VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"JNANA SANGAMA" , BELAGAVI – 590 018

A PROJECT REPORT
ON

"TREX GAME 3D"

Submitted by

ARJUN RAGHUVIR BHANDARI 4SF20CS017

KUMAR RISHAV 4SF20CS054

In the partial fulfilment of the requirement for VI Sem. B. E. (CSE)

COMPUTER GRAPHICS LABORATORY WITH MINI PROJECT (18CSL67)

Under the guidance of

Ms.Megha Rani R
Assistant Professor
Dept. of CSE

SAHYADRI
COLLEGE OF ENGINEERING & MANAGEMENT
(An Autonomous Institution)
Adyar, Mangaluru-575007
2022-23

SAHYADRI
COLLEGE OF ENGINEERING & MANAGEMENT
(An Autonomous Institution)
Adyar, Mangaluru – 575007

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the project entitled "**TREX GAME 3D"** is submitted in partial fulfilment for the requirement of VI sem. B.E. (Computer Science & Engineering) **"COMPUTER GRAPHICS LABORATORY WITH MINI PROJECTS (18CSL67)"** during the year 2022 – 2023 is a result of bonafide work carried out by

ARJUN RAGHUVIR BAHNDARI	**4SF20CS017**
KUMAR RISHAV	**4SF20CS054**

…………………………	..……………………….
Ms.Megha Rani R	**Dr.Nagesh H R**
Asst. Prof., Dept. of CS&E	HOD, Dept. Of CS&E
SCEM, Mangaluru	SCEM, Mangaluru

Signature of the Examiners

1. …………………………..

2. …………………………..

ABSTRACT

The TRex game 3D, also known as the Chrome No Internet Game, is a popular browser-based game that has captivated millions of users worldwide. This game was developed by Google as a fun and engaging way to entertain users when they encounter connection issues while browsing the internet using the Google Chrome browser. The game features a 3D rendering of a T-Rex dinosaur, and players are tasked with guiding the dinosaur through a treacherous landscape while avoiding various obstacles. The main objective of the TRex game 3D is to survive for as long as possible and achieve the highest score. The game starts with the T-Rex positioned at the left side of the screen, and as the game progresses, the dinosaur automatically begins to run at an increasing speed. The player's primary control is to make the dinosaur jump over obstacles such as cacti and flying pterodactyls. Timing and quick reflexes are essential to successfully navigate through the obstacles and continue the game. One of the key features that make the TRex game 3D highly addictive is its simplicity. The minimalist graphics, coupled with the fast-paced gameplay, create an engaging experience that appeals to players of all ages. The game's intuitive controls, which involve only a single input (either pressing the spacebar or tapping on the screen for a jump), make it easy for anyone to pick up and play. The project's purpose was to provide users with a source of entertainment during moments of internet downtime. Connection issues can be frustrating, but the TRex game 3D transforms this inconvenience into an opportunity for fun and diversion. The game's popularity has grown significantly since its release, with players worldwide embracing it as a way to pass the time and challenge themselves to achieve higher scores. In conclusion, the TRex game 3D is a browser-based game developed by Google that has become a widely beloved form of entertainment. Its simplicity, addictive gameplay, and ability to provide amusement during internet downtime have contributed to its massive popularity. By transforming a frustrating situation into an enjoyable experience, the TRex game 3D has solidified its place as a classic browser game and continues to captivate players worldwide.

ACKNOWLEDGEMENT

Before we get in-depth with the project, we want to include few expressions of appreciation for the people who has been a part of this project from its inception. The written work of this project has been one of the huge academic challenges we have faced and withoutthe help, patience and guidance of the people involved, this assignment would not have been completed satisfactorily.

It gives us immense pleasure in presenting this project report on “ **TREX GAME 3D** ”. It has been our privilege to have a project guide who had assisted us from the commencement of this project. The success of this project is a sheer diligent work, and determination put in by us with the help of our project guide.

We hereby take this chance to include a special note of much obliged for **Ms.Megha Rani R**, Assistant Professor, and Department of Computer Science who guided us in our project. We are additionally grateful to **Dr Nagesh H R**, Head of the Department, Computer Science and Engineering for furnishing us with the correct academic atmosphere inthe department, whose encouragement and support made our entire undertaking appreciable.

We are extremely thankful to our beloved Principal **Dr. Rajesha S** for encouraging us to come up with new ideas and to express them in a systematic manner.

We would also like to thank all our non-teaching staffs who also were very much supportive to us in building this project.

Last but not the least we want to extend our gratitude to our parents for their continuous love and support for which we are always indebted to them and also thank each and every one of those who helped or provided a helping hand in this project to be carried out.

ARJUN RAGHUVIR BHANDARI (4SF20CS017)

KUMAR RISHAV (4SF20CS054)

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Computer Graphics

Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D or 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly. [2] Computers have become a powerful medium for the rapid and economical production of pictures. Graphics provide a natural means of communicating with the computer that they have become widespread. Computer graphics system is a computer system with all the components of the general-purpose computer system. There are five major elements in a system: input devices, processor, memory, frame buffer, output devices. Today, computer graphics are widespread. Such imagery is found in and on television, newspapers, weather reports, and in a variety of medical investigations and surgical procedures.[3] A well-constructed graph can present complex statistics in a form that is easier to understand and interpret. Many tools have been developed to visualize data. Computer generated imagery can be categorized into several different types: two dimensional (2D), three dimensional (3D), and animated graphics. As technology has improved, 3D computer graphics have become more common, but 2D computer graphics are still widely used. Computer graphics has emerged as a sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content. Over the past decade, other specialized fields have been developed like information visualization, and scientific visualization more concerned with "the visualization of three-dimensional phenomena (architectural, meteorological, medical, biological, etc.), where the emphasis is on realistic renderings of volumes, surfaces, illumination sources, and so forth, perhaps with a dynamic (time) component". Computer graphics is responsible for displaying art and image data effectively and meaningfully to the consumer. It is also used for processing image data received from the physical world. Computer graphics development has had a significant impact on many types of media & revolutionized animation, movies, advertising, video games, and graphic design in general.

1.2 OpenGL

OpenGL is a low-level graphics library specification. It makes available to the programmer a small set of geometric primitives - points, lines, polygons, images, and bitmaps.[1] OpenGL provides a set of commands that allow the specification of geometric objects in two or three dimensions, using the provided primitives, together with commands that control how these objects are rendered (drawn). Since OpenGL drawing commands are limited to those that generate simple geometric primitives (points, lines, and polygons), the OpenGL Utility Toolkit (GLUT) has been created to aid in the development of more complicated three-dimensional objects such as a sphere, a torus, and even a teapot. GLUT may not be satisfactory for full-featured OpenGL applications, but it is a useful starting point for learning OpenGL. GLUT simplifies the implementation of programs using OpenGL rendering. The GLUT application programming interface (API) requires very few routines to display a graphics scene rendered using OpenGL. The GLUT routines also take relatively few parameters. [4] 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.

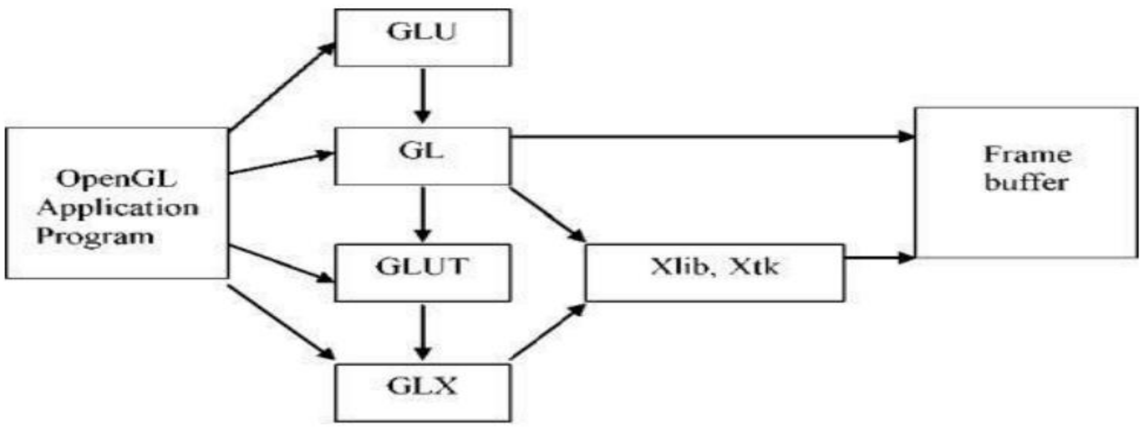


Figure 1.2 OpenGL Library Organization

1.3 Overview of project

The TRex game 3D, also referred to as the Chrome No Internet Game, is a browser-based project developed by Google. It aims to entertain users during instances of internet downtime by offering a simple yet engaging gaming experience. The game features a 3D-rendered T-Rex dinosaur as the protagonist, and players must navigate it through a challenging landscape while avoiding obstacles. The objective of the TRex game 3D is to survive as long as possible and achieve a high score. Players control the dinosaur's jumps to evade obstacles like cacti and pterodactyls. The game's minimalist graphics and intuitive controls, usually involving pressing the spacebar or tapping the screen, contribute to its widespread appeal across different age groups. The project's primary goal is to provide users with a source of entertainment when they encounter internet connectivity issues. By transforming frustration into amusement, the TRex game 3D has gained immense popularity. It has become a classic browser game, captivating players worldwide and challenging them to beat their own scores. With its simplicity and addictive gameplay, the TRex game 3D continues to be a go-to option for users seeking diversion during internet downtime.

1.4 Application

- Entertainment during internet downtime: The TRex Game 3D provides a fun and engaging way for users to pass the time when they encounter internet connection issues. It offers a source of entertainment during moments of frustration, transforming downtime into an enjoyable experience.
- Reflex and timing training: The game's fast-paced nature and obstacle avoidance mechanics help players develop and improve their reflexes and timing skills. By requiring precise jumps to navigate through obstacles, the game challenges players to think quickly and react appropriately.
- Casual gaming: The TRex Game 3D's simplicity and intuitive controls make it a popular choice for casual gamers. Its easy-to-understand mechanics and accessible gameplay appeal to a wide range of players, regardless of their gaming experience.

CHAPTER 2

REQUIREMENT ANALYSIS

2.1 Functional Requirements

Functional requirement includes all the functions and packages which are needed for the execution of this project. The project requires access to the OpenGL utility toolkit through the use of the header file "GL/glut.h". This header file, in addition to the usual header files is needed for the working of the project. For running the program, any basic C running compatible version of Code Blocks or Linux (Ubuntu) based platform is sufficient.

2.2 Non-Functional Requirements

The non-functional requirements include the software should use less memory as possible, which can be possible by using dynamic memory allocation. Other requirements include that the software should not accept wrong inputs and produce outputs. The project should be able to produce desired and quality outputs.

2.3 Hardware Requirements

The minimum/recommended hardware configuration required for developing the proposed software is given below:

- INTEL dual core and above compatible systems.
- Monitor, Keyboard, Mouse.
- RAM-512MB (minimum).

2.4 Software Requirements

- Ubuntu.
- OpenGL
- Code::Blocks

CHAPTER 3

SYSTEM DESIGN

3.1 Flowchart

Flowchart is a common type of chart that represents an algorithm or process showing the steps as boxes of various kinds, their order by connecting these with arrows.

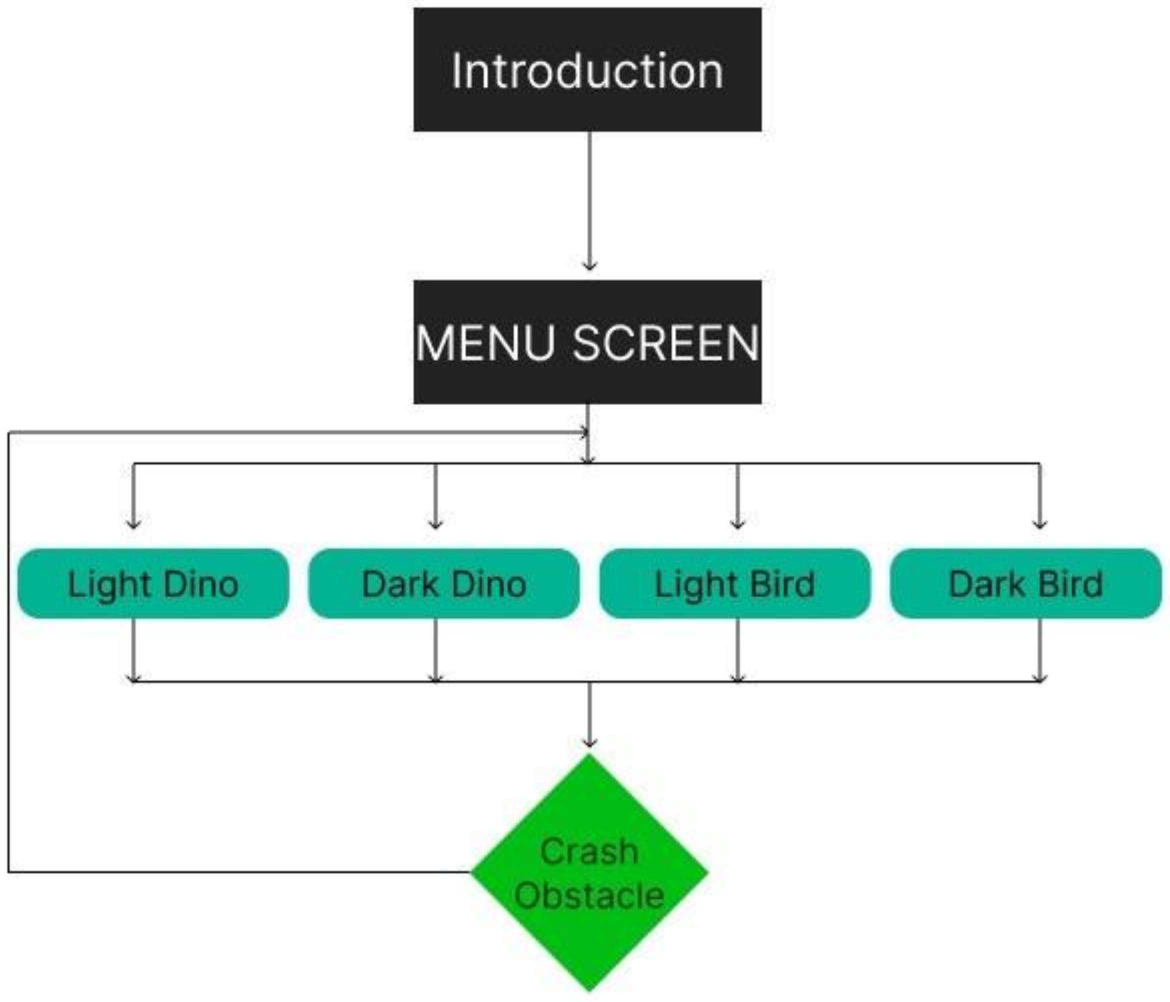


Figure 3.1 Flowchart

The provided flowchart depicts the flow of events in the TRex Game 3D :

Introduction:

The flowchart starts with the introduction stage, which sets the context and provides an overview of the game. This stage may include introductory graphics, game title, and basic instructions on how to play.

Menu Screen:

After the introduction, players are presented with a menu screen. The menu screen offers various options and choices for the player to select. It serves as a hub for accessing different game modes, customizations, and settings.

Character Selection:

Within the menu screen, the flowchart branches out into different character options: light dino, dark dino, light bird, and dark bird. This stage allows players to choose their preferred character, each with its unique appearance or attributes. Character selection adds variety and personalization to the gameplay experience.

Crash Obstacles:

Once the player has selected their character, the flowchart progresses to the main gameplay stage where the character starts running through the landscape. The objective is to navigate the character through the terrain while avoiding obstacles. The player must use appropriate controls, such as jumping, to overcome obstacles like cacti, flying birds, or other hazards. The flowchart implies that collision with these obstacles would lead to a game-ending crash.

The flowchart showcases the sequential progression of events in the TRex Game 3D. It highlights the importance of menu navigation, character selection, and the core gameplay element of avoiding obstacles. By providing clear stages and choices, the flowchart helps ensure a structured and engaging gameplay experience for the player.Overall, this flowchart outlines the key stages of the game, from introduction to character selection, and ultimately, the challenging obstacle-avoidance gameplay. It serves as a visual representation of the flow of events, guiding the player through different stages of the TRex Game 3D.

3.2 Interaction with program

This program includes interaction through keyboard.

- UP ARROW -→ jump.
- SPACEBAR --→ RESET

CHAPTER 4

IMPLEMENTATION

4.1 INBUILT FUNCTIONS

4.1.1 Interaction With Windows

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

glutInit is used to initialize the GLUT library. The argument from main are passed in and can be used by the application argc. A pointer to the unmodified argc, variable from main function argv a pointer to the unmodified argv, variable from the main function

int glutCreateWindow (char *tittle)

Create a window on display. The string tittle can be used to label window. The return value provides a reference to the window that can be used when there are multiple windows. The return value is the window handler for that particular window tittle sets the window tittle.

void glutInitDisplayMode (unsigned int mode)

Request a display with the properties in mode. The value of the mode is determined byte logical OR of options including the color model (GLUT_RGB,GLUT_INDEX) and buffering (GLUT_SINGLE,GLUT_DOUBLE) mode specifies the display mode GLUT_SINGLE single buffer window GLUT_DOUBLE double buffer window required to have smooth animation.

void glutInitWindowPosition (int x, int y)

glutInitWindowPosition set the initial window position and size respectively. x denotes Window X location in pixels. y denotes Window Y location in pixels.

int glutCreateWindow (char *name)

glutCreateWindow creates a top-level window. The `name` will be provided to the window system as the window's name. The intent is that the window system will label the window with the name.

void glutMainLoop ()

void glutMainLoop () cause the program to enter an event -processing loop. It should be the last statement in the loop.

glColor3f (TYPE r, TYPE g, TYPE b)

glColor3f () sets the present RGB colors. The maximum and minimum values of the floating-points types are 1.0 and 0.0 respectively.

void glClear (GL bitfield mask)

glClear () clears the specified buffers to their current clearing values. The mask argument is bitwise logical OR combination of the values i,e GL_COLOR_BUFFER-BIT,GL_DEPTH_BUFFER_BIT.

void glclearcolor (GLfloat R, GLfloat B, GLfloat alpha)

glclearcolor () will clears the background color to current clearing values.

void Ortho2D(GLfloat left, GLfloat right, GLfloat top, GLfloat bottom)

Ortho2D () sets the ortho as specified values in the function argument.

void glFlush ()

glflush () force all the previously issued OpenGL commands to begin execution thus guarantying that they complete in finite time.

void glVertex2f (GL float x, GLfloat y)

glVertex2f specifies a vertex in 2-dimensional plane.

void glBegin (GL_POLYGON)

glBegin () draws a polygonal diagram. Takes the end points from the glVertex function.

void glBegin(GL_LINES)

glBegin draws a line. Takes the end points from glVertex function.

void glRasterPos2f(TYPE*coordinates)

glRasterPos2f specifies the raster position.

void glutBitMapCharacter (void * font int char)

It renders the character with ASCII code at the current raster position using the raster font given in the font.

void glEnd ()

Indicates the end of glBegin ().

- **display()** All the entities mentioned below are displayed using display function .

- **mouse(int button, int state, int x, int y) -** This function is used to process the mouse button being pressed or released and the position of the cursor when the button was interacted with. In the project, it is used to provide a menu-like feature based on cursor position.

```
#include <GL/freeglut.h> // basic GLUT library
#include <stdlib.h>      // for dynamic memory allocation
#include <stdbool.h>     // for boolean variables
#include <time.h>        // for seed in random generation
#include <string.h>      // for text operation
#include <stdio.h> // generic terminal printing (for debugging)
#define WW 800 // Window        Width
#define WH 600 // Window        Height
#define DW 150 // Dinosaur      Width
#define DH 100 // Dinosaur      Height
#define CW 60  // Cactus  Width
#define CH 80  // Cactus  Height
#define LW 252 // Cloud  Width
#define LH 140 // Cloud  Height
// Struct for Image Objects
```

```
typedef struct figureObjects
{
    unsigned char **r, **g, **b;
} figure;

// Function Declarations
/* -- Initialization Functions -- */
void init(void);
void convColors(void);
void loadImages(void);
figure loadArray(FILE *, int, int);
/* -- Event Trigger Functions -- */
void keyPress(unsigned char, int, int);
void reset(void);
/* -- Action Functions -- */
void loop(int); // Callback Loop (timed)
void checkCollision(void);
void eventCollision(void);
void updateDino(void);
void updateCacti(void);
void placeCacti(void);
void disp(void); // Display Function
/* -- Drawing Functions -- */
void drawScene(void);
void drawFigure(int, int, int, int, int, int);
// Drawing Utilities
void drawLine(int, int, int, int);
void drawRect(int, int, int, int);
void drawText(void);
char *intToStr(int);
// Global Variables
/* -- Environment Variables -- */
int refreshPeriod = 5, runtime = 0, score = 0, hiscore;
```

```
bool halt = true, started = false;
/* -- Colour Definitions (RGB) -- */
GLfloat cols[7][3] = {
   {0, 0, 0},       // Black
   {230, 230, 230}, // Grey Light
   {85, 85, 85},    // Grey Dark
   {25, 50, 128},   // Blue
   {25, 75, 25},    // Green
   {76, 38, 10},    // Brown
   {255, 255, 255}  // White
};
/* -- Image Objects -- */
figure dino1, dino2, dino3, dino4, dino5, dino6; // Dinosaur
figure cacti, cloud;                             // Scenery
/* -- Control Variables for Dinosaur -- */
// Figure
int dinoState = 0, dinoX = 100, dinoY = WH / 4, dinoHS = 0;
int dinoHeights[20] = {
   0, 6, 12, 17, 21, 25, 28, 30, 32, 33,
   34, 33, 32, 30, 28, 25, 21, 17, 12, 6};
int dinoLeftEdge, dinoRightEdge;
// Motion
bool dinoJumpEnable = false;
int dinoPeriod = 20;
/* -- Control Variables for Cactus -- */
// Figure
int cactusOffset = WH / 4;
int cactiPos[5] = {WW * 2, WW * 2, WW * 2, WW * 2, WW * 2}, cactiLastPushed = -
999;
int winLeftEdge, winRightEdge;
// Motion
int cactiPeriod = 5, cactiShift = 5;
int gapPeriodOrig = 800, gapDelta = 0, gapPeriod = 800;
```

```
/* -- Control Variables for Cloud -- */
// Figure
int cloudOffset = 13 * WH / 20, cloudOffsetDelta = 0;
int cloudPos = WW / 3, cloudLastPushed = -999;
// Motion
int cloudPeriod = 20, cloudShift = 2;
// Function Definitions
/* -- Main Function -- */
int main(int argc, char *argv[])
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(WW, WH);
    glutInitWindowPosition(200, 200);
    glutCreateWindow("Dino 2D");
    init();
    glutKeyboardFunc(keyPress);
    glutDisplayFunc(disp);
    glutTimerFunc(refreshPeriod, loop, 0);
    glutMainLoop();
    return 0;
}
/* -- Initialization Functions  -- */
void init(void)
{
    gluOrtho2D(0.0, WW, 0.0, WH);
    dinoLeftEdge = 0 - (CW / 2) + dinoX;
    dinoRightEdge = DW + (CW / 2) + dinoX;
    winLeftEdge = 0 - (CW / 2);
    winRightEdge = WW + (CW / 2);
    gapPeriod = cactiPeriod * 100;
    convColors();
    loadImages();
    FILE *fp = fopen("res/hiscore", "r");
    fscanf(fp, "%d", &hiscore);
}
void convColors(void)
{
    for (int i = 0; i < (sizeof(cols) / (sizeof(GLfloat) * 3)); i++)
        for (int j = 0; j < 3; j++)
            cols[i][j] /= 255;
}
void loadImages(void)
{
    char **fn = (char **)malloc(sizeof(char *) * 8);
    fn[0] = "res/dino1.pbm";
    fn[1] = "res/dino2.pbm";
    fn[2] = "res/dino3.pbm";
    fn[3] = "res/dino4.pbm";
    fn[4] = "res/dino5.pbm";
    fn[5] = "res/dino6.pbm";
    fn[6] = "res/cacti.pbm";
    fn[7] = "res/cloud.pbm";
    for (int i = 0; i < 8; i++)
    {
        FILE *fp = fopen(fn[i], "r");
        if (fp == NULL)
        {
            printf(" -- file \"%s\" missing\n", fn[i]);
            break;
            exit(0);
        }
        else
        {
            switch (i)
        {
            case 0:
                dino1 = loadArray(fp, DW, DH);
                break;
            case 1:
                dino2 = loadArray(fp, DW, DH);
                break;
            case 2:
                dino3 = loadArray(fp, DW, DH);
                break;
            case 3:
                dino4 = loadArray(fp, DW, DH);
                break;
            case 4:
                dino5 = loadArray(fp, DW, DH);
                break;
            case 5:
                dino6 = loadArray(fp, DW, DH);
                break;
            case 6:
                cacti = loadArray(fp, CW, CH);
                break;
            case 7:
                cloud = loadArray(fp, LW, LH);
                break;
            }
        }
    }
}

figure loadArray(FILE *fp, int w, int h)
{
    unsigned char **r, **g, **b;
    r = malloc(sizeof(unsigned char *) * h);
    g = malloc(sizeof(unsigned char *) * h);
    b = malloc(sizeof(unsigned char *) * h);
    for (int i = 0; i < h; i++)
    {
        r[i] = malloc(sizeof(unsigned char) * w);
        g[i] = malloc(sizeof(unsigned char) * w);
        b[i] = malloc(sizeof(unsigned char) * w);
    }

    for (int i = 0; i < h; i++)
    {
        for (int j = 0; j < w; j++)
        {
            r[h - i - 1][w - j - 1] = fgetc(fp);
            g[h - i - 1][w - j - 1] = fgetc(fp);
            b[h - i - 1][w - j - 1] = fgetc(fp);
        }
    }

    figure img;
    img.r = r;
    img.g = g;
    img.b = b;
    return img;
}
/* -- Event Trigger Functions -- */
void keyPress(unsigned char key, int x, int y)
{
    switch (key)
    {
    case 27: // Key Esc
        exit(0);
```

```
            break;
        case 32: // Key Space
            if (halt == true)
                reset();
            else
                dinoJumpEnable = true;
            break;
        case 112: // Key 'p'
            halt = !halt;
            break;
        }
    }
    void reset(void)
    {
        runtime = 0;
        halt = false;
        started = true;
        score = 0;
        dinoState = 0;
        dinoHS = 0;
        cactiPos[0] = cactiPos[1] = cactiPos[2] = cactiPos[3] = cactiPos[4] = WW * 2;
        cactiLastPushed = -999;
    }
    /* -- Action Functions -- */
    // Callback Loop (timed)
    void loop(int val)
    {
        runtime += refreshPeriod;

        if (halt == true)
            goto skip;
        if (runtime % dinoPeriod == 0)
            updateDino();
        if (runtime % cactiPeriod == 0)
            updateCacti();
        checkCollision();
        if (runtime % (10 * cactiPeriod) == 0)
            score++;
        glutPostRedisplay();
    skip:
        glutTimerFunc(refreshPeriod, loop, 0);
    }
    void checkCollision(void)
    {
        for (int i = 0; i < 5; i++)
            if (cactiPos[i] > dinoLeftEdge && cactiPos[i] < dinoRightEdge)
            {
                if (cactiPos[i] > dinoX + 2 * DW / 5 && cactiPos[i] < dinoX + 3 * DW / 5)
                {
                    if (dinoHeights[dinoHS] * 4 < CH)
                    {
                        eventCollision();
                        break;
                    }
                }
                else
                {
                    if (dinoHeights[dinoHS] * 4 < CH - (13 * DH / 20))
                    {
                        eventCollision();
                        break;
                    }
                }
            }
    }
    void eventCollision(void)
```

```
{
    halt = true;
    if (score <= hiscore)
        return;
    FILE *fp = fopen("res/hiscore", "w");
    fprintf(fp, "%d", score);
    fclose(fp);
}
void updateDino(void)
{
    dinoState = ((dinoState + 1) % 6);
    if (dinoJumpEnable == true)
    {
        dinoHS = ((dinoHS + 1) % 19);
        if (dinoHS == 0)
            dinoJumpEnable = false;
    }
}
void updateCacti(void)
{
    if (cactiPos[4] == winRightEdge)
        cactiLastPushed = runtime;

    for (int i = 0; i < 5; i++)
        cactiPos[i] -= cactiShift;
    if (runtime == (cactiLastPushed + gapPeriod))
    {
        for (int i = 0; i < 4; i++)
            cactiPos[i] = cactiPos[i + 1];
        cactiPos[4] = winRightEdge;
        srand(time(0));
        gapDelta = rand() % 70;
        gapPeriod = gapPeriodOrig + gapDelta * 10;
    }
}
// Display Function
void disp(void)
{
    drawScene();
    // Draw Dinosaur
    drawFigure(dinoState, dinoX, dinoY, dinoHeights[dinoHS] * 4, DW, DH);
    // Draw Cacti
    placeCacti();
    // Draw Vvisible Text Content
    drawText();
    glutSwapBuffers();
}
/* -- Drawing Functions -- */
void drawScene(void)
{
    /* -- Background -- */
    glColor3fv(cols[1]);
    drawRect(0, 0, WW, WH);
    /* -- Clouds -- */
    if (runtime % cloudPeriod == 0)
    {
        if (cloudPos >= -(LW / 2) && cloudPos <= WW + (LW / 2))
            cloudPos -= cloudShift;
        else
        {
            cloudPos = WW + (LW / 2);
            srand(time(0));
            cloudOffsetDelta = (rand() % (WH / 40)) * 10;
        }
    }
    drawFigure(7, cloudPos - (LW / 2), cloudOffset + cloudOffsetDelta, 0, LW, LH);
    /* -- Ground -- */
    glColor3fv(cols[5]);
    drawRect(0, 0, WW, WH / 4);
}
void drawFigure(int f, int xpos, int ypos, int yoff, int w, int h)
{
    figure temp;
    if (yoff > 0)
    {
        temp = dino4;
        goto skip;
    }
    switch (f)
    {
    case 0:
        temp = dino1;
        break;
    case 1:
        temp = dino2;
        break;
    case 2:
        temp = dino3;
        break;
    case 3:
        temp = dino4;
        break;
    case 4:
        temp = dino5;
        break;
    case 5:
        temp = dino6;
        break;
    case 6:
        temp = cacti;
        break;
    case 7:
        temp = cloud;
        break;
    }
skip:
    glBegin(GL_POINTS);
    for (int i = 0; i < h; i++)
    {
        for (int j = 0; j < w; j++)
        {
            glColor3ub(temp.r[i][j], temp.g[i][j], temp.b[i][j]);
            glVertex2f(xpos + j, ypos + yoff + i);
        }
    }
    glEnd();
}
void placeCacti(void)
{
    for (int i = 0; i < 5; i++)
    {
        if (cactiPos[i] > winLeftEdge && cactiPos[i] < winRightEdge)
            drawFigure(6, cactiPos[i] - (CW / 2), cactusOffset, 0, CW, CH);
    }
}
/* -- Drawing Utilities -- */
void drawLine(int xa, int ya, int xb, int yb)
{
    glBegin(GL_LINES);
    glVertex2i(xa, ya);
    glVertex2i(xb, yb);
    glEnd();
```

```
}
void drawRect(int xa, int ya, int xb, int yb)
{
    glBegin(GL_POLYGON);
    glVertex2i(xa, ya);
    glVertex2i(xa, yb);
    glVertex2i(xb, yb);
    glVertex2i(xb, ya);
    glEnd();
}
void drawText(void)
{
    glColor3fv(cols[3]);
    glRasterPos2f(3 * WW / 4, 7 * WH / 8);
    unsigned char points[] = "Score : ";
    strcat(points, intToStr(score));
    glutBitmapString(GLUT_BITMAP_HELVETICA_18, points);
    if (score > hiscore)
        hiscore = score;
    glRasterPos2f(3 * WW / 4, 13 * WH / 16);
    unsigned char hipoint[] = "Hi-Score : ";
    strcat(hipoint, intToStr(hiscore));
    glutBitmapString(GLUT_BITMAP_HELVETICA_18, hipoint);
    glColor3fv(cols[1]);
    glRasterPos2f(11 * WW / 28, WH / 10);
    unsigned char pause[] = "Press P for Play/Pause";
    glutBitmapString(GLUT_BITMAP_HELVETICA_18, pause);
    if (halt == false)
        return;
    glColor3fv(cols[4]);
    glRasterPos2f(23 * WW / 56, WH / 2);
    unsigned char success[] = "Press Spacebar";
    glutBitmapString(GLUT_BITMAP_HELVETICA_18, success);
```

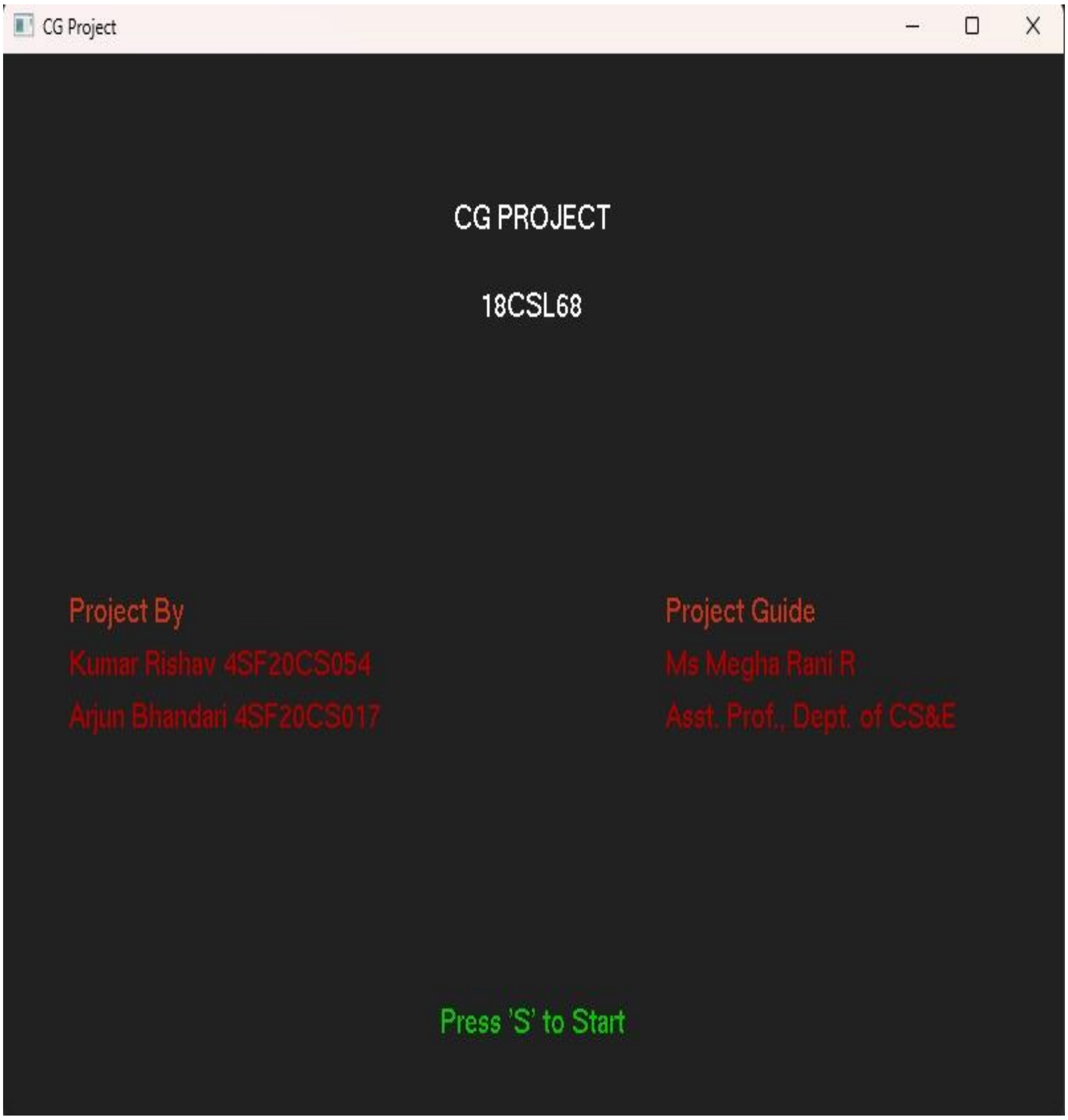
```
}
char *intToStr(int n)
{
    if (n == 0)
        return "0\0";
    char s[8];
    int k = 0;
    while (n)
    {
        s[k++] = (n % 10) + '0';
        n /= 10;
    }
    char *tem = (char *)malloc(sizeof(char) * k + 1);
    for (int i = 0; i < k; i++)
        tem[i] = s[k - i - 1];
    tem[k] = '\0';
    return tem;
}
```

CHAPTER 5

RESULT

5.1.1 Introduction page:

It is the first phase of the project which shows introduction. Here it contains project details and teammates details and press s to start.



CG Project

CG PROJECT

18CSL68

Project By

Kumar Rishav 4SF20CS054

Arjun Bhandari 4SF20CS017

Project Guide

Ms Megha Rani R

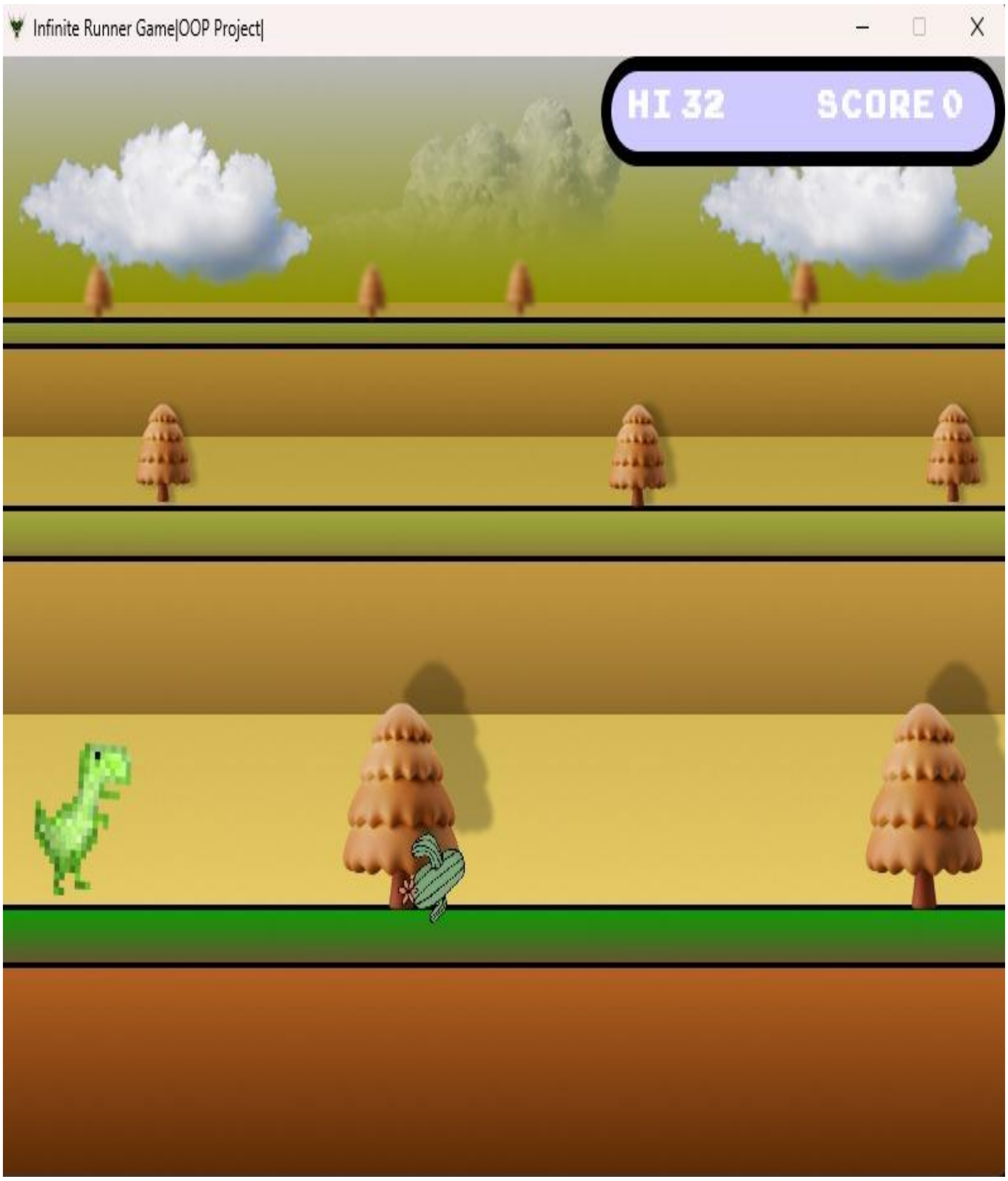
Asst. Prof., Dept. of CS&E

Press 'S' to Start

Fig.5.1.1. Introduction page.

5.1.2 Light Dino:

In the Phase of the project, it is shown where dino is running on the road and obstacles like cactus is coming if it touch it then game over for dino. Here it contains dino , cactus and road.



`

Fig.5.1.2 Light Dino Scenery

5.1.3 Dark Dino:

In the Phase of the project, it is shown where dino is running on the road and obstacles like fire ball is coming if it touch it then game over for dino. Here it contains dino , fire ball and road and theme is also dark.

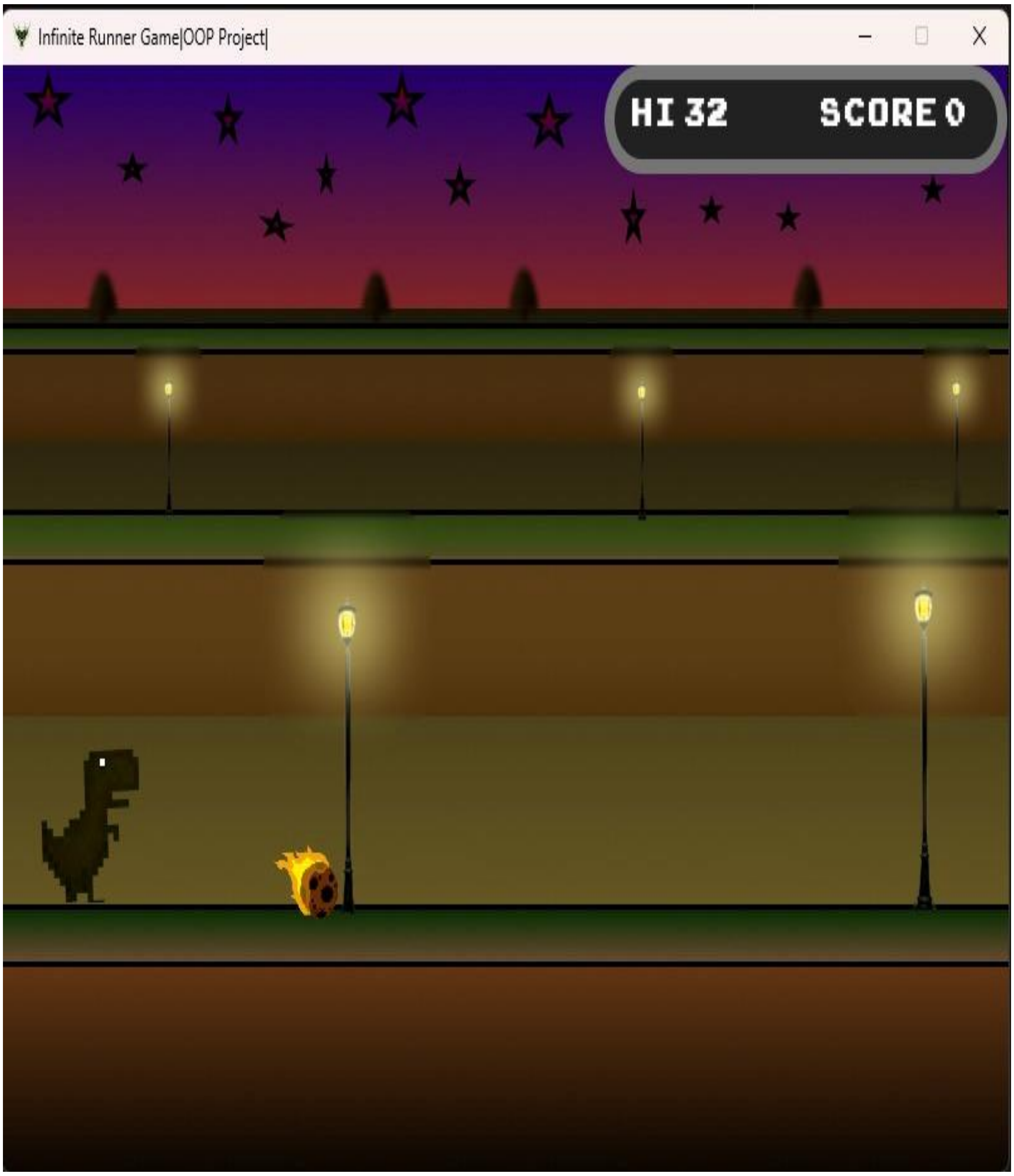


Fig.5.1.3 Dark Dino Scenery.

5.1.4 Light Bird:

In the Phase of the project, it is shown where bird is running on the road and obstacles like cactus is coming if it touch it then game over for dino. Here it contains bird, cactus and road.

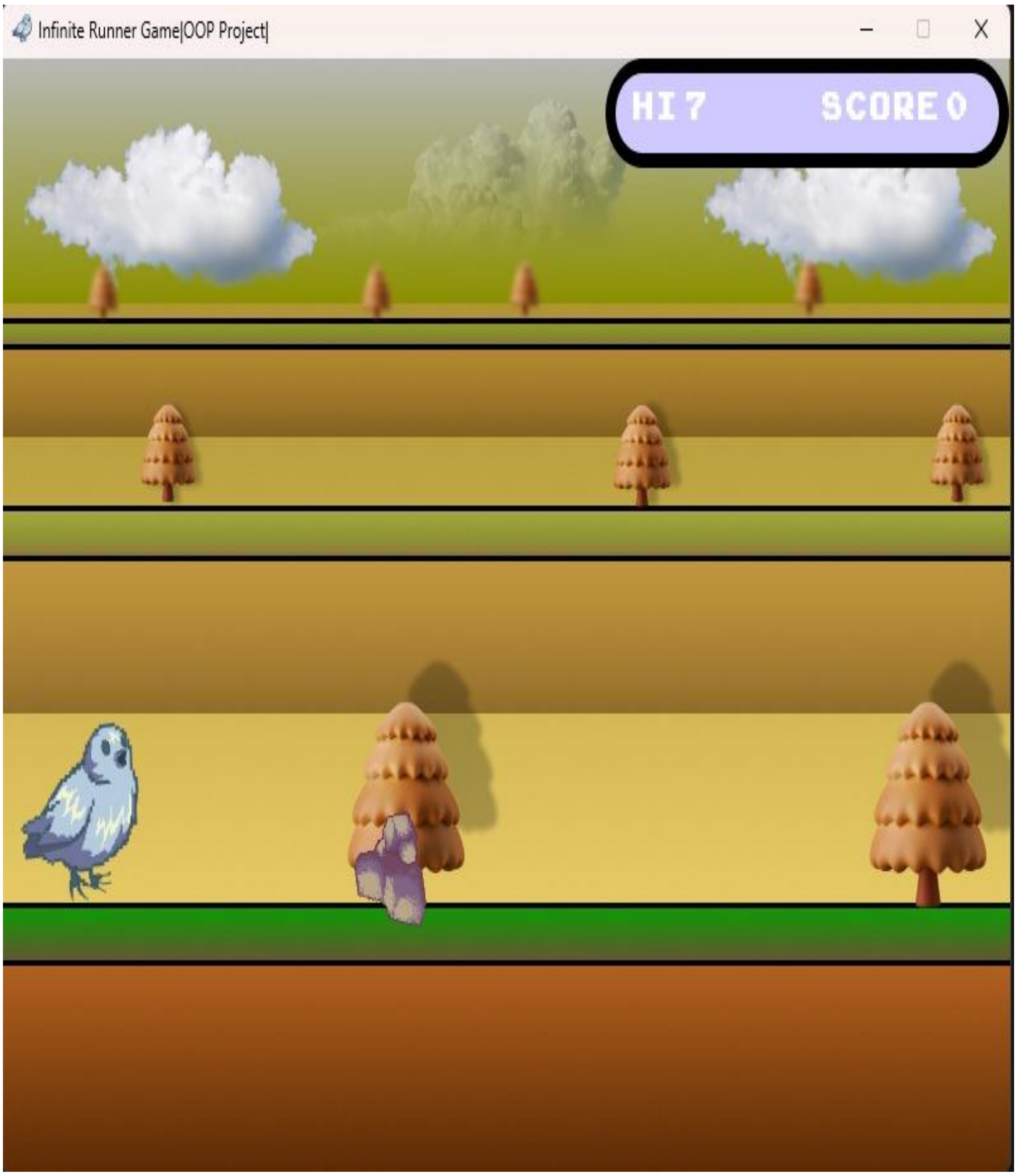


Fig.5.1.4 Light Bird Scenery.

5.1.5 Dark Bird:

In the Phase of the project, it is shown where bird is running on the road and obstacles like fire ball is coming if it touch it then game over for bird. Here it contains bird , fire ball and road and theme is also dark.

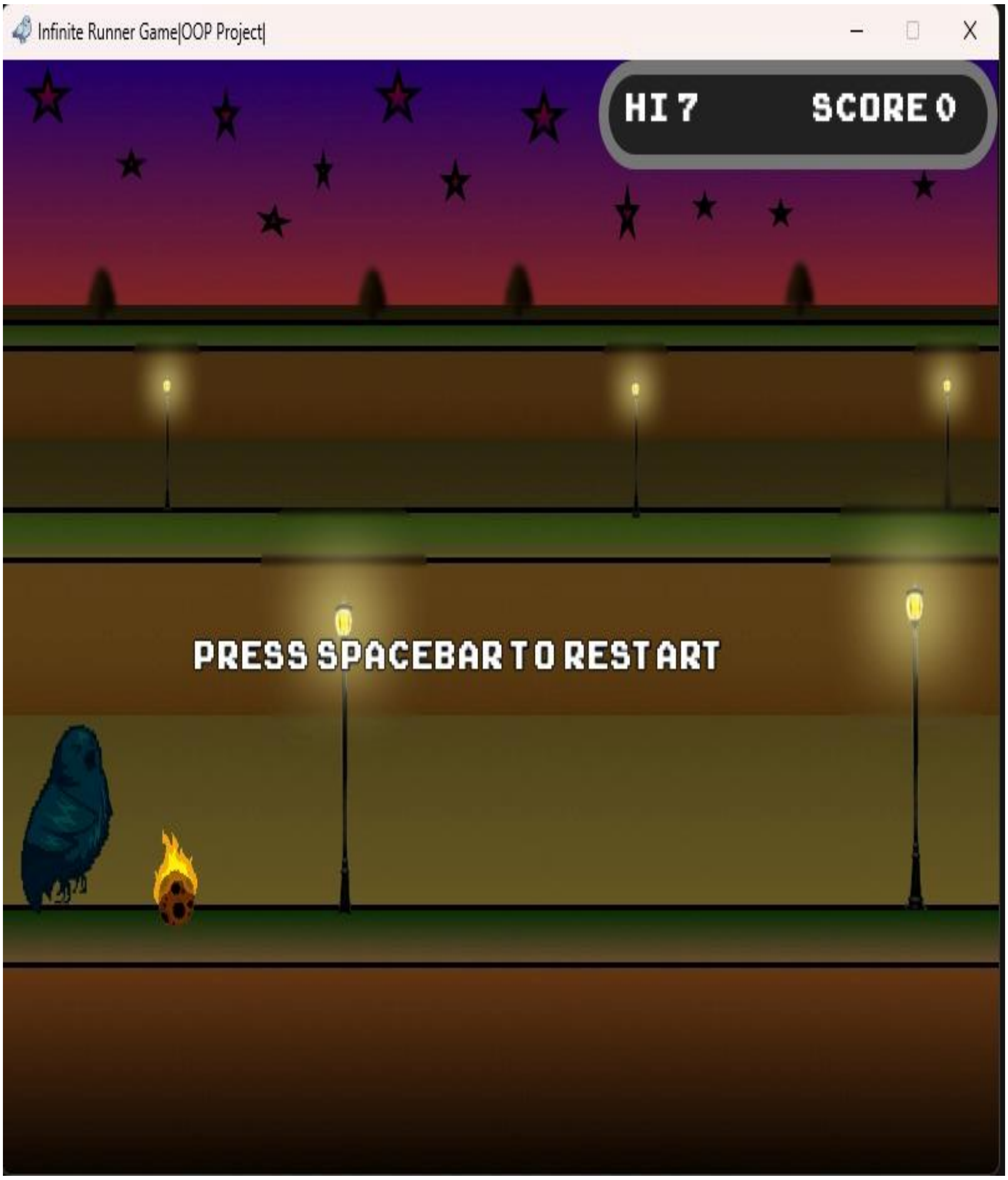


Fig.5.1.5 Dark Bird Scenery.

CONCLUSION

The TRex Game 3D, also known as the Chrome No Internet Game, has become a widely beloved browser-based project developed by Google. It has successfully transformed moments of frustration caused by internet downtime into opportunities for entertainment and diversion. With its simple yet addictive gameplay, the game has captured the attention of millions of users worldwide. The project's objective of providing entertainment during internet downtime has been achieved through the creation of a charming 3D-rendered T-Rex dinosaur and a challenging landscape filled with obstacles. The game's minimalist graphics and intuitive controls make it accessible to players of all ages and gaming backgrounds. Its fast-paced nature tests players' reflexes and timing, creating an engaging experience that keeps them coming back for more. The TRex Game 3D has found applications in various areas. It serves as a means of entertainment, offering a fun and engaging experience to users during internet connectivity issues. Additionally, the game provides an opportunity for reflex and timing training, helping players improve their cognitive skills in a captivating way. It also serves as a stress reliever, providing a temporary escape from daily pressures and allowing players to immerse themselves in a challenging and enjoyable gameplay experience. Furthermore, the game fosters competitiveness among players, encouraging them to achieve higher scores and engage in friendly competition. It has become a classic browser game, with players worldwide challenging themselves and others to beat their previous records. In conclusion, the TRex Game 3D has successfully achieved its goal of providing entertainment and amusement during internet downtime. Its simple yet captivating gameplay, coupled with its wide range of applications, has contributed to its widespread popularity. By transforming frustration into fun, the TRex Game 3D has established itself as a go-to source of entertainment for users around the world, leaving a lasting impact on the gaming community.

REFERENCES

[1] OpenGL: htps://www.opengl.org

[2] Mark J Kilgard: The OpenGL utility toolkit (GLUT) Programming API version 3

[3] Edward Angle - Interactive Computer graphics A Top-Down Approach with OpenGL, 5th Edition, Addition – Wesley, 2008.

[4] James D Foley, Andrei's Van Dam, Steven K Feigner, John F Hugs - Computer Graphics, Pearson Education