**COMPUTER PROGRAMMNG PROJECT**

**PAC-MAN GAME**

****

**SUMBITTED BY:**

**MAHNOOR KHALID (01-134251-098)**

**MAHNOOR ZAHRA (01-134251-035)**

**SUBMITTED TO:**

**MOHSIN JAVED**

**DEPARTMENT OF COMPUTER SCIENCE**

**BAHRIA UNIVERSITY, ISLAMABAD**

**SCOPE:**

The scope of this project is to develop a fully functional Pacman game in C++ that can be played in a console environment. The game will include features such as maze generation, character movement, score tracking, and ghost AI. The focus will be on implementing the core gameplay mechanics and ensuring a smooth and enjoyable gaming experience. Advanced graphics or additional features beyond the basic Pacman gameplay will not be included in this project.

OBJECTIVES:

The objectives of this project are as follows:

* Implement the Pacman game mechanics, including character movement, score tracking, and ghost AI.
* Create a visually appealing console-based representation of the game using ASCII characters.
* Ensure smooth and responsive gameplay by optimizing the algorithms and code.
* Test and debug the game to ensure it functions correctly and handles user input accurately.
* Provide a comprehensive documentation of the project, including an explanation of the technical approach, schedule, flowchart, implementation details, and game outputs.

TECHNICAL APPROACH:

**Defining Game Elements:** The key elements of the game, such as Pacman, the ghosts, walls, and pellets, will be represented using ASCII characters. The maze will be created dynamically based on preset patterns, and characters will move within the maze's boundaries.

**Handling User Input:** The program will detect user input to control Pacman's movement. The player can use the arrow keys or the WASD keys to change Pacman’s direction.

**Character Movement:** The movement of Pacman and the ghosts will be governed by simple algorithms. Characters will move one step at a time, with collision detection to ensure proper interaction between characters and the maze elements.

**Score Tracking:** The game will track the player's score as they collect pellets. Each pellet collected will increase the score, and the game will end when all pellets are eaten.

**Reading the previous scores:** We have added a function that allow the user to read all the previous scores.

**Deleting the whole score:** We have added a function using trunc mode that firstly ask the user for deleting or not and per users move it will delete the whole previous scores.

**Highest Score:** A function that updates the score as the user collect more pallets than the previous one.

**Ghost AI:** The ghosts will have their own AI, chasing Pacman to try and catch it. The AI will be based on basic distance calculations to figure out the best direction for the ghosts to move in order to approach Pacman.

**Game Over Conditions:** The game will conclude when either the player collects all the pellets or when Pacman is caught by a ghost. At the end of the game, the final score will be shown, and players will have the option to restart

PROJECT PLAN:

**Project Overview:**

* Define the project's objectives and scope.
* Confirm the roles and responsibilities of the team members.
* Establish project milestones and set the timeline.

**Requirements Gathering:**

* Identify the specific requirements for the game.
* Determine the gameplay mechanics, visuals, sound effects, and scoring system.
* Utilize SFML to enhance the game's visual appeal.

**Design Phase:**

* Collaboratively design the game's structure and flow.
* Create the maze layout using ASCII characters.
* Plan the ghost behavior and the user interface.
* Finalize the design documentation.

**Integration and Testing:**

* Integrate all components into a cohesive game.
* Perform thorough testing to ensure functionality.
* Ensure smooth interaction between all modules.

**Documentation:**

* Create comprehensive documentation covering design, implementation, and testing procedures.

**Final Review and Delivery:**

* Review the project to ensure completion.
* Address any outstanding issues.
* Package the game for delivery along with the documentation.

IMPLEMENTATION:

The implementation of the Pacman game involves writing C++ code to define the classes, functions, and algorithms required for the game mechanics. The code will follow best practices for readability, maintainability, and performance. The implementation will include the following key aspects:

1. **Maze generation:** The code will include an algorithm to generate the maze dynamically based on predefined patterns. The maze will consist of walls, pallets, and empty spaces.
2. **Character movement:** Functions will be implemented to handle the movement of the Pacman character and ghosts within the maze. The movement will be controlled based on user input or AI behavior.
3. **Score tracking:** The code will include variables and functions to track the player's score as they collect pallets. The score will be incremented for each pallet collected.
4. **Reading Scores:** Utilizes file input stream (ifstream) to open and read the score.txt file. Reads scores line by line and prints them to the console.
5. **High Score:** Reads all scores from score.txt. Compares each score to find the maximum. Returns the highest score as an integer.
6. **Deleting Scores:** Opens score.txt using ofstream with ios::trunc mode to truncate the file. Outputs a confirmation message after deletion.
7. **Ghost AI:** Algorithms will be implemented to determine the movement of the ghost characters. The AI behavior will aim to chase the Pacman character by calculating the optimal direction based on distance calculations.
8. **Game over conditions:** The code will check for game over conditions, such as the player collecting all pallets or the Pacman character getting caught by a ghost. The game will end when these conditions are met.

CODE OF THE GAME:

//Adding the Libraries.

#include<iostream>

#include<ctime>

#include<math.h>

#include<cstdlib>

#include<conio.h>

#include<fstream>

#include<string>

using namespace std;

//functions:

void writegameintro() {

ofstream introfile("gameintro.txt");

if (!introfile) {

cout << "Error opening game intro file." << endl;

return;

}

introfile << "Welcome to Pac-Man Game" << endl;

introfile << "Your goal is to collect all pallets." << endl;

introfile << "Use W, D, S, A for movement." << endl;

introfile.close();

}

void gameintroread() {

ifstream introfile("gameintro.txt");

if (!introfile) {

cout << "Error opening game intro file." << endl;

return;

}

system("CLS");

string line;

while (getline(introfile, line)) {

cout << line << endl;

}

introfile.close();

cout << "Enter any key to start..." << endl;

cin.ignore();

}

void clearScreen() {

system("cls");

}

void generateMap(char arr[20][20], int& count, int ran) {

for (int x = 0; x < 20; x++)//20 rows

{

for (int y = 0; y < 20; y++)//20 columns

{

if (x == 0 || x == 19 || y == 0 || y == 19) //Maze rows and columns

{

arr[x][y] = '#';//Wall

}

else if (x == ran + 1)// Random obstacle wall-1

{

arr[x][y] = '#';

if (y == 4 || y == 9 || y == 13) //adding pallets in these col

{

arr[x][y] = '.';

count++;

}

}

else if (x == ran + 3)// Random obstacle wall-2

{

arr[x][y] = '#';

if (y == 3 || y == 7 || y == 13)//adding pallets in these col

{

arr[x][y] = '.';

count++;

}

}

else if (x == ran + 9)//Random obstacle wall-3

{

arr[x][y] = '#';

if (y == 4 || y == 9 || y == 13)// adding pallets in these col

{

arr[x][y] = '.';

count++;

}

}

else {

arr[x][y] = '.';//adding overall pallets

count++;

}

}

}

}

void printMap(char arr[20][20], int gr1, int gc1, int gr2, int gc2, int score) {

char ghost1 = 'G', ghost2 = 'G';//Intializing

for (int x = 0; x < 20; x++)//rows

{

for (int y = 0; y < 20; y++)//columns

{

if (x == gr1 && y == gc1)

cout << "\033[1;34m" << ghost1 << " ";//Blue Ghost

else if (x == gr2 && y == gc2)

cout << "\033[1;34m" << ghost2 << " ";//Blue Ghost

else if (arr[x][y] == '#')

cout << "\033[1;32m" << arr[x][y] << " ";//Green Walls

else if (arr[x][y] == '.')

cout << "\033[1;32m" << arr[x][y] << " ";//Green Walls

else if (arr[x][y] == 'P' || arr[x][y] == '^' || arr[x][y] == 'v' || arr[x][y] == '<' || arr[x][y] == '>')

cout << "\033[1;31m" << arr[x][y] << " ";//Red pacman

else

cout << "\033[0m" << arr[x][y] << " ";

}

cout << endl;

}

}

//write file

void savescore(int score) {

ofstream scorefile("score.txt", ios::app);//Add new scores without disturbing the previous ones.

if (!scorefile) {

cout << "Error saving score." << endl;

return;

}

scorefile << score << endl;

scorefile.close();

}

//read file

void readscores() {

ifstream scorefile("score.txt");

if (!scorefile) {

cout << "No score history found." << endl;

return;

}

cout << "\nPrevious Scores:\n";

int s;

while (scorefile >> s)//Displays the scores line by line.

{

cout << "- " << s << endl;

}

scorefile.close();

}

//write file

void deletescores() {

ofstream scorefile("score.txt", ios::trunc);//Mode for deleting the overall scores

if (!scorefile) {

cout << "Error deleting scores." << endl;

return;

}

cout << "All scores deleted successfully.\n";

scorefile.close();

}

//read file

int highscore() {

ifstream scorefile("score.txt");

int highscore = 0, temp;//temporary storing the value

if (!scorefile) {

return 0;

}

while (scorefile >> temp) {

if (temp > highscore)

highscore = temp;

}

scorefile.close();

return highscore;

}

int main() {

writegameintro();

gameintroread();

//Assigning the variabes.

char arr[20][20];//Maze size 20X20 (20rows,20columns)

int x, y, a = 1, b = 1, count = 0, score = 0;

char move;

srand(time(0));

int gc1 = 18, gr1 = 10, gc2 = 3, gr2 = 9;//initial positions of the ghost

char ghost1 = 'G';

char ghost2 = 'G';

char pacman = 'P';

int ran = (rand() % 5 + 3);

//calling of the function

generateMap(arr, count, ran);

arr[b][a] = pacman;//initial position of pacman

cout << "\t\t\tScore: " << score << endl;

cout << "\t\t\tHigh Score: " << highscore() << endl;

printMap(arr, gr1, gc1, gr2, gc2, score);//function calling

//infinite loop for taking user moves

for (int m=0;m==0;) {

cout << "\nEnter Move (W/A/S/D): ";

if (score == count - 1)//winnig condition: all pallets are collected

{

cout << "Game over, You Win!" << endl;

cout << "\tYour Score: " << score << endl;

cout << "High Score: " << highscore() << endl;

savescore(score);

break;

}

move = \_getch();//ask for user move

if (move == 's')//move down

{

b++;

if (b < 19)//keep within bounds

{

arr[b - 1][a] = ' ';

if (arr[b][a] == '#')//prevent from moving into walls

b--;

}

}

else if (move == 'w')//move up

{

b--;

if (b >= 0) //keep within bounds

{

arr[b + 1][a] = ' ';

if (arr[b][a] == '#')//preventing from moveing into walls

b++;

}

}

else if (move == 'a') //left move

{

a--;

if (a >= 0) //keep within bounds

{

arr[b][a + 1] = ' ';

if (arr[b][a] == '#')//preventing from moving into walls

a++;

}

}

else if (move == 'd')//right move

{

a++;

if (a < 19)//keep within bounds

{

arr[b][a - 1] = ' ';

if (arr[b][a] == '#')//preventing from moving into walls

a--;

}

}

//hitng the wall

if (arr[b][a] == '#') {

cout << "\nYou hit a wall. Game over!" << endl;

cout << "\tYour score is: " << score << endl;

cout << "High Score: " << highscore() << endl;

savescore(score);

break;

}

//collecting pallets

if (arr[b][a] == '.') {

score++;

}

//Ghost1 chasing the Pacman(gr1,gc1)

if (rand() % 2 == 0) {

if (b < gr1) gr1--;

else if (b > gr1) gr1++;

}

else {

if (a < gc1) gc1--;

else if (a > gc1) gc1++;

}

//Ghost2 chasing the Pacman(gr2,gc2)

if (rand() % 2 == 0) {

if (b < gr2) gr2--;

else if (b > gr2) gr2++;

}

else {

if (a < gc2) gc2--;

else if (a > gc2) gc2++;

}

//if caught by ghost

if ((b == gr1 && a == gc1) || (b == gr2 && a == gc2)) {

cout << "Game over!" << endl;

cout << "\tYour Score: " << score << endl;

cout << "High Score: " << highscore() << endl;

savescore(score);

break;

}

clearScreen();// Clear the screen for the next round of display

// Assigning the shapes for Pacman (up/down/left/right arrows) and move

if (move == 's') pacman = 'v';//pacman facing down

else if (move == 'w') pacman = '^';//pacman facing up

else if (move == 'd') pacman = '>';//pacman facing right

else if (move == 'a') pacman = '<';//pacman facing left

else cout << "Invalid Move!";

arr[b][a] = pacman;

cout << "\t\t\tScore: " << score << endl;

cout << "\t\t\tHigh Score: " << highscore() << endl;

printMap(arr, gr1, gc1, gr2, gc2, score);//printing the maze after the move

}

//asking for user choice

char choice;

cout << "\nDo you want to view past scores? (y/n): ";

cin >> choice;

if (choice == 'y' || choice == 'Y') {

readscores();//function calling

}

cout << "\nDo you want to delete all past scores? (y/n): ";

cin >> choice;

if (choice == 'y' || choice == 'Y') {

deletescores();//function calling

}

return 0;

}

this C++ program implements a simple text-based **Pacman-like game** on a 20x20 maze grid. The game features Pacman, represented by 'P', which the player controls using the W, A, S, and D keys to navigate through the maze. The maze is composed of walls (#), empty spaces (.), and randomly placed obstacles.

The player collects pallets (denoted by .) scattered throughout the maze to earn points. The game ends when all pallets are collected, and the player wins. However, the game is also challenged by two ghosts ('G'), which randomly chase Pacman. If Pacman collides with any ghost, the game ends in a loss.

The game uses simple logic to handle movement, check for wall collisions, and update Pacman and ghost positions after each move. The clearScreen() function refreshes the screen to display the updated game state after each input. Additionally, the player’s score is displayed at the top of the screen, and the game continues until the player wins or is caught by a ghost. The game also uses colors to distinguish between different elements: Pacman (red), ghosts (blue), walls (green), and pallets (green).

CONCLUSION:

In conclusion, this project aimed to develop a Pacman game using C++ programming language. The project successfully implemented the game mechanics, including maze generation, character movement, score tracking, ghost AI, and game over conditions. The game provides an interactive and engaging experience for the player, with a visually appealing interface using ASCII characters. The project followed a systematic approach, starting with the problem statement, background research, and scope definition. A proposed technical approach was outlined, and a schedule was created to manage the project timeline effectively. The implementation phase involved writing the code, testing, and debugging to ensure correct functionality. The project concluded with the documentation of the project details and the preparation of the final submission. Overall, this project demonstrates the application of C++ programming skills in developing a classic arcade game and showcases the ability to design and implement complex game mechanics.

LINKS: