Bangladesh University of Business and Technology

Project Title: **Animated Story**

(Group 5)

Submitted By

Afrina Akter Mim (ID: 19202103310)

Rizvia Razzak Abani (ID: 19202103528)

Supervised By

Sweety Lima

Lecturer

Departement of Computer Science and Engineering

Bangladesh University of Business and Technology

Mirpur-2, Dhaka-1216

March 06, 2022

Acknowledgment

We would like to pay our gratitude to the Almighty Allah who created us with all the abilities to understand analysis and develop the process with patience. We are thankful to our project supervisor Sweety Lima, Lecturer, Computer Science and Engineering Department, Bangladesh University of Business and Technology for her professional guidance and motivation during the work of this project which is a major part of it. Without her valuable support and guidance, this project could not reach this level of development from our point of view.

We would like to thank all the Faculty members, Department of CSE, Bangladesh University of Business and Technology for their valuable time spend in requirements analysis and evaluation of the project work. We would like to express our sincere and warm gratitude to all those who have encouraged us directly, provided mental encouragement and criticized our work in several phases during the development of this project and for preparing this project indirectly.

Abstract

"Slow and Steady Wins the Race" is a classic animated story that teaches a valuable lesson about patience, perseverance, and the importance of hard work.We have made this animation video using GLUT in and this video is a great combination of difference type of algorithm such as DDA and circle drawing .The story follows a slow and steady tortoise who in a race against a fast and arrogant hare.This is animated video is a short time video but true meaningful. Despite being ridiculed by the hare and the other animals, the tortoise remains focused on his goal and steadily makes his way to the finish line. In the end, the tortoise's determination and steady pace lead him to victory, proving that slow and steady truly does win the race. Through its simple yet powerful narrative, the story emphasizes the value of persistence and the rewards that come from staying true to oneself and never giving up on one's dreams.we try to give a very important message to the society by creating this graphic design animated story.We implement a GLUT code which leads us to a very colorful animated video.The color used here are mainly associated with endurance motivation and strength.

Declaration

We hereby declare that the Project on **Animated Story** (Slow and steady wins the race) studies on Computer Graphics Lab submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering of Bangladesh University of Business and Technology (BUBT) is our own work and that it contains no material which has been accepted for the award to the candidate(s) of any other degree or diploma, except where due reference is made in the text of the project. To the best of our knowledge, it contains no materials previously published or written by any other person except where due reference is made in the project.

Afrina Akter Mim	Rizvia Razzak Abani
ID: 19202103310	ID: 19202103528

Copyright

Dedication

Dedicated to our parents, teachers, friends and who loved us for all their love and inspiration.

Certificate

This is to certify that Copyright by Afrina Akter Mim (ID: 19202103310) and Rizvia Razzak Abani (ID: 19202103528) were belong to the department of Computer Science and Engineering, have completed their Project on designing visual content Animated Story (Slow and steady wins the race) satisfactorily in partial fulfillment for the requirement of Bachelor of Science in Computer Science and Engineering of Bangladesh University of Business and Technology in the year 2023.

Supervisor
Sweety Lima
Lecturer
Department of Computer Science and Engineering
Bangladesh University of Business and Technology

Approval

A Project on Animated Story (Slow and steady wins the race) is submitted by Afrina Akter Mim (ID: 19202103310) and Rizvia Razzak Abani (ID: 19202103528) under the department of Computer Science and Engineering of Bangladesh University of Business and Technology is accepted in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering.

Supervisor
Sweety Lima
Lecturer
Department of Computer Science and Engineering
Bangladesh University of Business and Technology

Chairman
Md. Saifur Rahman
Assistant Professor & Chairman
Department of Computer Science and Engineering
Bangladesh University of Business and Technology

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

In this project we made a **Animated Story** of "Slow and Steady wins the race". We have grown up watching and hearing a lot of animated stories and in this project we will make a animated video clip on our very known story.

1.2 Motivation and Objectives

The goal of this project is to make a animated video. Objectives of our project is to,

- Make a dynamic background.
- Movement of elements to visualize the story.
- Show text on the video to tell what is happening.

1.3 Project Description

This is going to be a short animated video telling and showing the overall story which is created by using GLUT library and C++ language.

Chapter 2

Methodology

2.1 Algorithms

Figure drawing algorithms are techniques used in computer graphics to generate 2D or 3D images of human or animal figures. These algorithms use mathematical formulas to generate the shapes and movements of the figures, which can be animated or rendered in different styles and perspectives.

2.1.1 Line Drawing Algorithm

The line drawing algorithm using GL_LINE_LOOP is as follows:

1. Begin the glBegin() function with GL_LINE_LOOP as the parameter.
2. Specify the vertices of the line segments using glVertex*() functions, where * is the dimension of the vertex (2 for 2D, 3 for 3D, etc.). The vertices must be specified in a counter-clockwise order to ensure that the loop is closed properly.
3. End the loop with the glEnd() function.

2.1.2 Circle Drawing Algorithm

The algorithm to draw a circle using GL_POLYGON mode in OpenGL is written below:

1. Declare a variable for the number of points to be used in the circle.

2. Calculate the angle increment value for each point using the formula: (2.0 * PI)/(number of points).
3. Initialize the angle to zero.
4. Start the glBegin() function with GL_POLYGON as the parameter.
5. In a loop that iterates from zero to the number of points, calculate the x and y coordinates for each point on the circle using the formulae rad * cos(theta) and rad * sin(theta), respectively.
6. Use the glVertex2f() function to draw the current point on the circle.
7. Increment the angle by the angle increment value.
8. End the loop and glEnd() function to complete the circle.

2.1.3 Polygon Drawing Algorithm

Polygon drawing algorithm is the scanline algorithm, which works as follows:

1. Define the vertices of the polygon in a specified order.
2. Find the maximum and minimum y-coordinates of the polygon.
3. Begin scanning lines from the minimum y-coordinate to the maximum y-coordinate.
4. For each scanline, find the intersection points of the scanline with the edges of the polygon.
5. Sort the intersection points by their x-coordinate.
6. Begin drawing horizontal line segments between each pair of adjacent intersection points.
7. Fill the polygon by coloring in the pixels enclosed by the horizontal line segments.

Chapter 3

System Requirements

3.1 Software

3.1.1 Codeblock

Code::Blocks is an open-source Integrated Development Environment (IDE) for programming in C, C++, and other languages. It is available on Windows, Linux, and Mac OS. Code::Blocks provides an easy-to-use interface for writing and compiling code, debugging, and creating project files. The IDE supports a variety of compilers, including GCC and Visual C++, and includes a number of useful features like syntax highlighting, code folding, and project templates.

Figure 3.1: Code::Blocks

3.2 Library

3.2.1 GLUT

GLUT (OpenGL Utility Toolkit) is a free and open-source library for creating interactive 3D graphics applications. It provides a set of functions for creating windows, handling input events, and rendering 3D objects using OpenGL. GLUT simplifies the process of setting up a window and handling user input, allowing developers to focus on the graphics and animation aspects of their projects. Code::Blocks can be used with GLUT to develop 3D applications, games, and simulations. Together, Code::Blocks and GLUT provide a powerful platform for creating high-quality graphics applications for a variety of purposes.

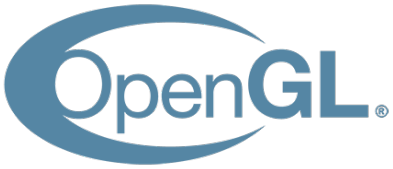

Figure 3.2: GLUT (OpenGL Utility Toolkit)

3.3 Languages

3.3.1 C++

C++ language is widely used in graphics design and programming due to its ability to interact directly with hardware, making it a powerful tool for creating high-performance graphics applications. In the field of computer graphics, C++ is used to create games, simulations, and other applications that require advanced graphics capabilities. Its object-oriented nature allows for the creation of modular and reusable code, making it easier to manage large and complex projects. C++ also provides access to libraries such as OpenGL and DirectX, which allow developers to create stunning 2D and 3D graphics with ease. Overall,C++ language plays a vital role in graphics design and programming, enabling developers to create visually appealing and interactive applications.

Figure 3.3: C++ language

Chapter 4

Implementation

4.1 Source Code

```
#ifdef _WIN32
#include <windows.h>
#else
#include <unistd.h>
#endif
#include <GL/glut.h>
#include<stdio.h>
#include <stdlib.h>
#include<windows.h>
#include<math.h>
#define PI 3.1416

GLfloat angle1 = 160.0;
GLfloat angle2 = 200.0;
GLfloat angle3 = 200.0;
float m=0;
float mount=0;
float mm=500;
float flag=0;
float flag2=500;
int F=0;
float b2_speed=5;
```

```
float ba_posion=0;
int counter=0;
float scale_cloud;
void sceenario(void);
void cloud();
void circle(GLdouble rad);
void drawstring(float x, float y, float z, char *string);
void tortoise();
void hare();
void hare_sleep();
void hare_walking();

void drawstring(float x, float y, float z, char *string)
{
    char *ct;
    glRasterPos3f(x,y,z);
    for(ct=string; *ct!='\0'; ct++)
    {
        glColor3f(0.0, 0.0, 0.0);
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *ct);
    }
}

void tortoise()
{
    /**----------------Head----------------**/
    glPushMatrix();
    glColor3f(0,.5,0);
    glScalef(.5,.7,.5);
    glTranslatef(110+350,245,1);
    circle(20);
    glPopMatrix();
    /**----------------Eyes----------------**/
    glPushMatrix();
    glColor3f(0,0,0);
    glScalef(.5,.7,.5);
    glTranslatef(110+350,245,1);
```

```
circle(5);
glPopMatrix();
/**----------Neck----------**/
glPushMatrix();
glColor3f(0,.5,0);
glScalef(.5,.7,.5);
glTranslatef(210+350,180,1);
glRotatef(20,0,0,1);
glBegin(GL_POLYGON);
glVertex3f(-70,90,1);
glVertex3f(-70,50,1);
glVertex3f(-50,50,1);
glVertex3f(-50,90,1);
glEnd();
glPopMatrix();
/**----------Leg1----------**/
glPushMatrix();
glColor3f(0,.7,0);
glScalef(.5,.7,.5);
glTranslatef(230+350,110,1);
glBegin(GL_POLYGON);
glVertex3f(-80,90,1);
glVertex3f(-85,70,1);
glVertex3f(-45,70,1);
glVertex3f(-50,90,1);
glEnd();
glPopMatrix();
/**----------LEg2----------**/
glPushMatrix();
glColor3f(0,.7,0);
glScalef(.5,.7,.5);
glTranslatef(300+350,110,1);
glBegin(GL_POLYGON);
glVertex3f(-80,90,1);
glVertex3f(-85,70,1);
glVertex3f(-45,70,1);
glVertex3f(-50,90,1);
```

```
    glEnd();
    glPopMatrix();
    /**----------Body----------**/
    glPushMatrix();
    float radius = 70;
    float twoPI = 2 * PI;
    glColor3f(0.5,0.2,0.1);
    glScalef(.5,0.7,.5);
    glTranslatef(200+350,200,1);
    glRotatef(-90,0,0,1);
    glBegin(GL_TRIANGLE_FAN);
    for (float i = PI; i <= twoPI; i += 0.001) {
        glVertex2f((sin(i)*radius), (cos(i)*radius)); }

    glEnd();
    glPopMatrix();
}
void hare()
{
    /**----------Head----------**/
    glPushMatrix();
    glColor3f(1,1,1);
    glScalef(.5,.7,.5);
    glTranslatef(110+360+2*m,245+100,1);
    circle(20);
    glPopMatrix();
    /**----------Eyes----------**/
    glPushMatrix();
    glColor3f(1,0,0);
    glScalef(.5,.7,.5);
    glTranslatef(110+350+2*m,245+100,1);
    circle(5);
    glPopMatrix();
    /**----------Neck----------**/
    glPushMatrix();
    glColor3f(1,1,1);
    glScalef(.5,.7,.5);
```

```
glTranslatef(210+360+2*m,180+100,1);
glRotatef(20,0,0,1);
glBegin(GL_POLYGON);
glVertex3f(-70,90,1);
glVertex3f(-70,50,1);
glVertex3f(-50,50,1);
glVertex3f(-50,90,1);
glEnd();
glPopMatrix();
/**----------Leg1----------**/
glPushMatrix();
glColor3f(1,1,1);
glScalef(.5,.7,.5);
glTranslatef(230+350+2*m,110+100,1);
glBegin(GL_POLYGON);
glVertex3f(-80,90,1);
glVertex3f(-85,70,1);
glVertex3f(-45,70,1);
glVertex3f(-50,90,1);
glEnd();
glPopMatrix();
/**----------LEg2----------**/
glPushMatrix();
glColor3f(1,1,1);
glScalef(.5,.7,.5);
glTranslatef(300+345+2*m,110+100,1);
glBegin(GL_POLYGON);
glVertex3f(-80,90,1);
glVertex3f(-85,70,1);
glVertex3f(-45,70,1);
glVertex3f(-50,90,1);
glEnd();
glPopMatrix();
/**----------Body----------**/
glPushMatrix();
glColor3f(1,1,1);
glScalef(.5,.7,.5);
```

```
glTranslatef(300+200+2*m,120+200,1);
circle(20);
glPopMatrix();
glPushMatrix();
glColor3f(1,1,1);
glScalef(.5,.7,.5);
glTranslatef(300+270+2*m,110+215,1);
circle(35);
glPopMatrix();

glPushMatrix();
glColor3f(1,1,1);
glScalef(.5,.7,.5);
glTranslatef(300+350+2*m,110+100,1);
glBegin(GL_POLYGON);
glVertex3f(-80,150,1);
glVertex3f(-85,80,1);
glVertex3f(-145,80,1);
glVertex3f(-150,130,1);
glEnd();
glPopMatrix();
/**----------tail----------**/
glPushMatrix();
glColor3f(1,1,1);
glScalef(.5,.7,.5);
glTranslatef(300+350+2*m,110+100,1);
glBegin(GL_TRIANGLES);
glVertex2i(-55,110);
glVertex2i(-30,120);
glVertex2i(-55,90);
glEnd();
glPopMatrix();
/**----------ear----------**/
glPushMatrix();
glColor3f(1,1,1);
glScalef(.5,.7,.5);
glTranslatef(300+235+2*m,110+145,1);
```

```
    glBegin(GL_TRIANGLES);
    glVertex2i(-55,110);
    glVertex2i(-25,120);
    glVertex2i(-55,90);
    glVertex2i(-55,110);
    glVertex2i(-25,135);
    glVertex2i(-55,90);
    glEnd();
    glPopMatrix();
}
void hare_walking()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(0.7,1,0);
    glPushMatrix();
    glTranslatef(200, 250, 0);
    glScalef(2,2,2);
    /**----------Head----------**/
    glBegin(GL_POLYGON);
    glVertex2i(-2,0);
    glVertex2i(-2,2);
    glVertex2i(0,4);
    glVertex2i(3,7);
    glVertex2i(5,8);
    glVertex2i(7,8);
    glVertex2i(9,7);
    glVertex2i(11,10);
    glVertex2i(13,12);
    glVertex2i(15,13);
    glVertex2i(18,14);
    glVertex2i(20,14);
    glVertex2i(21,13);
    glVertex2i(20,12);
    glVertex2i(14,8);
    glVertex2i(25,12);
    glVertex2i(26,12);
    glVertex2i(27,11);
```

```
    glVertex2i(27,10);
    glVertex2i(25,8);
    glVertex2i(22,5);
    glVertex2i(15,3);
    glVertex2i(14,1);
    glVertex2i(7,-2);
    glVertex2i(5,-3);
    glVertex2i(1,-2);
    glEnd();

    glBegin(GL_POLYGON);
    glVertex2i(48,-15);
    glVertex2i(50,-5);
    glVertex2i(47,0);
    glVertex2i(43,5);
    glVertex2i(40,7);
    glVertex2i(38,8);
    glVertex2i(35,8);
    glVertex2i(30,7);
    glVertex2i(25,5);
    glVertex2i(14,1);
    glVertex2i(7,-2);
    glVertex2i(10,-5);
    glVertex2i(13,-12);
    glVertex2i(15,-14);
    glVertex2i(19,-18);
    glVertex2i(25,-20);
    glVertex2i(40,-20);
    glVertex2i(43,-15);
    glEnd();

    glPopMatrix();
    glFlush();
}
void hare_sleep()
{
    /**----------Head----------**/
```

```
glPushMatrix();
glColor3f(1,1,1);
glScalef(.5,.5,1);
glTranslatef(110+60-2*m,640,1);
circle(20);
glPopMatrix();
/**----------Eyes----------**/
glPushMatrix();
glColor3f(0.5,0.5,0.5);
glScalef(.5,.5,1);
glTranslatef(110+50-2*m,640,1);
circle(2);
glPopMatrix();
/**----------Body----------**/
glPushMatrix();
glColor3f(1,1,1);
glScalef(.5,.5,1);
glTranslatef(200-2*m,620,1);
circle(20);
glPopMatrix();

glPushMatrix();
glColor3f(1,1,1);
glScalef(.5,.5,1);
glTranslatef(260-2*m,630,1);
circle(30);
glPopMatrix();

glPushMatrix();
glColor3f(1,1,1);
glBegin(GL_POLYGON);
glVertex3f(100-m,320,1);
glVertex3f(100-m,300,1);
glVertex3f(130-m,300,1);
glVertex3f(130-m,330,1);
glEnd();
glPopMatrix();
```

```
    /**----------tail----------**/
    glPushMatrix();
    glColor3f(1,1,1);
    glScalef(.5,.5,1);
    glTranslatef(340-2*m,220+300,1);
    glBegin(GL_TRIANGLES);
    glVertex2i(-55,110);
    glVertex2i(-30,120);
    glVertex2i(-55,90);
    glEnd();
    glPopMatrix();
    /**----------ear----------**/
    glPushMatrix();
    glColor3f(1,1,1);
    glScalef(.5,.5,1);
    glTranslatef(235-2*m,400+145,1);
    glBegin(GL_TRIANGLES);
    glVertex2i(-55,110);
    glVertex2i(-25,120);
    glVertex2i(-55,90);
    glVertex2i(-55,110);
    glVertex2i(-25,135);
    glVertex2i(-55,90);
    glEnd();
    glPopMatrix();

    glPushMatrix();
    glColor3f(0,0,0);
    glTranslatef(-m,0,1);
    drawstring(55, 320, 0.0, "Z Z");
    drawstring(45, 340, 0.0, "Z Z");
    drawstring(35, 360, 0.0, "Z Z");
    glPopMatrix();
}
void myInit(void)
{
    glColor3f(1.0f,0.0f,0.0f);
```

```
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0, 640.0, 0.0, 480.0);
}
void sceenario()
{
    /**----------road----------**/
    glColor3f(0.3,0.3,0.4);
    glPushMatrix();
    glTranslatef(0, 0, 0);
    glBegin(GL_POLYGON);
    glVertex2i(0,20);
    glVertex2i(700,20);
    glVertex2i(700,300);
    glVertex2i(0,300);
    glEnd();
    glPopMatrix();
    /**----------border grass----------**/
    glColor3f(0.5,0.8,0.4 );
    glPushMatrix();
    glTranslatef(0, 0, 0);
    glBegin(GL_POLYGON);
    glVertex2i(0,20);
    glVertex2i(700,20);
    glVertex2i(700,0);
    glVertex2i(0,0);
    glEnd();
    glPopMatrix();
    /**----------side border----------**/
    glColor3f(1,1,1);
    glPushMatrix();
    glTranslatef(0, 0, 0);
    glBegin(GL_POLYGON);
    glVertex2i(0,30);
    glVertex2i(700,30);
    glVertex2i(700,20);
    glVertex2i(0,20);
    glEnd();
```

```
glPopMatrix();

glColor3f(1,1,1);
glPushMatrix();
glTranslatef(0, 0, 0);
glBegin(GL_POLYGON);
glVertex2i(0,300);
glVertex2i(700,300);
glVertex2i(700,290);
glVertex2i(0,290);
glEnd();
glPopMatrix();
/**----------middle border----------**/
glColor3f(1,0.9,0.2);
glPushMatrix();
glTranslatef(0, 0, 0);
glBegin(GL_POLYGON);
glVertex2i(0,160);
glVertex2i(700,160);
glVertex2i(700,155);
glVertex2i(0,155);
glEnd();
glPopMatrix();

glColor3f(1,0.9,0.2);
glPushMatrix();
glTranslatef(0, 0, 0);
glBegin(GL_POLYGON);
glVertex2i(0,170);
glVertex2i(700,170);
glVertex2i(700,165);
glVertex2i(0,165);
glEnd();
glPopMatrix();
/**----------leftmost tree----------**/
glColor3f(0.8,0.5,0.1);
glPushMatrix();
```

```
glTranslatef(-m, 0, 0);
glBegin(GL_POLYGON);
glVertex2i(72,300);
glVertex2i(78,300);
glVertex2i(78,405);
glVertex2i(72,405);
glEnd();

glColor3f(0.1,0.5,0.1);
glBegin(GL_TRIANGLES);
glVertex2i(75,450);
glVertex2i(100,390);
glVertex2i(50,390);

glVertex2i(75,420);
glVertex2i(100,360);
glVertex2i(50,360);

glVertex2i(75,400);
glVertex2i(100,340);
glVertex2i(50,340);

glEnd();
glPopMatrix();
/**----------------middle tree----------------**/
glPushMatrix();
glColor3f(0.8,0.5,0.2);
glTranslatef(150-m, 150, 0);
glScalef(.5,.5,.5);
glBegin(GL_POLYGON);
glVertex2i(72,300);
glVertex2i(78,300);
glVertex2i(78,405);
glVertex2i(72,405);
glEnd();

glColor3f(.1,0.5,0.1);
```

```
glBegin(GL_POLYGON);
glVertex2i(75,450);
glVertex2i(100,390);
glVertex2i(50,390);

glEnd();
glBegin(GL_POLYGON);
glVertex2i(75,420);
glVertex2i(100,360);
glVertex2i(50,360);

glEnd();
glBegin(GL_POLYGON);
glVertex2i(75,400);
glVertex2i(100,340);
glVertex2i(50,340);

glEnd();
glPopMatrix();
/**----------------rightmost tree----------------**/
glPushMatrix();
glTranslatef(-m, 0, 0);
glColor3f(0.8,0.5,0.3);
glTranslatef(250, 0, 0);
glBegin(GL_POLYGON);
glVertex2i(72,300);
glVertex2i(78,300);
glVertex2i(78,405);
glVertex2i(72,405);
glEnd();

glColor3f(.1,0.5,0.1);
glBegin(GL_POLYGON);
glVertex2i(75,450);
glVertex2i(100,390);
glVertex2i(50,390);
```

```
    glEnd();
    glBegin(GL_POLYGON);
    glVertex2i(75,420);
    glVertex2i(100,360);
    glVertex2i(50,360);

    glEnd();
    glBegin(GL_POLYGON);
    glVertex2i(75,400);
    glVertex2i(100,340);
    glVertex2i(50,340);

    glEnd();
    glPopMatrix();

}
void mountain()
{
    /**-----------Sky-----**/
    glPushMatrix();
    glTranslatef(0, 0, 0);
    glBegin(GL_POLYGON);
    glColor3f(.8,.89,1);
    glVertex2i(0,350);

    glColor3f(.8,.89,1);
    glVertex2i(700,350);

    glColor3f(.2,.58,1);
    glVertex2i(700,500);

    glColor3f(.2,.58,1);
    glVertex2i(0,500);
    glEnd();
    glPopMatrix();
    /**-----------Grass-----------**/
    glColor3f(0.5,0.8,0.4);
```

```
glPushMatrix();
glTranslatef(0, 0, 0);
glBegin(GL_POLYGON);
glVertex2i(0,350);
glVertex2i(700,350);
glVertex2i(700,300);
glVertex2i(0,300);
glEnd();
glPopMatrix();

glPushMatrix();
int k=1,i;
/**----------Mountains----------**/
for (i=0; i<10; i++)
{
    glTranslatef(+mount*k-600*i*.5, -10, 0);
    k=k*.05;
    cloud();
    glColor3f(0.3,0.7,0.1);
    glBegin(GL_TRIANGLES);
    glVertex2i(0,360);
    glVertex2i(50,400);
    glVertex2i(100,360);

    glVertex2i(80,350);
    glVertex2i(200,450);
    glVertex2i(300,350);

    glVertex2i(280,360);
    glVertex2i(360,500);
    glVertex2i(450,360);

    glVertex2i(480,340);
    glVertex2i(560,430);
    glVertex2i(640,340);
    glEnd();
}
```

```
    glPopMatrix();
}
void cloud()
{
    float scale_cloud=.3;
    int trns_x=-1000;
    int trns_y=1150;
    glColor3f(0.8,.9,1);
    /**----------------------Cloud 1-------------------**/
    glPushMatrix();
    glScalef(scale_cloud,scale_cloud,scale_cloud);
    glTranslatef(250+trns_x,360+trns_y,0);
    circle(60);
    glPopMatrix();

    glPushMatrix();
    glScalef(scale_cloud,scale_cloud,scale_cloud);
    glTranslatef(260+trns_x,400+trns_y,0);
    circle(60);
    glPopMatrix();

    glPushMatrix();
    glScalef(scale_cloud,scale_cloud,scale_cloud);

    glTranslatef(300+trns_x,370+trns_y,0);
    circle(50);
    glPopMatrix();

    glPushMatrix();
    glScalef(scale_cloud,scale_cloud,scale_cloud);
    glTranslatef(330+trns_x,450+trns_y,0);
    circle(50);
    glPopMatrix();



    glPushMatrix();
    glScalef(scale_cloud,scale_cloud,scale_cloud);
```

```
glTranslatef(350+trns_x,350+trns_y,0);
circle(40);
glPopMatrix();

glPushMatrix();
glScalef(scale_cloud,scale_cloud,scale_cloud);
glTranslatef(365+trns_x,400+trns_y,0);
circle(40);
glPopMatrix();

glPushMatrix();
glScalef(scale_cloud,scale_cloud,scale_cloud);
glTranslatef(370+trns_x,430+trns_y,0);
circle(40);
glPopMatrix();

glPushMatrix();
glScalef(scale_cloud,scale_cloud,scale_cloud);
glTranslatef(390+trns_x,350+trns_y,0);
circle(40);
glPopMatrix();



glPushMatrix();
glScalef(scale_cloud,scale_cloud,scale_cloud);
glTranslatef(390+trns_x,400+trns_y,0);
circle(40);
glPopMatrix();
/**----------------------Cloud 2----------------------**/
trns_x=-500;
glPushMatrix();
glScalef(scale_cloud,scale_cloud,scale_cloud);
glTranslatef(250+trns_x,360+trns_y,0);
circle(60);
glPopMatrix();

glPushMatrix();
```

```
glScalef(scale_cloud,scale_cloud,scale_cloud);
glTranslatef(260+trns_x,400+trns_y,0);
circle(60);
glPopMatrix();

glPushMatrix();
glScalef(scale_cloud,scale_cloud,scale_cloud);
glTranslatef(300+trns_x,370+trns_y,0);
circle(50);
glPopMatrix();

glPushMatrix();
glScalef(scale_cloud,scale_cloud,scale_cloud);
glTranslatef(330+trns_x,450+trns_y,0);
circle(50);
glPopMatrix();

glPushMatrix();
glScalef(scale_cloud,scale_cloud,scale_cloud);
glTranslatef(350+trns_x,350+trns_y,0);
circle(40);
glPopMatrix();

glPushMatrix();
glScalef(scale_cloud,scale_cloud,scale_cloud);
glTranslatef(365+trns_x,400+trns_y,0);
circle(40);
glPopMatrix();

glPushMatrix();
glScalef(scale_cloud,scale_cloud,scale_cloud);
glTranslatef(370+trns_x,430+trns_y,0);
circle(40);
glPopMatrix();

glPushMatrix();
glScalef(scale_cloud,scale_cloud,scale_cloud);
```

```
glTranslatef(390+trns_x,350+trns_y,0);
circle(40);
glPopMatrix();

glPushMatrix();
glScalef(scale_cloud,scale_cloud,scale_cloud);
glTranslatef(390+trns_x,400+trns_y,0);
circle(40);
glPopMatrix();
/**--------------------Cloud 3--------------------**/
trns_x=-1200;
glPushMatrix();  // tree leaves
glScalef(scale_cloud,scale_cloud,scale_cloud);
glTranslatef(250+trns_x,360+trns_y,0);
circle(60);
glPopMatrix();

glPushMatrix();
glScalef(scale_cloud,scale_cloud,scale_cloud);
glTranslatef(260+trns_x,400+trns_y,0);
circle(60);
glPopMatrix();

glPushMatrix();
glScalef(scale_cloud,scale_cloud,scale_cloud);
glTranslatef(300+trns_x,370+trns_y,0);
circle(50);
glPopMatrix();

glPushMatrix();
glScalef(scale_cloud,scale_cloud,scale_cloud);
glTranslatef(330+trns_x,450+trns_y,0);
circle(50);
glPopMatrix();

glPushMatrix();
glScalef(scale_cloud,scale_cloud,scale_cloud);
```

```
    glTranslatef(350+trns_x,350+trns_y,0);
    circle(40);
    glPopMatrix();

    glPushMatrix();
    glScalef(scale_cloud,scale_cloud,scale_cloud);
    glTranslatef(365+trns_x,400+trns_y,0);
    circle(40);
    glPopMatrix();

    glPushMatrix();
    glScalef(scale_cloud,scale_cloud,scale_cloud);
    glTranslatef(370+trns_x,430+trns_y,0);
    circle(40);
    glPopMatrix();

    glPushMatrix();
    glScalef(scale_cloud,scale_cloud,scale_cloud);
    glTranslatef(390+trns_x,350+trns_y,0);
    circle(40);
    glPopMatrix();

    glPushMatrix();
    glScalef(scale_cloud,scale_cloud,scale_cloud);
    glTranslatef(390+trns_x,400+trns_y,0);
    circle(40);
    glPopMatrix();
}
void sceenario2()
{
    /**----------------leftmost tree-------------------**/
    glPushMatrix();
    glColor3f(0.8,0.5,0.1);
    glTranslatef(-mm, 150, 0);
    glScalef(.5,.5,.5);
    glBegin(GL_POLYGON);
    glVertex2i(72,300);
```

```
glVertex2i(78,300);
glVertex2i(78,405);
glVertex2i(72,405);
glEnd();

glColor3f(.1,0.5,0.1);
glBegin(GL_POLYGON);
glVertex2i(75,450);
glVertex2i(100,390);
glVertex2i(50,390);

glEnd();
glBegin(GL_POLYGON);
glVertex2i(75,420);
glVertex2i(100,360);
glVertex2i(50,360);

glEnd();
glBegin(GL_POLYGON);
glVertex2i(75,400);
glVertex2i(100,340);
glVertex2i(50,340);

glEnd();
glPopMatrix();
/**----------------second leftmost tree----------------**/
glColor3f(0.8,0.5,0.3);
glPushMatrix();
glTranslatef(50-mm, 0, 0);
glBegin(GL_POLYGON);
glVertex2i(72,300);
glVertex2i(78,300);
glVertex2i(78,405);
glVertex2i(72,405);
glEnd();

glColor3f(.1,0.5,.1);
```

```
glBegin(GL_POLYGON);
glVertex2i(75,450);
glVertex2i(100,390);
glVertex2i(50,390);

glEnd();
glBegin(GL_POLYGON);
glVertex2i(75,420);
glVertex2i(100,360);
glVertex2i(50,360);

glEnd();
glBegin(GL_POLYGON);
glVertex2i(75,400);
glVertex2i(100,340);
glVertex2i(50,340);

glEnd();
glPopMatrix();
/**----------------second rightmost tree----------------**/
glPushMatrix();
glColor3f(0.8,0.5,0.2);
glTranslatef(150-mm, 150, 0);
glScalef(.5,.5,.5);
glBegin(GL_POLYGON);
glVertex2i(72,300);
glVertex2i(78,300);
glVertex2i(78,405);
glVertex2i(72,405);
glEnd();

glColor3f(.1,0.5,0.1);
glBegin(GL_POLYGON);
glVertex2i(75,450);
glVertex2i(100,390);
glVertex2i(50,390);
```

```
glEnd();
glBegin(GL_POLYGON);
glVertex2i(75,420);
glVertex2i(100,360);
glVertex2i(50,360);

glEnd();
glBegin(GL_POLYGON);
glVertex2i(75,400);
glVertex2i(100,340);
glVertex2i(50,340);

glEnd();
glPopMatrix();
/**----------------rightmost tree----------------**/
glPushMatrix();
glColor3f(0.8,0.5,0.4);
glTranslatef(200-mm, 0, 0);
glBegin(GL_POLYGON);
glVertex2i(72,300);
glVertex2i(78,300);
glVertex2i(78,405);
glVertex2i(72,405);
glEnd();

glColor3f(.1,0.5,.1);
glBegin(GL_POLYGON);
glVertex2i(75,450);
glVertex2i(100,390);
glVertex2i(50,390);

glEnd();
glBegin(GL_POLYGON);
glVertex2i(75,420);
glVertex2i(100,360);
glVertex2i(50,360);
```

```
    glEnd();
    glBegin(GL_POLYGON);
    glVertex2i(75,400);
    glVertex2i(100,340);
    glVertex2i(50,340);

    glEnd();
    glPopMatrix();
}
void my_control_func()
{
    counter++;
    if(counter>16800||counter<3800)
    {
        m=m;
        mm=mm;
        mount=mount;
    }
    else
    {
        mount+=.2;
        if(flag>-800)
        {
            m-=.2;
            flag-=.2;
        }
        else
        {
            m=350;
            flag=700;
            F++;
        }
        if(flag2>-1200)
        {
            mm-=.2;
            flag2-=.2;
        }
```

```
        else
        {
            mm=350;
            flag2=700;
        }
        if (F==2)
        {
            b2_speed=10;
        }
    }
}
void starting()
{
    /**----------Starting Mark----------**/
    glColor3f(0.8,0.1,0.3);
    glPushMatrix();
    glTranslatef(200-m, 00, 0);
    glBegin(GL_POLYGON);
    glVertex2i(10,20);
    glVertex2i(20,20);
    glVertex2i(20,300);
    glVertex2i(10,300);
    glEnd();
    glPopMatrix();
}
void ending()
{
    /**----------Ending Mark----------**/
    glColor3f(0.8,0.1,0.3);
    glPushMatrix();
    glTranslatef(400-m, 0, 0);
    glBegin(GL_POLYGON);
    glVertex2i(10,20);
    glVertex2i(20,20);
    glVertex2i(20,300);
    glVertex2i(10,300);
    glEnd();
```

```
    glPopMatrix();
}
void victory()
{
    /**--------------Victory signboard--------------**/
    glColor3f(0.4,0.5,0.9);
    glPushMatrix();
    glTranslatef(300-m, 25, 0);
    glBegin(GL_POLYGON);
    glVertex2i(10,100);
    glVertex2i(40,50);
    glVertex2i(40,250);
    glVertex2i(10,300);
    glEnd();
    glPopMatrix();
    /**--------------pole 1--------------**/
    glColor3f(.7,0.1,0.7);
    glPushMatrix();
    glTranslatef(300-m, 0, 0);
    glBegin(GL_POLYGON);
    glVertex2i(10,360);
    glVertex2i(50,290);
    glVertex2i(50,300);
    glVertex2i(10,370);
    glEnd();
    glPopMatrix();
    /**--------------Pole 2--------------**/
    glColor3f(.7,0.1,0.7);
    glPushMatrix();
    glTranslatef(300-m, -270, 0);
    glBegin(GL_POLYGON);
    glVertex2i(10,360);
    glVertex2i(50,290);
    glVertex2i(50,300);
    glVertex2i(10,370);
    glEnd();
    glPopMatrix();
```

```
    /**----------Rope1----------**/
    glColor3f(.7,0.9,0.7);
    glPushMatrix();
    glTranslatef(300-m, 0, 0);
    glBegin(GL_LINE_LOOP);
    glVertex2i(10,360);
    glVertex2i(10,325);
    glVertex2i(40,275);
    glVertex2i(50,290);
    glEnd();
    glPopMatrix();
    /**----------Rope----------2 **/
    glColor3f(.7,0.9,0.7);
    glPushMatrix();
    glTranslatef(300-m, -260, 0);
    glBegin(GL_LINE_LOOP);
    glVertex2i(10,360);
    glVertex2i(10,385);
    glVertex2i(40,335);
    glVertex2i(50,290);
    glEnd();
    glPopMatrix();
}
void texts()
{
    glColor3f(1.0, 1.0, 1.0);
    if(counter>400&& counter<800)
        drawstring(260, 265, 0.0, "Ha! Look at you, you slow and clumsy tortoise.");
    if(counter>800&& counter<1200)
        drawstring(260, 265, 0.0, "You'll never be able to beat me in a race.");

    if(counter>2400&& counter<2800)
        drawstring(260, 265, 0.0, "A race? You're kidding me, right?");
    if(counter>3200&& counter<3600)
        drawstring(260, 265, 0.0, "I'll beat you in a heartbeat.");

    if(counter>1200&& counter<1600)
```

```
        drawstring(250, 100, 0.0, "I may be slow, but I'm not clumsy.");
    if(counter>1600&& counter<2000)
        drawstring(250, 100, 0.0, "And I'm not afraid of a challenge");
    glColor3f(1.0, 1.0, 1.0);
    if(counter>2000&& counter<2400)
        drawstring(250, 100, 0.0,"How about a race, Mr. Hare?");
    if(counter>2800&& counter<3200)
        drawstring(250, 100, 0.0, "We'll see about that.");
    if (counter>17300)
    {
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(0.4,0.5,0.9);
        glPushMatrix();
        glTranslatef(0, 0, 0);
        glBegin(GL_POLYGON);
        glVertex2i(0,0);
        glVertex2i(700,0);
        glVertex2i(700,1350);
        glVertex2i(0,1350);
        glEnd();
        glPopMatrix();
        glColor3f(0,0,0);

        drawstring(250, 335, 0.0, "++++++++++++++++");
        glColor3f(0,0,0);
        drawstring(250, 350, 0.0, "+  Moral of the story +");
        glColor3f(0,0,0);
        drawstring(250, 365, 0.0, "++++++++++++++++");
        glColor3f(0,0,0);
        drawstring(235, 315, 0.0, "++++++++++++++++++++++++");
        glColor3f(0,0,0);
        drawstring(235, 300, 0.0, "+ Slow and steady wins the race +");
        glColor3f(0,0,0);
        drawstring(235, 285, 0.0, "++++++++++++++++++++++++");
        glColor3f(1,1,1);
        drawstring(500, 100, 0.0, "Presented by:");
        glColor3f(1,1,1);
```

```
        drawstring(500, 80, 0.0, "Afrina Akter Mim (ID: 19202103310)");
        glColor3f(1,1,1);
        drawstring(500, 60, 0.0, "Rizvia Razzak Abani (ID: 19202103528)");
        glColor3f(1,1,1);
        drawstring(500, 40, 0.0, "Int/Sec: 44/8, Dept of CSE,");
        glColor3f(1,1,1);
        drawstring(500, 20, 0.0, "BUBT.");
    }
}
void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    mountain();
    sceenario();
    sceenario2();
    if(F==0)
    {
        starting();
        hare();
    }
    if(F==1)
    {
        hare_sleep();
    }
    tortoise();
    if(F==2)
    {
        ending();
        tortoise();
        hare();
        victory();
        ba_posion=1500;
    }
    texts();
    glFlush();
    glutPostRedisplay();
    my_control_func();
```

```
}
int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(1350,700);
    glutCreateWindow("Slow and Steady wins the race");
    glutDisplayFunc(display);
    myInit();
    glutMainLoop();
}
int i;
void circle(GLdouble rad)
{
    GLint points = 50;
    GLdouble delTheta = (2.0*PI)/(GLdouble)points;
    GLdouble theta = 0.0;
    glBegin(GL_POLYGON);
    {
        for( i = 0; i <=50; i++, theta += delTheta )
        {
            glVertex2f(rad * cos(theta),rad * sin(theta));
        }
    }
    glEnd();
}
```

4.2 Snapshots

Conversation between the hare and the tortoise,

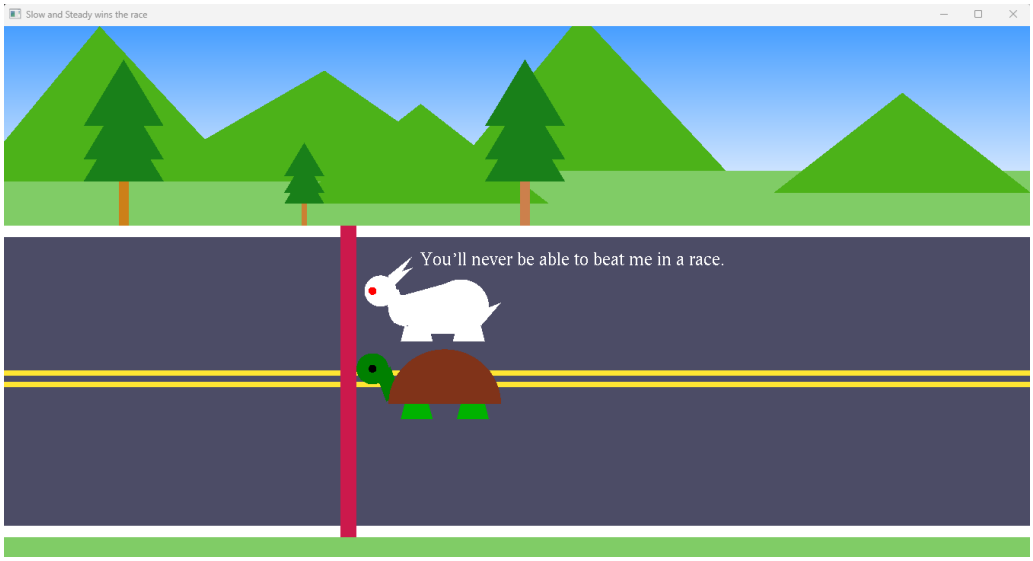

Figure 4.1: Scenario 1

Running tortoise and sleeping hare,

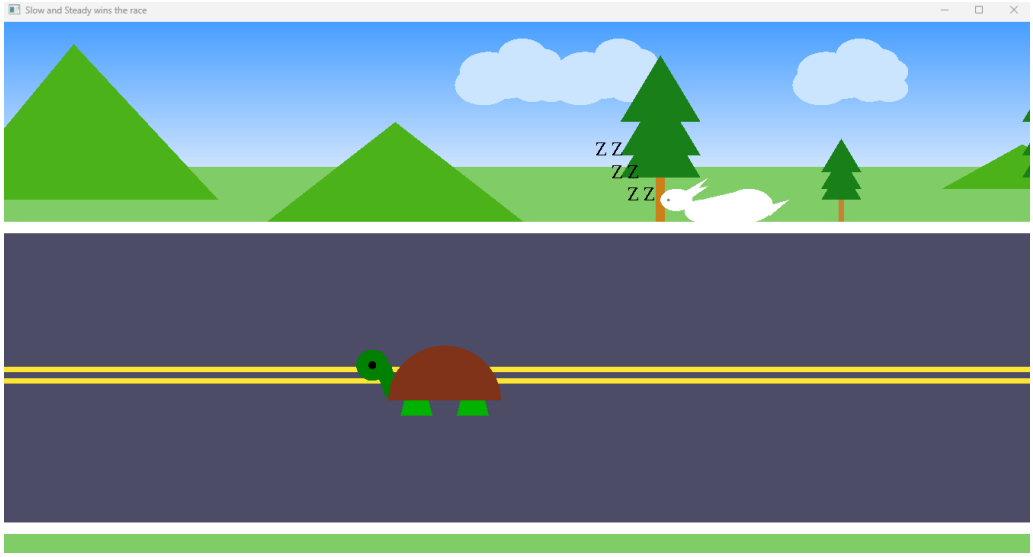

Figure 4.2: Scenario 2

Winning part,

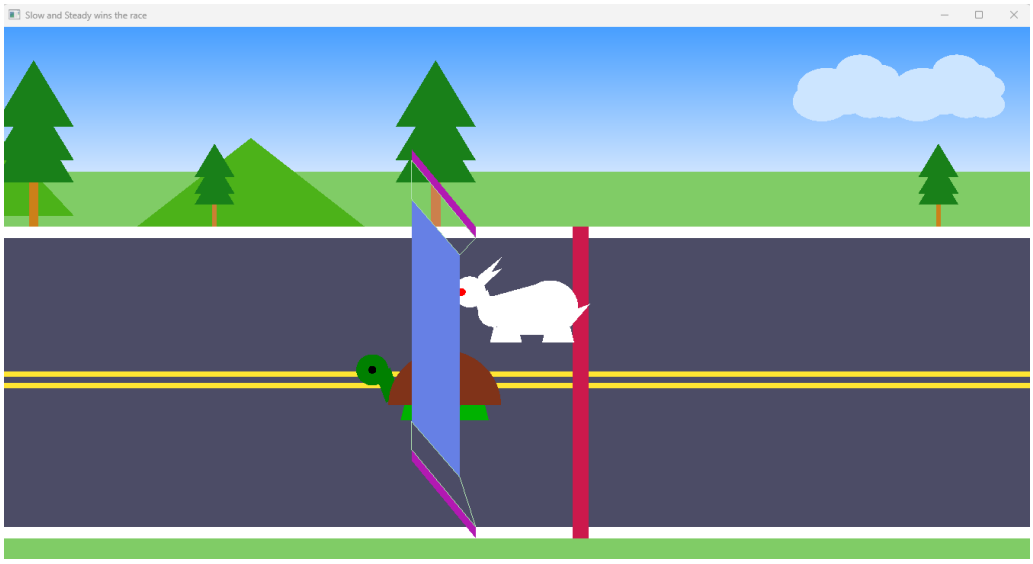

Figure 4.3: Scenario 3

Last scene,

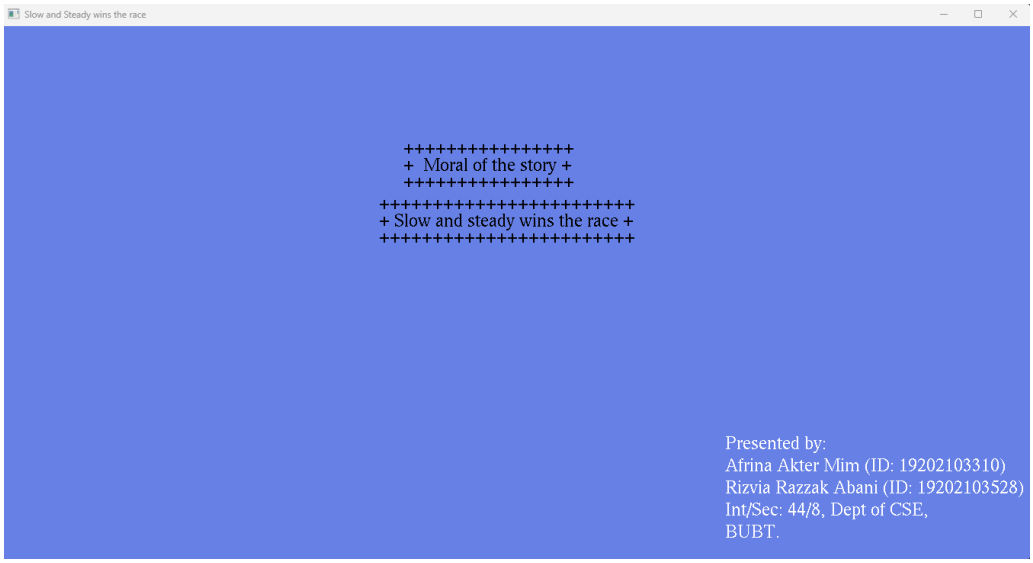

Figure 4.4: Scenario 4

Chapter 5

Conclusion

5.1 Conclusion

To conclude, we are going to implement our today's knowledge to make a a animated video on our very know childhood story. The timeless lesson of "slow and steady wins the race" is exemplified by the classic fable of the tortoise and the hare. This story reminds us that success is not always about speed or talent, but about the persistence, determination, and consistency that the tortoise displayed throughout the race. Through this project, we hope to inspire viewers to stay focused on their goals, even when progress seems slow or difficult. By embracing the spirit of the tortoise and the hare, we can learn to be patient, persistent, and determined in our own journey towards success.