COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

UCS505 COMPUTER GRAPHICS

PROJECT REPORT ON:

Classic Snake Game

SUBMITTED TO: Jyoti Rani Ma'am

SUBMITTED BY: -

101917140 Saina Agrawal
102097025 Suvigya Pandey
101917151 Bhuvnesh Arora

Batch: CSE6

19th May 2022

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY, PATIALA, PUNJAB

Table of Contents

Introduction of Project

“Every great story seems to begin with a snake “ - Nicolas Cage

Our Mr. Protagonist is a hungry dude that roams around in order to survive. He has a lot to eat but the right fruits prove to be fruitful. Save him from eating bombs. Save him from reaching the boundary. But most importantly save him from eating himself. Save him and the guy lives.

In this project, the player controls snake movements by changing its direction. Snake automatically moves forward in the current direction. To collect a piece of fruit, the player has to move through it by directing the snake's head to the piece. The player also has to avoid bombs in the path, they will reduce the size of the snake and the game score as well.

Challenge is to eat more and more fruits without getting out of the boundary, with the primary goal to survive as much as possible. How long can you last before your tail becomes your dinner?

Working / Explanation

As the window appears the player will be given with a set of rules/instructions as discussed in the Introduction part. After going through these the player jumps over to start the game.

On starting the game, the Snake in its standard set-size starts to move in a random direction. There starts to appear a collection of fruits and bombs randomly on the screen and the player tries to make the snake eat yellow squares (Fruits) and repel red circles (Bombs) by moving the snake in all 4 directions.

- up arrow key (For moving up)
- left arrow key (For moving left)
- down arrow key (For moving down)
- left arrow key (For moving right)

After eating every fruit, the length of the snake gets increased by one segment and then it goes to eat other fruits. The game ends when:

- Snake reaches boundary

Now anytime the snake eats itself at a certain segment, automatically the length of the snake gets reduced by that many segments. A binary file with the name Snake.bin is also maintained saving current and highest scores of each game play.

Computer Graphics concepts used

- OpenGL is a cross language, cross platform application programming interface forrendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit to achieve hardware-accelerated rendering.

- For drawing various objects in the game, we have used the GL_POLYGON argument of the glBegin and glend functions which delimit the vertices of a primitive or a group of like primitives.

  To draw a primitive on the screen first we call glBegin, specifying what kind of primitive it is that you want to draw in the mode parameter of glBegin, and then list all vertices one by one (by sequentially calling glVertex3f) and finally call glEnd to let OpenGL know that we're done drawing a primitive. The function glVertex2f is also used at some places, it specifies the x and y coordinates of the vertex, and the z coordinate is set to zero. The function glVertex3f that specifies all three coordinates. The "2" or "3" in the name tells how many parameters are passed to the function

- To make the snake move, we used the glTranslatef function which multiplies the current matrix by a translation matrix. It will move the snake by the given x-, y-, z-values. The coordinates will be adjusted according to the direction of the snake given by the arrow keys of the keyboard.

- For setting the background colour to white, we used the glClearColor function. glClearColor specifies the red, green, blue, and alpha values used by glClear to clear the color buffers.

- To fill various polygons like snake, fruits, bombs, text boxes etc, we used the glColor3f function which sets the current color of the shape. We need to specify new red, green, and blue values for the current color.

- To set the mouse callback for the current window, we used the glutMouseFunc function. When a user presses and releases mouse buttons in the window, each press and each release generates a mouse callback. The button parameter is one of GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, or GLUT_RIGHT_BUTTON.

User defined functions

1) **DrawFruct():** Draws a yellow coloured fruit in a shape of square.

2) **DrawBomb():** Draws a red coloured bomb in a shape of a circle.

3) **draw_string():** It is used to render the inputted string.

4) **DrawMenu():** Draws the initial menu with “Start” , “Rules” ,”Exit” options.

5) **DrawExit():** Draws the exit menu with a message displaying “Game Over!” and also showing the Final and highest score with option to return to to menu or exit finally.

6) **New():** It is used to calculate x and y coordinates using rand() function.

7) **DrawRules():** Displays the rules of the game with an option to return to the menu at the bottom.

8) **Tick():** Controls movement of the snake’s body, increase and decrease of snake’s length on eating a fruit and a bomb, respectively, and ending of game on touching the boundary or when snake eats its own tail.

9) **DrawSnake():** Displays the snake’s body.

10) **DrawScore():** Displays the current score of the player and the high score.

11) **display():** Displays the game on clicking on Start, displays the rules on clicking on Rules and exits the game on clicking on Exit.

12) **fjfjfh():** Function for declaration and initialization of variables used.

13) **MyKeyboard(int key, int a, int b):** Snake moves up, down, left and right when respective keys are pressed.

14) **MousePressed(int button, int state, int ax, int ay):** Provides the arguments for the function glutMouseFunc(), which sets the mouse callback for the current window.

15) **CreateGlutWindow():** Sets the initial display mode, initial window position and size, colour, current matrix mode, replaces current matrix with identity matrix and sets up a 2D orthographic projection matrix.

16) **timer (int = 0):** Sets timer callback to be triggered in 80 milliseconds.

Source code

```
#include <GL/glut.h>
#include <fstream>

#include "math.h"

using namespace std;


int dir;
int d = 1; int num =
7; int key1 = 3; int
Score = 0; int
hightScore; int
Scale = 25; int N =
50, M = 30; int w =
Scale * N; int h =
Scale * M;

char sScore[15]; char
sHightScore[15];
float xm = 0.0; float
ym = 0.0; bool down
= false;

struct {
int x; int
y; }
s[100];

class Fruct
{ public: int
x, y;
   void New()
   { x = rand() % N; y =
      rand() % (M - 3);
   } void
   DrawFruct()
   { glColor3f(1.0, 0.9, 0.0); glRectf(x * Scale, y * Scale, (x + 1)
      * Scale, (y + 1) * Scale);
```

```
   } }
m[2];

class Bomb
{ public: int
x, y;

  void New()
  { x = rand() % N; y =
     rand() % (M - 3);
  } void
  DrawBomb()
  { glColor3f(1.0, 0.0, 0.0);
     glBegin(GL_POLYGON); for (float i = 0;
     i < 2 * 3.14; i += 3.14 / 4) {
        glVertex2f((x + 0.5) * Scale + (0.5) * Scale * (1.1) * sin(i),
           (y + 0.5) * Scale + (0.5) * Scale * (1.1) * cos(i));
     }
     glEnd();
  }
} u[10];

void draw_string(void* font, const char* string)
{ while
  (*string)
     glutStrokeCharacter(font, *string++);
}

void DrawMenu()
{
  glClear(GL_COLOR_BUFFER_BIT);

  glBegin(GL_POLYGON); // Start
  glColor3f(0.0, 0.0, 0.0);
  glVertex3f(705.0, 650.0, 0.0);
  glColor3f(0.0, 0.0, 0.0);
  glVertex3f(500.0, 650.0, 0.0);
  glColor3f(0.0, 0.0, 0.0);
  glVertex3f(500.0, 550.0, 0.0);
  glColor3f(0.0, 0.0, 0.0);
  glVertex3f(705.0, 550.0, 0.0); glEnd();
```

```
    glBegin(GL_POLYGON); // Rules
    glColor3f(0.0, 0.0, 0.0);
    glVertex3f(705.0, 510.0, 0.0);
    glColor3f(0.0, 0.0, 0.0);
    glVertex3f(500.0, 510.0, 0.0);
    glColor3f(0.0, 0.0, 0.0);
    glVertex3f(500.0, 410.0, 0.0);
    glColor3f(0.0, 0.0, 0.0);
    glVertex3f(705.0, 410.0, 0.0); glEnd();
    glBegin(GL_POLYGON); // Exit
    glColor3f(0.0, 0.0, 0.0);
    glVertex3f(670.0, 230.0, 0.0);
    glColor3f(0.0, 0.0, 0.0);
    glVertex3f(540.0, 230.0, 0.0);
    glColor3f(0.0, 0.0, 0.0);
    glVertex3f(540.0, 140.0, 0.0);
    glColor3f(0.0, 0.0, 0.0);
    glVertex3f(670.0, 140.0, 0.0); glEnd();

    glLineWidth(4.0f);
    glColor3f(1.0, 1.0, 1.0);

    glPushMatrix(); glTranslatef(w /
    (2.38), h / (1.3), 0); glScalef(0.4f,
    0.4f, 0.4f);
    draw_string(GLUT_STROKE_ROMAN, "START");
    glPopMatrix();

    glPushMatrix(); glTranslatef(w /
    (2.38), h / (1.7), 0); glScalef(0.4f,
    0.4f, 0.4f);
    draw_string(GLUT_STROKE_ROMAN, "RULES");
    glPopMatrix();

    glPushMatrix(); glTranslatef(w /
    (2.229), h / (4.4), 0); glScalef(0.4f,
    0.4f, 0.4f);
    draw_string(GLUT_STROKE_ROMAN, "EXIT");
    glPopMatrix();
    glFinish();
    glutSwapBuffers();
}
```

```
void DrawExit()
{
  glClear(GL_COLOR_BUFFER_BIT);
  glBegin(GL_POLYGON);
  glColor3f(0.0, 0.0, 0.0);
  glVertex3f(750.0, 150.0, 0.0);
  glColor3f(0.0, 0.0, 0.0);
  glVertex3f(500.0, 150.0, 0.0);
  glColor3f(0.0, 0.0, 0.0);
  glVertex3f(500.0, 90.0, 0.0);
  glColor3f(0.0, 0.0, 0.0);
  glVertex3f(750.0, 90.0, 0.0);
  glEnd();
  glBegin(GL_POLYGON);
  glColor3f(0.0, 0.0, 0.0);
  glVertex3f(670.0, 80.0, 0.0);
  glColor3f(0.0, 0.0, 0.0);
  glVertex3f(580.0, 80.0, 0.0);
  glColor3f(0.0, 0.0, 0.0);
  glVertex3f(580.0, 30.0, 0.0);
  glColor3f(0.0, 0.0, 0.0);
  glVertex3f(670.0, 30.0, 0.0);
  glEnd();

  glLineWidth(7.0f);
  glColor3f(1.0, 0.0, 0.0);
  glPushMatrix(); glTranslatef(w /
  (6), h / (1.5), 0); glScalef(1.1f,
  1.1f, 1.1f);
  draw_string(GLUT_STROKE_ROMAN, "Game over!");
  glPopMatrix();

  glLineWidth(2.5f);
  glColor3f(1.0, 1.0, 1.0);
  glPushMatrix();
  glTranslatef(w / (2.4), h / 7,
  0); glScalef(0.2f, 0.2f, 0.2f);
  draw_string(GLUT_STROKE_ROMAN, "Return to MENU");
  glPopMatrix();
  glPushMatrix(); glTranslatef(w /
  (2.07), h / (15.3), 0); glScalef(0.2f,
  0.2f, 0.2f);
```

```
  draw_string(GLUT_STROKE_ROMAN, "EXIT");
  glPopMatrix();

  glLineWidth(3.5f); glColor3f(0.9, 0.3, 0.5);
  glPushMatrix(); glTranslatef(w / (2.8), h / (2.1), 0);
  glScalef(0.4f, 0.4f, 0.4f);
  draw_string(GLUT_STROKE_ROMAN, "Final score:");
  glPopMatrix();

  sprintf(sScore, "%9d", Score);
  glPushMatrix(); glTranslatef(w /
  (2.8), h / (2.1), 0); glScalef(0.4f,
  0.4f, 0.4f);
  draw_string(GLUT_STROKE_ROMAN, sScore);
  glPopMatrix();

  ifstream inFile("Snake.bin", ios_base::binary);
  while (inFile.peek() != EOF)
     inFile >> sHightScore;
  inFile.close(); hightScore =
  atoi(sHightScore); if (Score >
  hightScore) {
     sprintf(sHightScore, "%9d", Score); ofstream
  outFile("Snake.bin", ios_base::binary); outFile <<
  sScore; outFile.close(); } glPushMatrix();
  glTranslatef(w / (2.8), h / (2.55), 0); glScalef(0.4f,
  0.4f, 0.4f);
  draw_string(GLUT_STROKE_ROMAN, "High score:");
  glPopMatrix(); glPopMatrix(); glPushMatrix();
  glTranslatef(w / (1.6), h / (2.55), 0); glScalef(0.4f, 0.4f,
  0.4f);
  draw_string(GLUT_STROKE_ROMAN, sHightScore);
  glPopMatrix();
  glFinish();
  glutSwapBuffers();
} void
DrawRules()
{
  glClear(GL_COLOR_BUFFER_BIT);

  glBegin(GL_POLYGON);
```

```
glColor3f(0.0, 0.0, 0.0);
glVertex3f(710.0, 150.0, 0.0);
glColor3f(0.0, 0.0, 0.0);
glVertex3f(490.0, 150.0, 0.0);
glColor3f(0.0, 0.0, 0.0);
glVertex3f(490.0, 90.0, 0.0);
glColor3f(0.0, 0.0, 0.0);
glVertex3f(710.0, 90.0, 0.0);
glEnd();

glLineWidth(1.8f);
glColor3f(0.0, 0.0, 0.0);

glPushMatrix();
glTranslatef(w / 3, h / (1.2),
0); glScalef(0.5f, 0.5f, 0.5f);
draw_string(GLUT_STROKE_ROMAN, "How to play:");
glPopMatrix(); glPushMatrix(); glTranslatef(w / (3), h / (1.5), 0);
glScalef(0.2f, 0.2f, 0.2f); draw_string(GLUT_STROKE_ROMAN,
"1.Eat fruit to score points:"); glPopMatrix(); glPushMatrix();
glTranslatef(w / (3), h / (1.7), 0); glScalef(0.2f, 0.2f, 0.2f);
draw_string(GLUT_STROKE_ROMAN, "2.Do not eat bombs:");
glPopMatrix(); glPushMatrix(); glTranslatef(w / (3), h / (2.0), 0);
glScalef(0.2f, 0.2f, 0.2f); draw_string(GLUT_STROKE_ROMAN,
"3.Do not eat yourself."); glPopMatrix(); glPushMatrix();
glTranslatef(w / (3), h / (2.4), 0); glScalef(0.2f, 0.2f, 0.2f);
draw_string(GLUT_STROKE_ROMAN, "4.Do not leave the window
border."); glPopMatrix();

glPushMatrix(); glColor3f(1.0,
1.0, 1.0); glTranslatef(w /
(2.5), h / 7, 0); glScalef(0.19f,
0.2f, 0.2f);
draw_string(GLUT_STROKE_ROMAN, "Return to MENU");
glPopMatrix();

glLineWidth(3.0f); glColor3f(0.0,
0.0, 0.0); glPushMatrix();
glTranslatef(w / (2.7), h / (3.3),
0); glScalef(0.4f, 0.4f, 0.4f);
draw_string(GLUT_STROKE_ROMAN, "Good luck!");
glPopMatrix();
```

```
  glColor3f(1.0, 0.9, 0.0);
  glRectf(780.0, 500.0, 805.0, 525.0);
  glColor3f(1.0, 0.0, 0.0);
  glBegin(GL_POLYGON); for (float i = 0; i < 2 *
  3.14; i += 3.14 / 4) { glVertex2f(720.0 + (0.5) *
  Scale * (1.1) * sin(i),
          450.0 + (0.5) * Scale * (1.1) * cos(i));
  }
  glEnd();

  glFinish();
  glutSwapBuffers();
}

void Tick()
{
  // Movement of the snake's body:
  for (int i = num; i > 0; --i) {
     s[i].x = s[i - 1].x;
     s[i].y = s[i - 1].y;
  }

  // Movement of the snake's head:
  switch (dir) {
  case 0: s[0].y
  += 1; break;
  case 1: s[0].x
     -= 1;
     break;
  case 2: s[0].x
     += 1;
     break;
  case 3: s[0].y
  -= 1; break;
  } int h = 0;
  // The snake's length increases on eating a fruit:
  for (int i = 0; i < 5; i++)
      if ((s[0].x == m[i].x) && (s[0].y == m[i].y)) {
         num++;
         m[i].New();
         if (h != 11) {
```

```
            u[h].New();
        h++; } else {
            h = 0;
            u[h].New();
        }
        Score += 2;
    }

  // The snake's length decreases on eating a bomb:
  for (int i = 0; i < 10; i++)
     if ((s[0].x == u[i].x) && (s[0].y == u[i].y)) {
        if (num == 2)
            key1 = 2;
        if (num > 3)
            num = num - 2;
        else
            num = 2;
        u[i].New();
        if (Score >
        0)
            Score--;
        if (Score < 0)
            Score = 0;
    }

  // Game ends on touching the boundary:
  if (s[0].x > N || s[0].x < 0 || s[0].y > (M - 3) || s[0].y < 0) {
     key1 = 2;
  }

  // The snake's length decreases if it eats its tail:
  for (int i = 1; i < num; i++)
     if (s[0].x == s[i].x && s[0].y == s[i].y) {
        num = 3;
        if (Score > 0)
            Score -= 3;
        if (Score < 0)
        Score = 0;
      }
}
```

```
void DrawSnake()
{ glColor3f(0.0, 1.0, 0.0); for
  (int i = 0; i < num; i++) {
     glRectf(s[i].x * Scale, s[i].y * Scale, (s[i].x + 1) * Scale,
        (s[i].y + 1) * Scale);
   }
}

void DrawScore()
{ glLineWidth(1.5f);
  glColor3f(1.0, 1.0, 1.0);

  glPushMatrix(); glTranslatef(w /
  (5.4), h / (1.05), 0); glScalef(0.3f,
  0.3f, 0.3f);
  draw_string(GLUT_STROKE_ROMAN, "Your
  score:"); glPopMatrix(); sprintf(sScore, "%9d", Score);
  glPushMatrix(); glTranslatef(w / (5), h / (1.05), 0);
  glScalef(0.3f, 0.3f, 0.3f);
  draw_string(GLUT_STROKE_ROMAN, sScore);
  glPopMatrix();

  ifstream inFile("Snake.bin", ios_base::binary);
  while (inFile.peek() != EOF)
     inFile >> sHightScore;
  inFile.close(); hightScore =
  atoi(sHightScore); glPushMatrix();
  glTranslatef(w / (1.6), h / (1.05), 0);
  glScalef(0.3f, 0.3f, 0.3f);
  draw_string(GLUT_STROKE_ROMAN, "High score:");
  glPopMatrix(); glPushMatrix(); glTranslatef(w / (1.2), h
  / (1.05), 0); glScalef(0.3f, 0.3f, 0.3f);
  draw_string(GLUT_STROKE_ROMAN, sHightScore);
  glPopMatrix();

  glFinish();
  glutSwapBuffers();
}

void display()
```

```
{ switch (key1)
  { case 1:
     glClear(GL_COLOR_BUFFER_BIT);
     glBegin(GL_POLYGON);
     glColor3f(0.0, 0.0, 0.0);
     glVertex3f(0.0, 800.0, 0.0);
     glColor3f(0.0, 0.0, 0.0);
     glVertex3f(0, 700.0, 0.0);
     glColor3f(0.0, 0.0, 0.0);
     glVertex3f(1400.0, 700.0,
     0.0); glColor3f(0.0, 0.0, 0.0);
     glVertex3f(1400.0,  800.0,
     0.0); glEnd(); for (int i = 0; i <
     2; i++) m[i].DrawFruct();
     for (int i = 0; i < 10; i++)
        u[i].DrawBomb();
     DrawSnake();
     DrawScore();
     break;
  case 2:
     DrawExit();
     break;
  case 3:
     DrawMenu();
     break;
  case 4:
  DrawRules();
  break; }
  glFlush();
}

void fjfjfh()
{ for (int i = 0; i < 2;
  i++) m[i].New();
  for (int i = 0; i < 10; i++)
     u[i].New();

    s[0].x = 25;
s[0].y = 15; }

void MyKeyboard(int key, int a, int b)
```

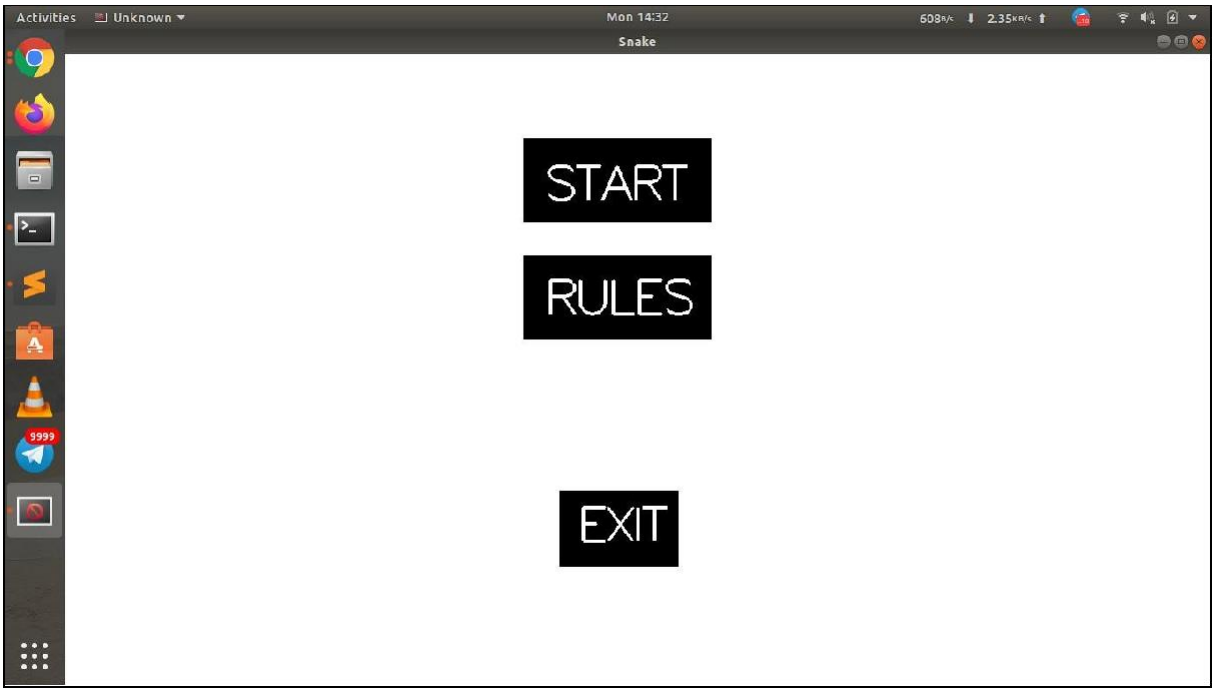
```
{ switch (key)
  { case 101: //
  up dir = 0;
  break;
  case 102: // right
     dir = 2;
     break;
  case 100: // left
     dir = 1;
     break;
  case 103: // down
     dir  =  3;
     break;
  case 27: // escape
     exit(0);
     break;
    }
}

void MousePressed(int button, int state, int ax, int ay)
{ down = button == GLUT_LEFT_BUTTON && state == GLUT_LEFT;
  if (down) {
     if (key1 == 3) { if (ax > (540.0) && ax < (670.0) && ay > (490.0) &&
        ay < (600.0)) { exit(0); } if (ax > (500.0) && ax < (705.0) && ay >
        (230.0) && ay < (330.0)) {
           d = 1;
           glClear(GL_COLOR_BUFFER_BIT);
           fjfjfh();
           key1 = 4; display(); } if (ax > (500.0) && ax < (705.0) && ay >
        (100.0) && ay < (190.0)) {
           key1 = 1;
           d = 2;
           num = 5;
           Score =
           0;
           fjfjfh();
           display();
        } } if (key1 == 2) { if (ax > (580.0) && ax < (670.0) && ay > (660.0)
     && ay < (690.0)) { exit(0); } if (ax > (540.0) && ax < (670.0) && ay >
     (580.0) && ay < (610.0)) {
```

```
            key1 = 3; d = 1;
            glClear(GL_COLOR_BUFFER_BIT)
            ;
            fjfjfh();
            DrawMenu();
        } } if (key1 == 4) { if (ax > (520.0) && ax < (680.0) && ay > (580.0)
      && ay < (610.0)) {
            key1 = 3;
            d = 1;
            glClear(GL_COLOR_BUFFER_BIT);
            fjfjfh();
            DrawMenu();
        }
      } }
   glutMouseFunc(MousePressed);
}

void CreateGlutWindow()
{ glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA); // Mode selection: single buffer
and RGBA colors
   glutInitWindowSize(w, h);
   glutCreateWindow("Snake");
   glClearColor(1.0, 1.0, 1.0, 1.0);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D(0, w, 0, h);
```

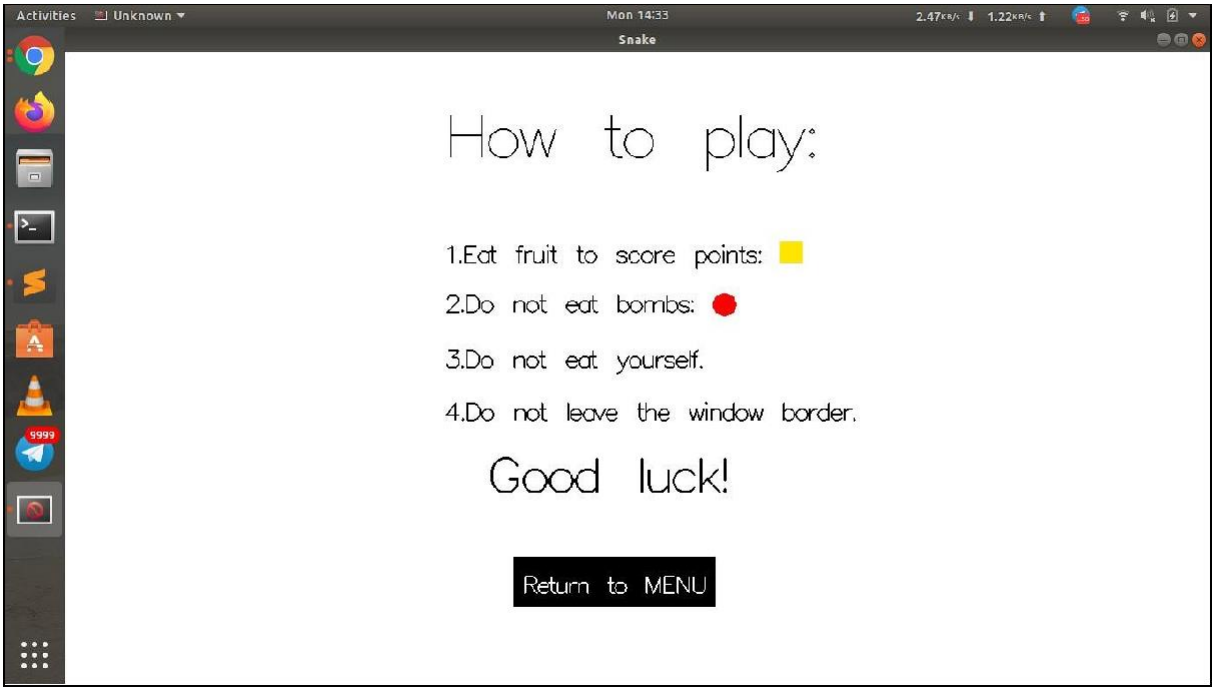
```
}

void timer(int = 0)
{ if (d == 2)
  { display();
  }
  Tick();

  glutTimerFunc(80, timer, 0);
}

int main(int argc, char** argv)
{
  glutInit(&argc, argv);
  CreateGlutWindow();
  glutDisplayFunc(display);
  glutTimerFunc(80, timer, 0);
  glutSpecialFunc(MyKeyboard);
  glutMouseFunc(MousePressed);
  glutMainLoop(); //  enter GLUT event processing loop
  return 0;
}
```

Screenshots

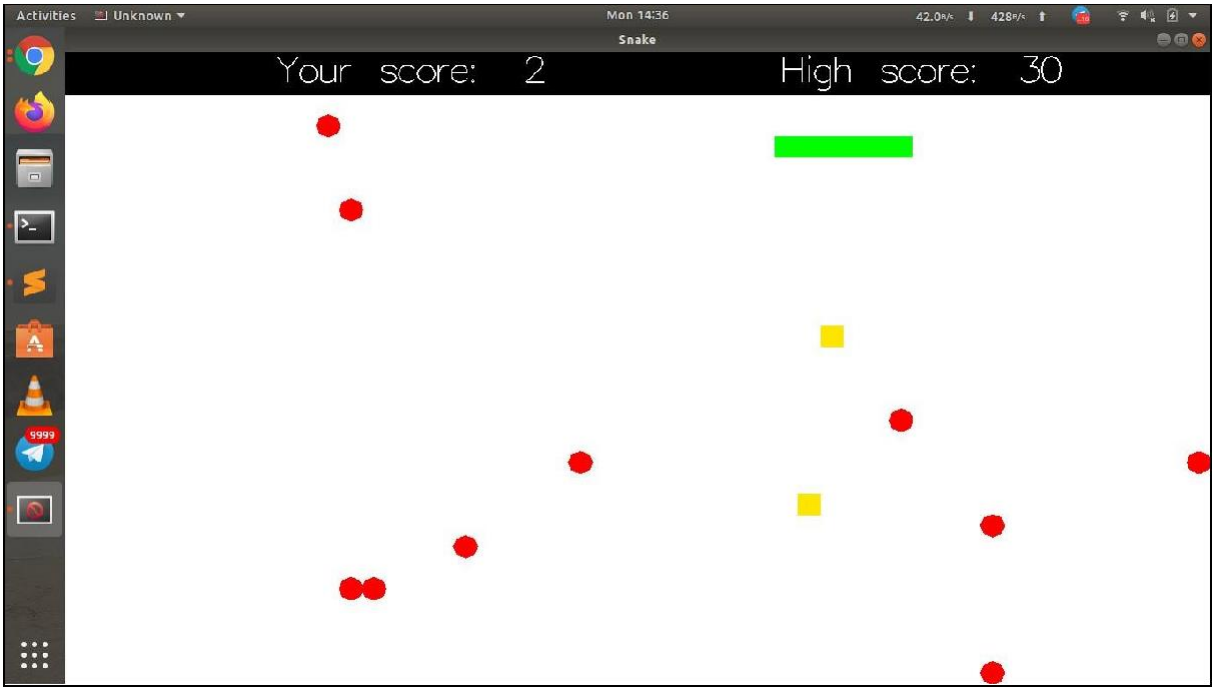
Start screen:



Upon clicking the rules button:



Start the game:



Game over window: