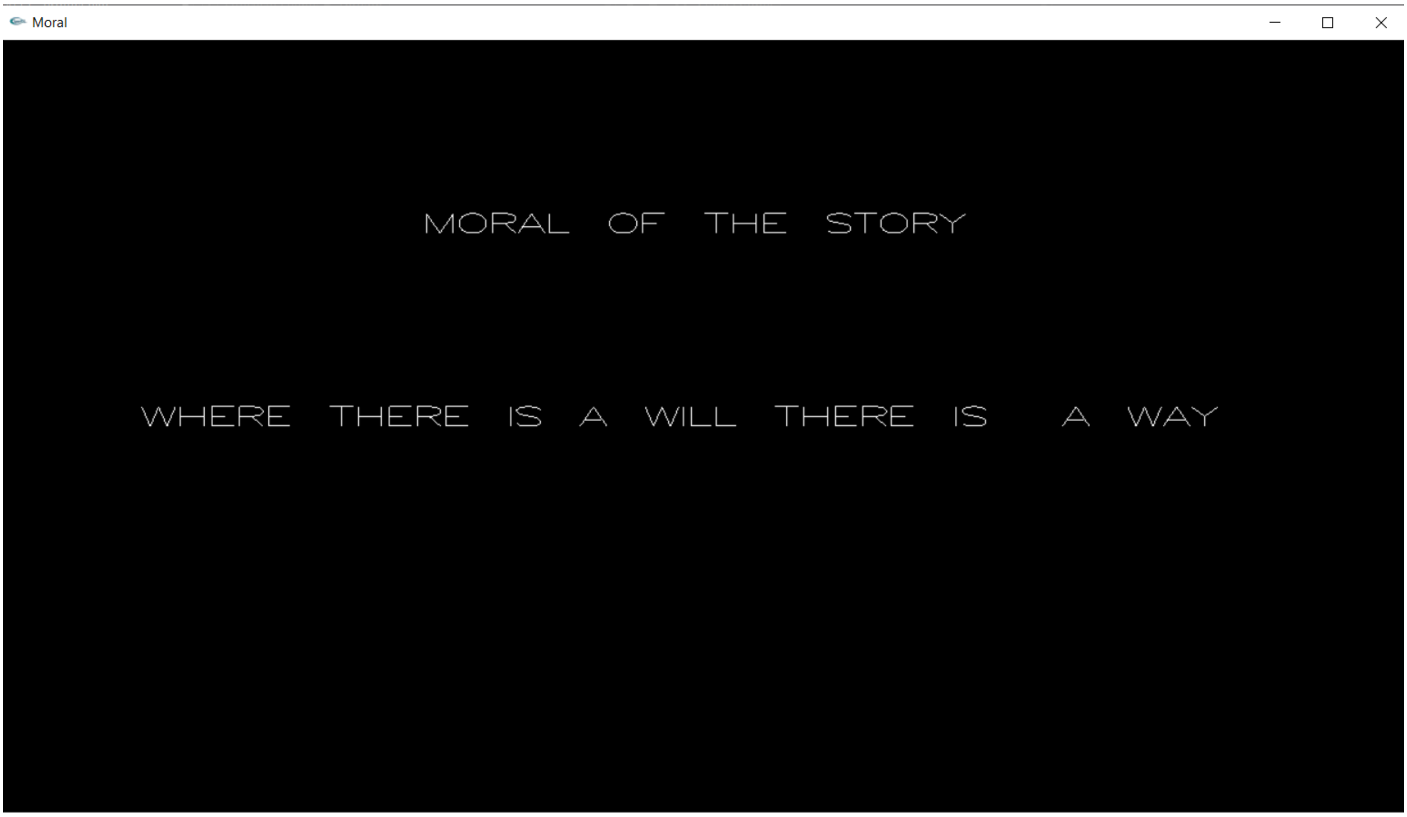
So it picked the pebbles one by one & put it in the pot..

Figure 6.5: Crow throwing pebbles



Water rised to the top of the pot,
it drank water and flew away...

Figure 6.6: Crow drinks water and flies away



Moral

MORAL OF THE STORY

WHERE THERE IS A WILL THERE IS A WAY

Figure 6.7: Moral of the story

CHAPTER 7

CONCLUSION

Simulation of the ‘story thirsty crow’ is designed and implemented using graphics software called OpenGL which has become widely accepted standard for developing graphic application. Open Graphics Library (OpenGL) is a cross-language, cross-platform application programming interface (API) for rendering 2D and 3D vector graphics. The development of the Simulation of this story has given a good exposure to OpenGL by which I have learnt some of the techniques.

The user-friendly interface allows the user to interact with it very effectively. So, I conclude on note that this project has given me a great exposure to the OpenGL and computer graphics. This is very reliable graphics package supporting various primitive objects like polygon, line loops, etc. Also, color selection, keyboard, menu and mouse-based interface are included. Transformations like translation, rotation, scaling is also provided.

CHAPTER 8

REFERENCES

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

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

[3]. https://open.gl/

[4]. https://en.wikipedia.org/wiki/FreeGLUT

[5]. https://stackoverflow.com