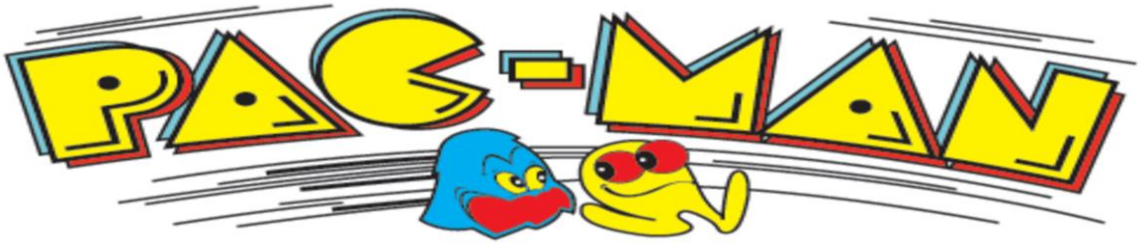
Computer Graphics (UCS505)

Project on



Submitted By

Siddharth Maithani	102153035
Tanisha Maheshwary	102153037
Kashvi Jain	102153038

Group No. 7

B.E. Third Year – COE

Submitted To:

<Lab Instructor name>

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Computer Science and Engineering Department Thapar Institute of Engineering and Technology Patiala – 147001

Table of Contents

INTRODUCTION

Welcome to our project report on Pacman, where we bridge the past and the present through the combined power of OpenGL and C++. Drawing inspiration from the timeless arcade classic, our endeavor is to breathe new life into Pacman, infusing it with modern graphics and functionality. Leveraging the capabilities of OpenGL, a powerful graphics library, we've crafted a visually stunning experience that transports players into a vibrant world of mazes, ghosts, and cherries. With OpenGL's robust rendering capabilities, every pixel of our Pacman game comes alive, immersing players in a captivating adventure.

Complementing OpenGL's graphical prowess is the efficiency and flexibility of C++. As the backbone of our project, C++ provides the framework for Pacman's logic and gameplay mechanics. From managing character movement to handling collision detection, C++ ensures smooth and responsive gameplay that stays true to the original while introducing exciting enhancements. With its extensive standard library and support for object-oriented programming, C++ empowers us to build a Pacman experience that is both nostalgic and innovative.

In this report, we delve into the intricacies of developing Pacman using OpenGL and C++. We'll explore the technical challenges encountered along the way and the creative solutions devised to overcome them. From optimizing rendering performance to implementing complex game mechanics, every aspect of our project reflects a harmonious blend of technology and creativity. Join us as we journey through the development process, uncovering the secrets behind bringing Pacman into the modern age while honoring its timeless legacy. Our report aims to not only showcase the technical achievements but also to inspire future game developers to explore the endless possibilities of combining classic gameplay with modern technology.

Computer Graphics Concepts Use

1. **Initialization and Setup:** Functions like *glutInit*, *glutInitDisplayMode*, *glutInitWindowPosition*, *glutInitWindowSize*, and *glutCreateWindow* are used to set up the graphics environment, initialize the window, and configure display settings.
2. **Rendering:** The *glutDisplayFunc* function is used to define a callback function for rendering graphics. Rendering involves generating images from geometric models and drawing them onto the screen.
3. **Keyboard Input Handling:** Functions like *glutKeyboardFunc* and *glutSpecialFunc* are used to set up callback functions for handling keyboard input, including standard and special keys. This allows for user interaction in the game.
4. **Idle Function:** The *glutIdleFunc* function sets a callback function that is called repeatedly when the application is idle. This function is used for continuous updates, animations, and background tasks.
5. **Event Loop:** The *glutMainLoop* function enters the GLUT event processing loop, where it continuously waits for and handles events such as keyboard input, mouse input, or window resizing. This loop is essential for interactive applications to respond to user actions in real-time.
6. **Matrices and Transformations:** Functions like *glPushMatrix*, *glPopMatrix*, *glTranslatef*, and *glRotatef* are used to apply translation and rotation transformations to objects in the scene. Matrices are fundamental for representing transformations in 3D space.
7. **Texture Mapping:** Functions such as *glGenTextures*, *glBindTexture*, *glTexEnvf*, *glTexParameteri*, and *glTexImage2D* are used to apply textures to 3D objects, enhancing their appearance with detailed imagery or patterns.
8. **Drawing Primitives:** Functions like *glBegin* and *glVertex* are used to define and draw geometric primitives such as points, lines, and polygons. These primitives form the building blocks for creating complex shapes and scenes.
9. **State Management:** Functions like *glEnable* and *glDisable* are used to enable or disable various OpenGL features or states, such as texture mapping or lighting. Proper state management ensures correct rendering behavior and performance optimization.
10. **Coordinate Systems:** Understanding and managing coordinate systems, including object coordinates, world coordinates, and screen coordinates, is crucial for positioning and transforming objects within a 3D scene.

User Defined Functions

Pacman.cpp file functions:

1. **detectGhost():** Checks if the pacman collides with any of the ghosts and updates the game state accordingly.
2. **detectPill():** Checks if the pacman consumes a pill and updates the game state accordingly.
3. **idle():** Handles the game logic when the application is idle, including updating the game state, moving the pacman and ghosts, and checking for collisions.
4. **special(int key, int , int):** Handles special keyboard input (arrow keys) for controlling the pacman's movement direction.
5. **init():** Initializes the OpenGL environment, sets the projection matrix, and loads all necessary textures.
6. **display():** Draws the game scene based on the current game state, including the maze, pacman, ghosts, and interface elements.
7. **resetGame():** Resets the game to its initial state, including resetting the score, lives, game tick, pills, ghosts' positions, and pacman's position.
8. **keyboard(unsigned char key, int,int):** Handles standard keyboard input (such as 'q' to quit, 'p' to pause, 'r' to reset, and space to start) for controlling various aspects of the game.
9. **main(int argc, char** argv):** Initializes GLUT, sets up the window size, creates the window, registers callback functions, and enters the GLUT event processing loop.

Ghost.h header file function:

1. **Class Ghost**: Define a class Ghost representing individual ghosts in the game.
2. **Ghost(float x,float y,colour ghost_colour):** Constructor to initialize the ghost's position, color, direction, speed, and state based on the parameters passed.
3. **void draw():** Draw the ghost using OpenGL functions, applying appropriate textures based on the ghost's state and color.
4. **Utility Functions:**
   - **isAtCentre():** Check if the ghost is at the center of a tile.
   - **getTile():** Get the type of tile at a given position.
   - **getFollowingTile():** Get the type of tile in the direction the ghost is moving.
   - **setGhostNextDirection():** Set the next direction for the ghost based on the position of the pacman.
   - **isWall():** Check if there's a wall in the given direction.
   - getGhostX(): To get the current x-coordinates of the ghost in the maze.
   - getGhostY(): To get the current y-coordinates of the ghost in the maze
5. **Movement Functions:**
   - movement(): Move the ghost in the specified direction.
   - ghostLeavePen(): Move the ghost out of the pen.
   - penMovement(): Move the ghost within the pen.
   - sendGhostToPen(): Send the ghost back to the pen.
   - resetAllGhosts(): Reset the position and state of all ghosts.
6. **Ghost Movement Functions:**
   - redGhostMove(), blueGhostMove(), pinkGhostMove(), yellowGhostMove(): Handle movement logic for each type of ghost based on their state.
7. **void move():** Determine the movement logic for the ghost based on its color.

pacman.h header file functions:

The "pacman.h" header file defines the Pacman class, responsible for managing the player-controlled character in a Pacman game. It includes functions for Pacman movement, collision detection with maze elements, scoring, and drawing Pacman on the screen. Additionally, it provides utilities for handling Pacman's position, direction, and interactions with game elements like pills and portals.

1. **Class Pacman:** Define the Pacman class.
2. **Pacman():** Constructor to initialize the Pacman's position, current direction, and angle.
3. **checkTile():** Function: Check the type of tile Pacman is on and perform appropriate actions such as scoring points, eating pills, or teleporting through portals.
4. **Utility Functions:**
    - ***getTile():*** Get the type of tile at a given position.
    - ***getFollowingTile():*** Get the type of tile in the direction Pacman is moving.
    - ***isAtCentre():*** Check if Pacman is at the center of a tile.
    - ***setDirectionStore():*** Set the direction that Pacman intends to move.
    - ***getPacmanX():*** To return the x-coordinates of Pacman in the maze.
    - ***getPacmanY():*** To return the y-coordinates of Pacman in the maze.
    - ***isWall():***To check if the next tile in the direction of the pacman is a wall/gate.
5. **void draw():** Function to draw the pacman on the screen, applying appropriate textures based on the game tick to simulate mouth movement.
6. **void move():** Update Pacman's position based on the current direction and handle collision detection with walls or gates.

Interface.h header file functions:

The ***"Interface.h"*** header file provides functions and utilities for managing the visual interface elements of a Pacman game. It includes functions for displaying text, drawing the score, lives, high score, and managing game states such as start, game over, and pause. Additionally, it defines functions to draw the background and initialize the start screen. The header also handles the retrieval and updating of the high score from a file. Overall, it serves as a central hub for managing the graphical aspects and user interface of the Pacman game.

- display_Text(): Displays text at a specified position on the screen.
- drawStart(): Draws the "Press SPACE to start" text on the screen.
- drawGameOver(): Draws the "GAME OVER" text on the screen.
- drawPressToPlay(): Draws the "Press 'r' to play again" text on the screen.
- draw_Ready(): Draws the "READY", countdown, or "GO!!" text depending on the game state.
- drawScore(): Draws the current score on the screen.
- drawLives(): Draws the remaining lives of the player.
- getHighScore(): Reads the high score from a file.
- setHighScore(): Sets the high score if the current score is higher.
- drawHighScore(): Draws the high score on the screen.
- drawInterface(): Draws the interface elements including score, high score, and lives.
- drawBackground(): Draws the game background.
- drawStartScreen(): Draws the start screen with the background.

maze.h header file functions:

This header file, "maze.h", provides functionality related to the maze in a Pacman game. It includes definitions for maze tiles, such as walls, paths, pills, big pills, and portals, represented by an enum. Additionally, it declares external texture variables for different maze elements. Functions are provided for translating coordinates to maze positions, drawing the maze on the screen, and resetting the maze state. The maze is represented as a matrix, and drawing is done bottom-up according to this matrix.

- **translateToMazeCoords():** A utility function that translates world coordinates to maze coordinates. It's used to position elements within the maze.
- **translateBottomLeft():** Another utility function that translates the origin to the bottom-left corner of the maze. This is used to position the maze and its elements correctly on the screen.
- **drawMaze():** Draws the maze on the screen using OpenGL. It iterates over the maze matrix, drawing each tile based on its type. Pills and big pills are drawn as textures, and the maze layout is constructed according to the matrix.
- **resetMaze():** Resets the maze state. It sets all eaten pills and big pills back to their initial state, allowing the player to replay the game without affecting the maze layout.

textures.h header file functions:

The textures.h header file manages the loading and binding of textures used in the game.

- loadandbindallTextures(): This function loads all the necessary textures used in the game and binds them to OpenGL texture objects. It enables blending for transparency and then sequentially loads each texture using the load_and_bind_texture function from the loadAndBindTextures.h header.
- drawTexture(int texture, int length, int height, float angle): This function draws a texture on the screen. It takes parameters for the texture ID, length, height, and angle of rotation. It sets up the OpenGL environment for texture drawing, including enabling texture 2D and binding the specified texture. Then, it draws a textured quad with the given dimensions and angle.

loadAndBindTextures.h header file functions:

This header file, load_and_bind_texture.h, provides a function to load and bind textures for use in OpenGL-based applications. Here's an explanation of its functionality:

- **load_and_bind_texture(const char* filename):** This function takes the filename of the image texture to load as input. It uses an external function png_load to read the PNG image data into an image buffer. If the image loading fails, it prints an error message and exits the program.After successfully loading the image, it generates a new texture object using glGenTextures and binds it to GL_TEXTURE_2D. Then, it sets various parameters for the texture, such as wrapping mode and filtering mode, using glTexParameteri.Finally, it uploads the image data to the texture object using glTexImage2D and frees the image buffer memory. The function returns the texture handle, which can be used to refer to the texture in OpenGL rendering operations.

png_load.h header file functions:

This header file, load_and_bind_texture.h, provides a function to load and bind textures for use in OpenGL-based applications. Here's an explanation of its functionality:

- **load_and_bind_texture(const char* filename):** This function takes the filename of the image texture to load as input. It uses an external function png_load to read the PNG image data into an image buffer. If the image loading fails, it prints an error message and exits the program. After successfully loading the image, it generates a new texture object using glGenTextures and binds it to GL_TEXTURE_2D. Then, it sets various parameters for the texture, such as wrapping mode and filtering mode, using glTexParameteri. Finally, it uploads the image data to the texture object using glTexImage2D and frees the image buffer memory. The function returns the texture handle, which can be used to refer to the texture in OpenGL rendering operations.

CODE

For complete code refer: https://github.com/SidM24/PacManGame

```
#include <GL/glut.h>
#include<stdio.h>
#include<string.h>
#include<algorithm>
#include<iostream>
#include<stdlib.h>
#include <png.h>
#include<vector>
#include<fstream>
#include<sstream>
#include<math.h>

using namespace std;

#include "maze.h"
#include "textures.h"
#include "loadAndBindTextures.h"
#include "Interface.h"
#include "pacman.h"
#include "Ghost.h"

//Variables
int frighten_time = 0;
bool frighten = false;
int total_lives = 3;

//score variable would store the score at present moment of time
int score = 00;

//pills stores the total number of pills present on the screen
int pills = 244; //Initially there are 244 pills

//Custom variable to store the current state of the game
typedef enum { BEGIN, PLAY, DIE, OVER, START } gameState;
gameState current_game_state = START;

//Creating the pacman object of the Pacman class
Pacman pacman;

//To keep a record of the time-stamp the current game is in:
int gameTick = 0;

//ghosts
Ghost ghosts[4] = {
    Ghost(13.5f,19.0f,RED),
    Ghost(11.5f,16.0f,BLUE),
    Ghost(13.5f,16.0f,PINK),
    Ghost(15.5f,16.0f,YELLOW)
};

```

```
51    bool paused = true;
52
53
54    void detectGhost() {
55        float pacX = pacman.getPacmanX();
56        float pacY = pacman.getPacmanY();
57        //Checking if any of the ghost has collided with the pacman
58        for (int i = 0; i < 4; i++)
59        {
60            Ghost g = ghosts[i];
61            float gX = g.getGhostX();
62            float gY = g.getGhostY();
63            if (pacX == gX && pacY == gY) {
64                //Checking if any ghost is present at the staring position of the pacman, if yes , then we will reset his position
65                if (g.ghostState == FRIGHTEN) {
66                    //If we have collided with a frightened ghost
67                    score += 200;
68                    ghosts[i].eaten = true;
69                    ghosts[i].eaten_time = gameTick;
70                    ghosts[i].sendGhostToPen();
71                }
72                else {
73                    for (int i = 0; i < 4; i++)
74                    {
75                        if (ghosts[i].getGhostY() >= 6.0f
76                            && ghosts[i].getGhostY() <= 9.0f
77                            && ghosts[i].getGhostX() >= 5.0f
78                            && ghosts[i].getGhostX() <= 19.0f) {
79                            ghosts[i].ghostX = 13.0f;
80                            ghosts[i].ghostY = 19.0f;
81                        }
82                    }
83                    current_game_state = DIE;
84                }
85            }
86        }
87    }
```

```
86    void detectPill()
87    {
88        if (pacman.getTile(pacman.pacmanX, pacman.pacmanY) == 0) {
89            frighten = true;
90            frighten_time = gameTick;
91            for (int i=0; i< 4; i++)
92            {
93                if (ghosts[i].ghostState == SCATTER) {
94                    ghosts[i].eaten = false;
95                    ghosts[i].ghostState = FRIGHTEN;
96                    if (ghosts[i].ghostCurrentDirection == LEFT) { // Reverse their direction
97                        ghosts[i].ghostCurrentDirection = RIGHT;
98                    }
99                    else if (ghosts[i].ghostCurrentDirection == UP) {
100                       ghosts[i].ghostCurrentDirection = DOWN;
101                   }
102                   else if (ghosts[i].ghostCurrentDirection == RIGHT) {
103                       ghosts[i].ghostCurrentDirection = LEFT;
104                   }
105                   else {
106                       ghosts[i].ghostCurrentDirection = UP;
107                   }
108               }
109           }
110       }
111   }
```

```
117  void idle(){
118      if (!paused) {
119          switch (current_game_state) {
120          case BEGIN:
121              //after 2.5 seconds the game mode is set to play mode
122              if (gameTick >= 3850)
123                  current_game_state = PLAY;
124              break;
125          case PLAY:
126              if (pills == 0) {
127                  //reset all the pills in the maze, the pacman is set in the middle
128                  pills = 244;
129                  resetMaze();
130              }
131              detectPill();
132              pacman.checkTile();
133              detectGhost();
134              pacman.move();
135              detectGhost();
136              for (int i = 0; i < 4; ++i) {
137                  ghosts[i].move(ghosts[0]);
138              }
139              detectGhost();
140              if (frighten) {
141                  if (gameTick - frighten_time == 5000) {
142                      frighten = false;
143                      for (int i = 0; i < 4; i++){
144                          if (ghosts[i].ghostState == FRIGHTEN) {
145                              ghosts[i].eaten = false;
146                              ghosts[i].ghostState = SCATTER;
147                          }
148                      }
149                  }
150              }
151              break;
152          case DIE:
153              cout << "Pacman Died" << endl;
154              gameTick = 0;
155              --total_lives;
156              if (total_lives == 0) {
157                  current_game_state = OVER;
158                  break;
159              }
160              //Reset the coordinates of the pacman to the centre again
161              pacman.pacmanX = 13.5f;
162              pacman.pacmanY = 7.0f;
163              pacman.currentDirection = LEFT;
164              pacman.directionStore = LEFT;
165              current_game_state = BEGIN;
166              break;
167          }
168          gameTick++;
169          glutPostRedisplay();
170      }
171  }
```

```
173  void special(int key, int , int) {
174      switch (key)
175      {
176      case GLUT_KEY_LEFT:
177          pacman.setDirectionStore(LEFT);
178          break;
179      case GLUT_KEY_RIGHT:
180          pacman.setDirectionStore(RIGHT);
181          break;
182      case GLUT_KEY_UP:
183          pacman.setDirectionStore(UP);
184          break;
185      case GLUT_KEY_DOWN:
186          pacman.setDirectionStore(DOWN);
187          break;
188      }
189  }
190  void resetGame() {
191      score = 0;
192      total_lives = 3;
193      gameTick = 0;
194      pills = 244;
195      frighten = false;
196      frighten_time = 0;
197      //Reset the position of all ghosts
198      for (int i = 0; i < 4; i++)
199      {
200          ghosts[i].resetAllGhosts();
201      }
202      //Reset the pacman
203      pacman.pacmanX = 13.5f;
204      pacman.pacmanY = 7.0f;
205      pacman.currentDirection = LEFT;
206      pacman.directionStore = LEFT;
207      current_game_state = BEGIN;
208  }
```

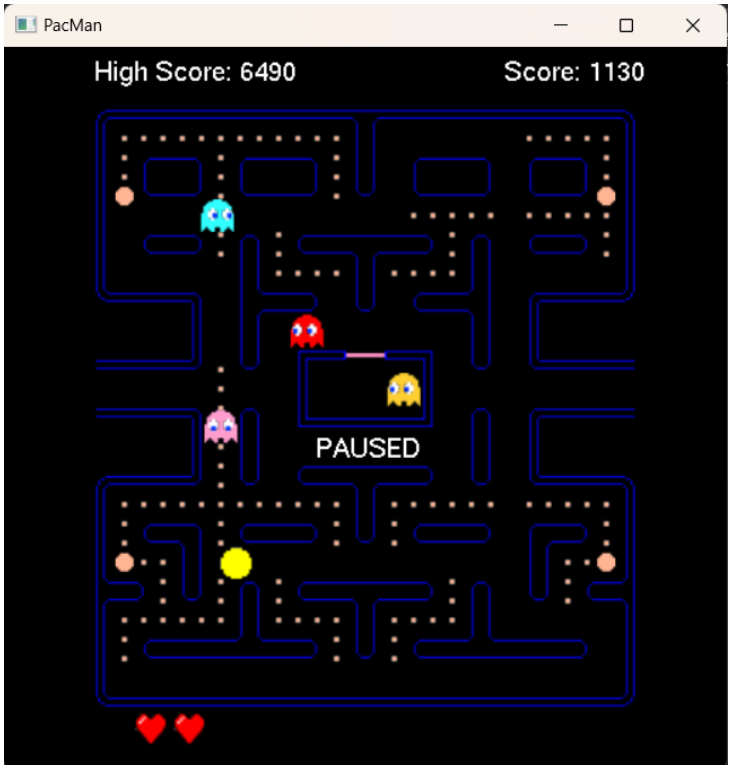
```
209  void keyboard(unsigned char key, int, int) {
210      switch (tolower(key)) {
211      case 'q':
212          exit(1);
213      case 'p':
214          if (paused)
215              paused = false;
216          else
217              paused = true;
218          break;
219      case 'r':
220          //To resest the game
221          if (current_game_state == OVER) {
222              resetMaze();
223              resetGame();
224          }
225          break;
226      case ' ':
227          //To start the game when the game is first opened
228          if (current_game_state == START) {
229              paused = false;
230              current_game_state = BEGIN;
231          }
232          break;
233      default:
234          break;
235      }
236      glutPostRedisplay();
237  }
238
239  void init() {
240      glMatrixMode(GL_PROJECTION);
241      glLoadIdentity();
242      gluOrtho2D(0, 300, 0, 300);
243      loadandbindallTextures();
244  }
```

```
246  void display() {
247      glClear(GL_COLOR_BUFFER_BIT);
248
249      switch (current_game_state)
250      {
251      case START:
252          drawStartScreen();
253          drawStart();
254          break;
255      case BEGIN:
256          drawMaze();
257          //Drawing the pacman
258          pacman.draw();
259          //Drawing the ghosts
260          for (int i = 0; i < 4; ++i) {
261              ghosts[i].draw();
262          }
263          drawInterface();
264          draw_Ready();
265          break;
266      case PLAY:
267          drawMaze();
268          pacman.draw();
269          for (int i = 0; i < 4; ++i) {
270              ghosts[i].draw();
271          }
272          drawInterface();
273          draw_Ready();
274          break;
275      case DIE:
276          drawMaze();
277          pacman.draw();
278          drawInterface();
279          break;
280      case OVER:
281          setHighScore(score);
282          drawMaze();
283          drawInterface();
284          drawGameOver();
285          drawPressToPlay();
286          break;
287      }
288
289      glutSwapBuffers();
290  }
291
```

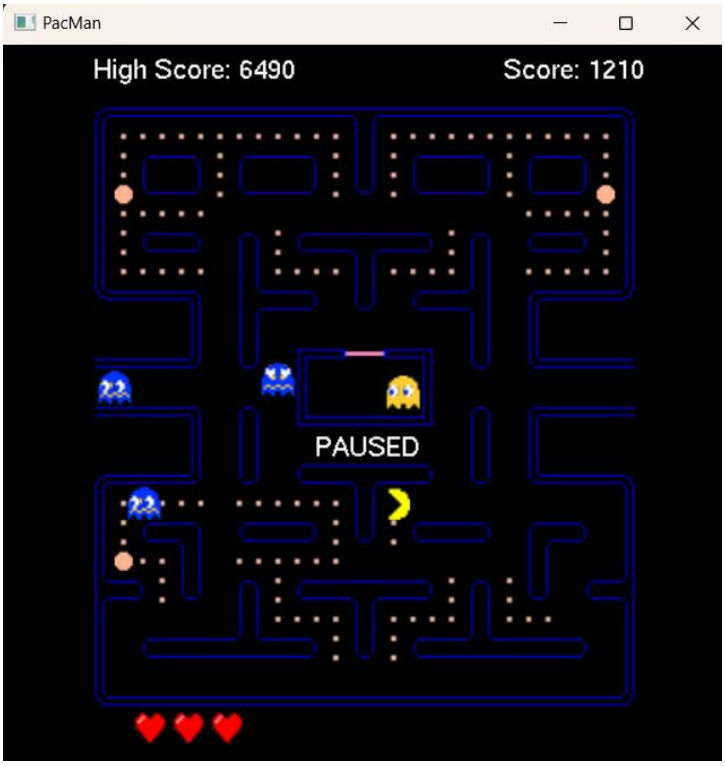
```
294  int main(int argc, char** argv) {
295      glutInit(&argc, argv);
296      glutInitDisplayMode(GLUT_DOUBLE| GLUT_RGB); // Single buffer and RGB color mode
297
298      int screenWidth = glutGet(GLUT_SCREEN_WIDTH);
299      int screenHeight = glutGet(GLUT_SCREEN_HEIGHT);
300      int windowWidth = 512; // Set your desired window width
301      int windowHeight = 512; // Set your desired window height
302      int windowPosX = (screenWidth - windowWidth) / 2;
303      int windowPosY = (screenHeight - windowHeight) / 2;
304      glutInitWindowPosition(windowPosX, windowPosY);
305
306      glutInitWindowSize(512, 512); // Set window size
307      glutCreateWindow("PacMan"); // Create a window with the given title
308
309
310      // Register display
311      glutDisplayFunc(display);
312
313      //To change the game state
314      glutKeyboardFunc(keyboard);
315      //For movement we will use the special Function
316      glutSpecialFunc(special);
317
318      //For performing the background processing
319      glutIdleFunc(idle);
320
321      init();
322      glutMainLoop(); // Enter the GLUT event processing loop
323
324      return 0;
325  }
326
```

Output/Screenshots

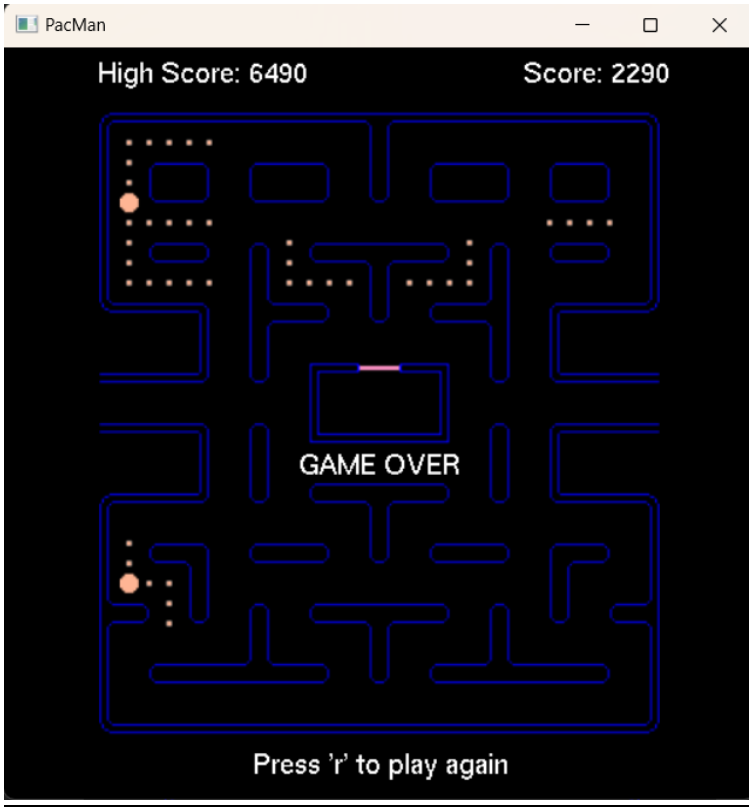
Start Page



Mid-Game Phase



Pacman-eaten super pill



Game Over Screen