**INTRODUCTION**

* 1. **Introduction to Computer Graphics**

Computer graphics started with the display of data on hardcopy plotters and Cathode Ray Tube (CRT) screens soon after the Introduction of computers themselves. It has grown to include the Creation, Storage and Manipulation of Models and Images of objects. These models come from a diverse and expanding set of fields, and include physical, mathematical, engineering,

architectural and even conceptual structures, natural phenomenon and so on. Computer graphics today is largely interactive: the user controls the contents, structure, and appearance of objects and of their displayed images by using input devices, such as a keyboard, mouse, or touchsensitive

panel on the screen. Because of the close relationship between the input devices and the display, the handling of such devices is included in the study of computer graphics.

* 1. **Uses of Computer Graphics:**

Computer graphics is used in many different areas of industry, business, government, education,

entertainment etc.

**User Interfaces**

Word-processing, spreadsheet and desktop-publishing programs are typical applications of such

user-interface techniques.

**Interactive Plotting in Business, Science and Technology**

The common use of graphics is to create 2D and 3D graphs of mathematical, physical and

economic functions, histograms, and bar and pie charts.

**Computer Aided Drafting and Design (CAD)**

In CAD, interactive graphics is used to design components and systems of mechanical, electrical

and electronic devices including structures such as buildings, automobile bodies, aero planes,

ship hulls etc.

**Simulation and Animation for Scientific Visualization and Entertainment**

Computer-produced animated movies are becoming increasing popular for scientific and engineering visualization. Cartoon characters will increasingly be modeled in the computer as 3D shape descriptions whose movements are controlled by computer commands.

**2D Graphics**

These editors are used to draw 2D pictures (line, rectangle, circle and ellipse) alter those with operations like cut, copy and paste. These may also support features like translation, rotation etc.

**3D Graphics**

These editors are used to draw 3D pictures (line, rectangle, circle and ellipse).These may also support features like translation, rotation etc.

* 1. **ADVANTAGES**
* Scientific visualization
* Information visualization
* Computer vision
* Image processing
* Computational geometry
* Computational topology
* Applied mathematics

**SYSTEM REQUIREMENTS**

System requirements are intended to communicate in precise way, the functions that the system must provide. To reduce ambiguity, they may be written in a structured form of natural language supplemented by tables and system models.

**3.1** **HARDWARE REQUIREMENTS**

The physical components required are:

* Processor - Pentium Pro
* Memory - 128MB RAM
* 40GB Hard Disk Drive

**3.2 SOFTWARE REQUIREMENTS**

The software used in building this program are as specified:-

* Operating system – Windows 10
* Tools : DevC++
* Graphics Library – graphics.h

**Introduction to Snake Game**

The Snake Game is a classic game that has been popular for several decades. In this game, the player controls a snake that moves around on a board, trying to eat food and avoid running into walls or the snake's own tail. This project is a console-based implementation of the Snake Game in C++.

**Objective:**

The main objective of this project is to create a simple and fun game that can be played in the console window. It is also an opportunity to practice C++ programming and learn about game development.

**Features:**

* The game has a simple UI, with the game board and the score displayed in the console window.
* The snake is controlled using arrow keys.
* The snake grows longer as it eats food, making the game progressively more challenging.
* The game ends when the snake eats its own tail.
* The player score is displayed at the end of the game.

**Algorithm:**

**STEP 1:**  Initialize the game board and the snake’s starting position:

1. Create a 2D array for the game board.
2. Set the snake’s starting position in the middle of the board.
3. Place a randomly generated food item on the board.

**STEP 2:** Display the game board and the score:

1. Print the game board to the console.
2. Print the score to the console.

**STEP 3:** Update the snake’s position based on the direction:

1. If the user input is left, decrement the snake’s x-coordinate.
2. If the user input is right, increment the snake’s x-coordinate.
3. If the user input is up, decrement the snake’s y-coordinate.
4. If the user input is down, increment the snake’s y-coordinate.

**STEP 4:** If the snake’s head collide with its own body, end the game:

**STEP 5:** If the snake’s head collides with the food, increment the score and add a new segment to the snake:

1. Check if snake’s head collides with the food.
2. If there is a collision, increment the score and add a new segment to the snake.

**STEP 6:** Remove the tail segment to the snake to keep it the same length:

1. Remove the last segment of the snake.

**STEP 7:**  Update the game board with the new positions of the snake and the food.

**STEP 8:** Display the updated game board and score:

1. Print the updated game board to the console.
2. Print the updated score to the console.

**STEP 9:** Display the final score and end the game:

1. Print the final score to the console.
2. End the game.

**CODE:**

* **Main.cpp**

*#include <graphics.h>*

*#include "snakeBody.h"*

*#include "food.h"*

*int main()*

*{*

*initwindow(800,510,"SNAKE GAME");*

*snakeBody body;*

*food fruit;*

*int length,count=0;*

*bool playing=true;*

*char lengtharr[3];*

*fruit.generate(body.getPosx(), body.getPosy());*

*int page=0;*

*while(1)*

*{*

*setactivepage(page);*

*setvisualpage(1-page);*

*cleardevice();*

*setcolor(BLUE);*

*setfillstyle(SOLID\_FILL, BLUE);*

*if(GetAsyncKeyState(VK\_LEFT))*

*{ body.changeDirTo(LEFT); }*

*if(GetAsyncKeyState(VK\_UP))*

*{ body.changeDirTo(UP); }*

*if(GetAsyncKeyState(VK\_RIGHT))*

*{ body.changeDirTo(RIGHT); }*

*if(GetAsyncKeyState(VK\_DOWN))*

*{ body.changeDirTo(DOWN); }*

*if(GetAsyncKeyState(VK\_ESCAPE))*

*break;*

*if(playing==true && !body.update())*

*{*

*playing = false;*

*}*

*body.drawSnake();*

*if(fruit.update(body.getPosx(), body.getPosy()))*

*{*

*fruit.generate(body.getPosx(), body.getPosy());*

*body.appendSnake();*

*}*

*// BOXES*

*setcolor(BLUE);*

*rectangle(0,0,30,510);*

*rectangle(30,0,480,30);*

*rectangle(480,0,510,510);*

*rectangle(30,480,480,510);*

*rectangle(510,0,800,20);*

*rectangle(510,250,800,270);*

*rectangle(780,20,800,250);*

*rectangle(510,490,800,510);*

*rectangle(780,270,800,490);*

*setfillstyle(SOLID\_FILL, BLUE);*

*floodfill(15,250,BLUE);*

*floodfill(250,15,BLUE);*

*floodfill(495,250,BLUE);*

*floodfill(250,495,BLUE);*

*floodfill(550,260,BLUE);*

*floodfill(550,10,BLUE);*

*floodfill(790,150,BLUE);*

*floodfill(550,500,BLUE);*

*floodfill(790,300,BLUE);*

*// SCORE*

*settextstyle(SANS\_SERIF\_FONT, HORIZ\_DIR, 5);*

*setcolor(GREEN);*

*outtextxy(550, 70, "LENGTH");*

*outtextxy(520, 130, "ACHIEVED");*

*length = body.getlength();*

*lengtharr[0]=char(48+length/10);*

*lengtharr[1]=char(48+length%10);*

*lengtharr[2]='\0';*

*outtextxy(620, 190, lengtharr);*

*// STATUS*

*settextstyle(SANS\_SERIF\_FONT, HORIZ\_DIR, 4);*

*outtextxy(520, 300, "STATUS :-");*

*if (body.getlength() == 30)*

*{*

*outtextxy(520, 350, "YOU WON !");*

*playing=false;*

*}*

*else if(playing)*

*{*

*outtextxy(520, 350, "PLAYING");*

*}*

*else*

*{*

*outtextxy(520, 350, "GAME OVER");*

*}*

*settextstyle(SANS\_SERIF\_FONT, HORIZ\_DIR, 2);*

*outtextxy(520,400,"PRESS 'ESC' to EXIT");*

*fruit.draw();*

*page = 1-page;*

*delay(100);*

*}*

*getch();*

*closegraph();*

*}*

* **Snakebody.h**

*#include <graphics.h>*

*enum DIR {LEFT , UP , RIGHT , DOWN}; // ENUM MAKES THE WORK EASY BY PROVIDING NAMES INSTEAD OF NUMBERS*

*////// POSITION STRUCTURE ////// -- ESPICIALLY TO BE USED IN SNAKE ONLY DUE TO ITS PRESET VALUES*

*struct POS*

*{*

*int x,y;*

*POS()*

*{*

*x = -50;*

*y = 0;*

*}*

*};*

*////// SNAKE CLASS //////*

*class snakeBody*

*{*

*private:*

*POS arr[31]; // THIHS ARRAY IS GOING TO CONTAIN THE POSITIONS OF WHOLE SNAKE*

*int direction; // WORKING WITH ENUM !!!*

*int length; // OBVIOUSLY LENGTH*

*public :*

*snakeBody()*

*{*

*arr[0].x = 30; // PROVIDING THE SNAKEHEAD ITS INITIAL VALUE*

*arr[0].y = 30; // PROVIDING THE SNAKEHEAD ITS INITIAL VALUE*

*length = 2; // INITILAL LENGTH = 2*

*direction = RIGHT; // GOING IN RIGHT DIRECTION*

*}*

*void drawSnake();*

*void appendSnake(); // IT IS RESPONSIBLE FOR INCREMENT IN SNAKE'S LENGTH*

*void changeDirTo(int);*

*int update();*

*int getPosx(); // THESE FUNCTIONS ARE FOR OBVIOUS PURPOSES*

*int getPosy(); // THESE FUNCTIONS ARE FOR OBVIOUS PURPOSES*

*int getlength(); // THESE FUNCTIONS ARE FOR OBVIOUS PURPOSES*

*};*

*void snakeBody::drawSnake() // DRAWING THE WHOLE SNAKE*

*{*

*for(int i=0; i<length ; i++)*

*{*

*setcolor(BLUE);*

*rectangle(arr[i].x , arr[i].y , arr[i].x+30 , arr[i].y+30);*

*if(i==0) // HEAD IS OF DIFFERENT COLOR*

*setfillstyle(SOLID\_FILL , GREEN);*

*else // REST OF THE BODY*

*setfillstyle(SOLID\_FILL , LIGHTGREEN);*

*floodfill(arr[i].x+15 , arr[i].y+15 , BLUE);*

*}*

*}*

*void snakeBody::appendSnake()*

*{*

*if(length < 30) // PREVENTS THE LENGTH FROM GOING GREATER THAN 29*

*length++;*

*}*

*void snakeBody::changeDirTo(int newdir)*

*{*

*if(newdir == LEFT || newdir == RIGHT)*

*{*

*if(direction == UP || direction == DOWN)*

*direction = newdir;*

*}*

*else if(newdir == UP || newdir == DOWN)*

*{*

*if(direction == LEFT || direction == RIGHT)*

*direction = newdir;*

*}*

*}*

*int snakeBody::update() // THIS FUNCTION ENFORCES THE BASIC*

*{ // ALGORITHM FOR MOVING SNAKE*

*for(int i=1 ; i<length ; ++i) // BY PROVIDING ALL THE PREVIOUS PARTS*

*{ // THE COORDINATES OF CURRENT PART*

*if(arr[0].x == arr[i].x && arr[0].y == arr[i].y) // HENCE [1] GETS THE POSITION OF [0]*

*{ // [2] GETS THE POSITION OF [1] AND SO ON...*

*return 0; // AND LASTLY HEAD([0]) IS GIVEN LATEST VALUES*

*}*

*}*

*for(int i=length ; i>=0 ; --i)*

*{*

*if(i==0)*

*{*

*arr[1].x = arr[0].x;*

*arr[1].y = arr[0].y;*

*}*

*else*

*{*

*arr[i].x = arr[i-1].x;*

*arr[i].y = arr[i-1].y;*

*}*

*}*

*if(direction == LEFT)*

*{*

*arr[0].x -= 30;*

*if(arr[0].x == 0)*

*{*

*arr[0].x = 450;*

*}*

*}*

*else if(direction == RIGHT)*

*{*

*arr[0].x += 30;*

*if(arr[0].x == 480)*

*{*

*arr[0].x = 30;*

*}*

*}*

*else if(direction == UP)*

*{*

*arr[0].y -= 30;*

*if(arr[0].y == 0)*

*{*

*arr[0].y = 450;*

*}*

*}*

*else if(direction == DOWN)*

*{*

*arr[0].y += 30;*

*if(arr[0].y == 480)*

*{*

*arr[0].y = 30;*

*}*

*}*

*return 1;*

*}*

*int snakeBody::getPosx()*

*{*

*return arr[0].x;*

*}*

*int snakeBody::getPosy()*

*{*

*return arr[0].y;*

*}*

*int snakeBody::getlength()*

*{*

*return length;*

*}*

* **food.h**

*#include <graphics.h>*

*#include <ctime> // required for generating random numbers*

*class food*

*{*

*private:*

*POS foodPos; // position*

*public :*

*void draw(); // functtion to draw the food on the graphics window*

*void generate(int, int); // function to generate food everytime snake eats it*

*bool update(int, int); // this is to give us the status of food(whether it is been eaten or not)*

*};*

*void food::draw() // DRAW THE FOOD*

*{*

*setcolor(RED);*

*rectangle(foodPos.x, foodPos.y, foodPos.x+30, foodPos.y+30);*

*setfillstyle(INTERLEAVE\_FILL, RED);*

*floodfill(foodPos.x+15, foodPos.y+15, RED);*

*}*

*void food::generate(int snakeHeadx, int snakeHeady) // GENERATE NEW POSITION FOR THE PARTICLE*

*{*

*srand(time(0));*

*foodPos.x = 30\*(rand()%15 + 1);*

*srand(time(0));*

*foodPos.y = 30\*(rand()%15 + 1);*

*if (foodPos.x == snakeHeadx && foodPos.y == snakeHeady)*

*generate(snakeHeadx, snakeHeady);*

*}*

*bool food::update(int snakeHeadx, int snakeHeady) // GIVE US THE STATUS OF THAT DAMN FOOD !!*

*{*

*if (foodPos.x == snakeHeadx && foodPos.y == snakeHeady)*

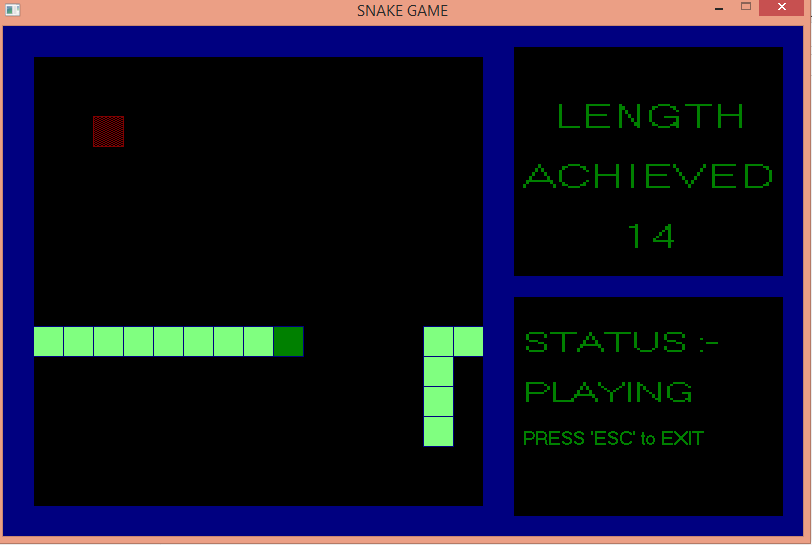
*return true;*

*else*

*return false;*

*}*

**Output:**



**Conclusion:**

In conclusion, the Snake Game in C++ is a fun and challenging game that is easy to play and learn. It is a great project for anyone interested in game development or C++ programming. The code is well-documented and can be easily modified or extended to add new features or functionality.