Rajshahi University of Engineering & Technology

Computer Science & Engineering

Project on Computer Graphics Animation

Course Title: Sessional based on CSE 4201

Course Code: CSE 4202

Submitted To:

Md. Zahirul Islam

Assistant Professor

Department of Computer Science & Engineering

Submitted By:

Sumiaya Akter

Roll: 1803018

Section: A

Department: Computer Science & Engineering

Date of Submission: April 22, 2024

Problem Statement:

Implementation of a Computer Graphics and Animations Project.

Summary:

The provided code is a simple graphical animation that simulates a scene with fish swimming across the screen and food balls rising from the top. It uses various libraries to accomplish this, including graphics.h for drawing operations and cstdlib, cmath, and ctime for randomization and mathematical operations. The code initializes a custom graphics window with a specified width and height, providing a larger space for the animation to occur.

Within the main function, arrays are used to manage the properties of fish and food balls, such as position, speed, color, and size. The animation loop continuously updates the scene by clearing the screen, drawing the background, and rendering the fish and food balls. The fish move horizontally with slight vertical variations, while the food balls move downward. If a fish and a food ball collide, the food ball is "eaten" and reset to the bottom of the screen with a new random x-position. The loop includes a delay to control the animation speed, providing a smoother visual experience.

Code:

```
#include <iostream>

#include <graphics.h>

#include <cstdlib>

#include <cmath>

#include <ctime>


using namespace std;


void fish(int x, int y, int color, int bodyWidth) {

   setcolor(color);
```

```
    ellipse(x, y, 0, 360, bodyWidth, 20);
    circle(x + bodyWidth / 2, y - 5, 5);
    line(x - bodyWidth, y, x - bodyWidth - 20, y - 30);
    line(x - bodyWidth, y, x - bodyWidth - 20, y + 30);
    line(x - bodyWidth - 20, y - 30, x - bodyWidth - 20, y + 30);
    line(x + 10, y - 18, x - 15, y - 40);
    line(x - 10, y - 18, x - 15, y - 40);
    line(x + 10, y + 18, x - 15, y + 40);
    line(x - 10, y + 18, x - 15, y + 40);
}

bool fishEatsFood(int fishX, int fishY, int foodX, int foodY) {
    int dx = fishX - foodX;
    int dy = fishY - foodY;
    return sqrt(dx * dx + dy * dy) < 30;
}

int main() {
    srand(time(0));
    int width = 1024;
    int height = 768;
    initwindow(width, height, "Fish Animation");

    const int fishCount = 20;
    int fishX[fishCount];
    int fishY[fishCount];
    int fishSpeed[fishCount];
```

```
int fishColor[fishCount];
int fishWidth = 80;

for (int i = 0; i < fishCount; i++) {
   fishX[i] = rand() % width;
   fishY[i] = rand() % height;
   fishSpeed[i] = 2 + rand() % 4;
   fishColor[i] = COLOR(rand() % 256, rand() % 256, rand() % 256);
}

const int foodCount = 20;
int foodX[foodCount];
int foodY[foodCount];
int foodColor[foodCount];
int foodSize[foodCount];

for (int i = 0; i < foodCount; i++) {
   foodX[i] = rand() % width;
   foodY[i] = rand() % 100;
   foodColor[i] = COLOR(rand() % 256, rand() % 256, rand() % 256);
   foodSize[i] = 10 + rand() % 10;
}

while (true) {
   cleardevice();

   setfillstyle(SOLID_FILL, CYAN);
```

```
bar(0, 0, width, height);

for (int i = 0; i < fishCount; i++) {
   fish(fishX[i], fishY[i], fishColor[i], fishWidth);
   fishX[i] += fishSpeed[i];
   fishY[i] += rand() % 11 - 5;

   if (fishX[i] > width) {
      fishX[i] = -fishWidth;
   }
}

for (int i = 0; i < foodCount; i++) {
   setcolor(foodColor[i]);
   setfillstyle(SOLID_FILL, foodColor[i]);
   circle(foodX[i], foodY[i], foodSize[i]);
   floodfill(foodX[i], foodY[i], foodColor[i]);

   foodY[i] += 5;



   for (int j = 0; j < fishCount; j++) {
      if (fishEatsFood(fishX[j], fishY[j], foodX[i], foodY[i])) {
         foodX[i] = rand() % width;
         foodY[i] = 0;
         break;
      }
```

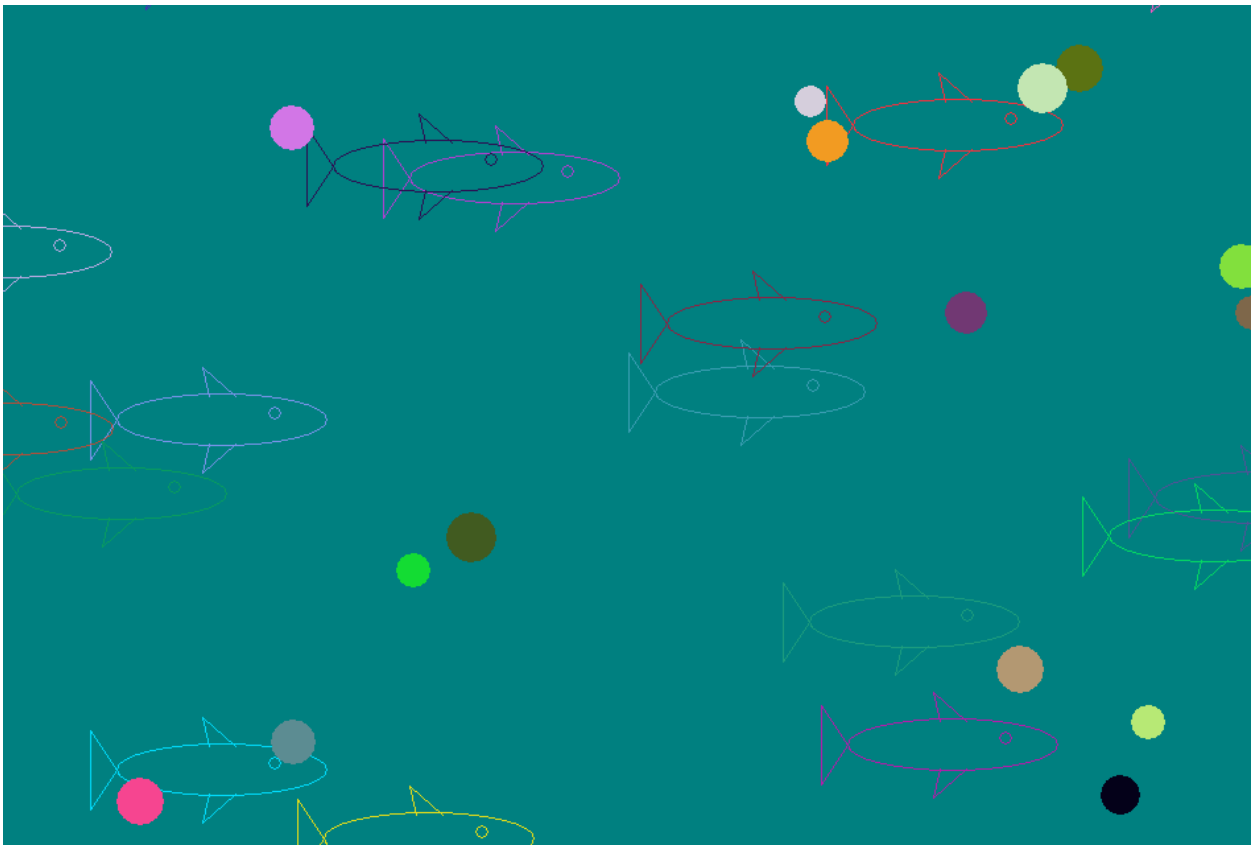
```
        }

        if (foodY[i] > height) {
            foodX[i] = rand() % width;
            foodY[i] = 0;
        }
      }

      delay(50);
   }

   closegraph();
   return 0;
}
```

Output:



Conclusion:

This code provides a simple yet effective demonstration of a graphical animation featuring fish and food balls. Through the use of randomization, the animation exhibits varied movement patterns and a dynamic environment. The code effectively manages the position and speed of multiple objects, allowing for interaction between fish and food, leading to a more engaging and realistic scene. With clear step-by-step logic and the use of graphics functions, this code serves as a foundational example for creating animated scenes in a graphical environment.