Faculty of Computers & Artificial Intelligence

Benha University

Computer Graphic Project

3D Game

PAC_MAN GAME

by

Hazem Refai Mohammed Refai

Khaled Ahmed Slama Eldifrawy.

Ibrahim Mousa El_Sayed Habib.

Eslam Khaled Soliman Gad.

Ahmed Samer Eid Abd Elhamed.

1. Abstract about the game

This game is about an object called **Pac-Man** that you control to collect gems across the entire map, but be careful, these gems are protected by some monsters, your role is to collect these gems without any of these monsters catch you or you will die.

2. The components [objects] of the game

✓ There are 3 main creatures in this game

- **Pac-Man Object**

Figure[1]

It is the object that you control with the arrows of the keyboard that moves on the x and z axes back and forth.
Figure[1] shows what it looks like

- **Monster Object**

Figure[2]

It is the object that protects the gems to be collected and it moves automatically in specific paths and they are three monsters, each of them protects two gems.
Figure[2] shows what it looks like.

- **Gem**

Figure[3]

The crystals to be collected are 8 in this level
Figure[3] shows what it looks like.

✓ The map on which all these objects are located

It is a group of boxes arranged in a specific order to create specific paths in which each object moves

Figure[4]

3. How does each object move?

✓ . Pac_Man

To move this object, we use the up, down, right and left arrows on the keyboard [special keys]

- **Up arrow :** It is used to move the object on the Z axis inside and rotates the object at an angle of 180 to make the object face in the direction of its movement on the axis

```
if (key == GLUT_KEY_UP) {
    if (chanal != 1) std::swap(option1, option2);
    if (position_x >= 0 && position_x < 17 && position_z >= 0 && position_z < 17 && !map_arr[position_z - 1][position_x])
    {
        p_angel = 180;
        Pmove_z -= 3;
        p_tempz -= 3;
         //eyez -= 0.5;
        plat_z += 0.5;
         //upz -= 0.5;
```

- **Down arrow :** It is used to move the object on the Z axis outside and rotates the object at an angle of 0 to make the object face in the direction of its movement on the axis

```
if (key == GLUT_KEY_DOWN) {
    if (chanal != 1) std::swap(option1, option2);
    if (position_x >= 0 && position_x < 17 && position_z >= 0 && position_z < 17 && !map_arr[position_z + 1][position_x])
    {
        p_angel = 0;
        Pmove_z += 3;
        p_tempz += 3;
        //eyez += 0.5;
        plat_z -= 0.5;
       // upz += 0.5;
```

- **Right arrow :** It is used to move the object on the X axis (positive) and rotates the object at an angle of 90 to make the object face in the direction of its movement on the axis

```
if (key == GLUT_KEY_RIGHT) {
    if (position_x >= 0 && position_x < 17 && position_z >= 0 && position_z < 17 && !map_arr[position_z][position_x + 1])
    {
        p_angel = 90;
        Pmove_x += 3;
        p_tempx += 3;
        plat_x -= 0.5;
        //eyex += 0.5;
        //upx += 0.5;
```

- **Left arrow :** It is used to move the object on the X axis (negative) and rotates the object at an angle of 270 to make the object face in the direction of its movement on the axis

```
if (key == GLUT_KEY_LEFT) {
    if (position_x >= 0 && position_x < 17 && position_z >= 0 && position_z < 17 && !map_arr[position_z][position_x - 1])
    {
        p_angel = 270;
        Pmove_x -= 3;
        p_tempx -= 3;
        plat_x += 0.5;
        //eyex -= 0.5;
        //upx -= 0.5;
```

The object does not move if it collides with one of the map boxes, as it travels in the paths devoid of boxes [Collision].

✓ . **Monsters**

We do not control this object, but it moves automatically in a specific path, two of them move back and forth on the Z axis, and the third moves on the X axis.

The direction and angle of flow of this object changes by 180 degrees to move in the opposite direction if it collides with the core or one of the map boxes that obstruct its path.

```
if (e_flag == 1)
{
    e1_angel = 0;
    if (temp + 3 <= -8)
    {
        if (temp + 3 == -10.5)
        {
            if (map_dima[5] == 0)
            {
                move_z = move_z + 1;
                temp = temp + 1;
            }
            else
                e_flag = 0;
        }
        else
        {
            move_z = move_z + 1;
            temp = temp + 1;
        }
    }
    else
        e_flag = 0;
    Distance = sqrtf((temp - p_tempz) * (temp - p_tempz));
    if (Distance <= 3 && p_tempx == e_mainx)
    {
        std::cout << Distance << std::endl;
        initial_value();
        life--;

    }
}
//-----------
```

The code for the monster to flow in one direction

```
if (e_flag == 0)
{
    e1_angel = 180;
    if (temp - 3 >= -43)
    {
        if (temp - 3 == -40.5)
        {
            if (map_dima[3] == 0)
            {
                move_z = move_z - 1;
                temp = temp - 1;
            }
            else
                e_flag = 1;
        }
        else
        {
            move_z = move_z - 1;
            temp = temp - 1;
        }
    }
    else
        e_flag = 1;
    Distance = sqrtf((temp - p_tempz) * (temp - p_tempz));
    if (Distance <= 3 && p_tempx == e_mainx)
    {
        std::cout << Distance << std::endl;
        initial_value();
        life--;

    }
}
```

The code for the monster to flow in oppisite direction

✓ **Gems**

The gem is placed at a specific position and rotates around its y axis.

```
angle1 += 10;
```

4. Game roles [How to play ?]

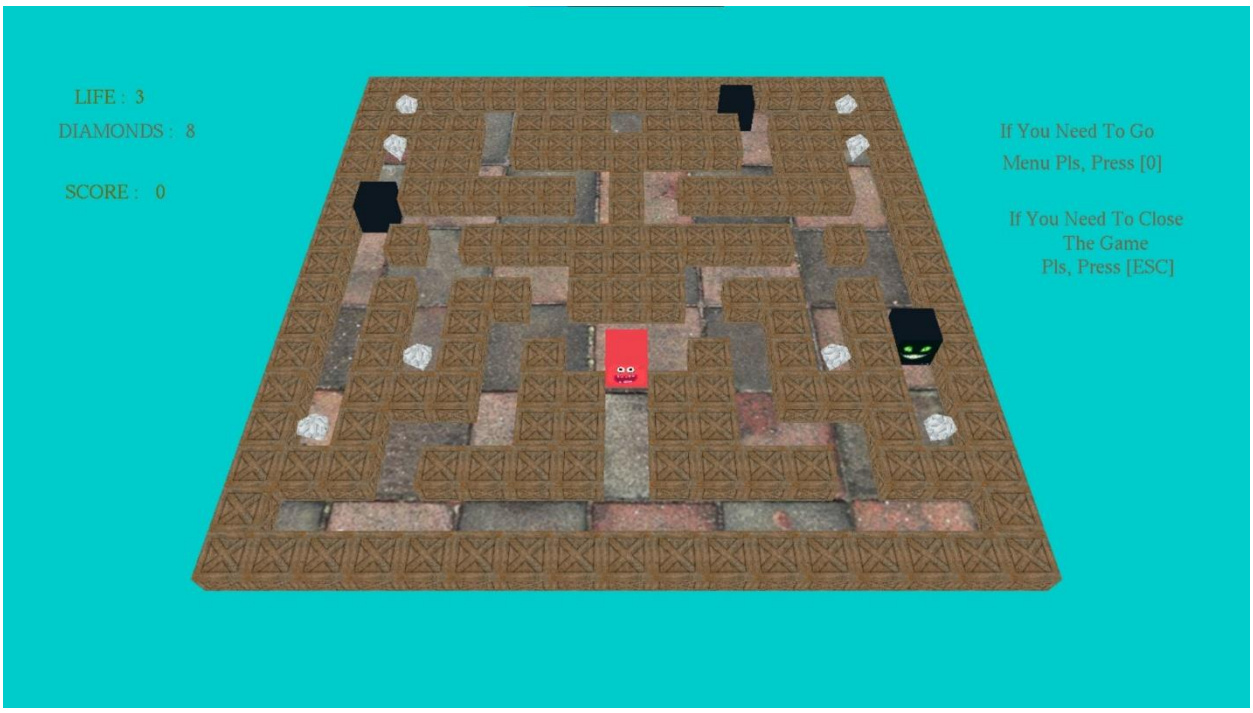
When the game starts, the first channel (cover();) is activated and contains a welcome message and two choices:

- Start game
- Close

When you click on Close, the game will be closed.

And when you press Start the game, the second channel (drow();) will be activated, which contains the game, and you will find it on the left

- lifes
- The number of gems left
- the score



.....

Note: You can switch between the options using the Up and Down .

It also contains two notes on the right

- If you want to return to the start page (menu) press [0] Button.
- If you want to close the game, press [ESC] Button.

As we mentioned earlier, you control the Pac-Man with the keyboard arrows. You have to follow the following rules to win this game :-

Role$_1$: You have to avoid collision with monsters.

You have 3 attempts every time you collide with a monster, it decreases , You return to the starting point to start a new attempt, but you do not lose the gems you collected.

Role$_2$: Move in the empty paths of the boxes, otherwise you won't be able to move.

The Pac-Man moves by 3 in each step, and before taking this step, he checks the collision condition. He will not make the move if there is a box in front of him.

Role$_3$: You have to collect all the gems.

In order to collect the gem, you must go to the same place where the gem is.

Each gem you collect increases your score in proportion to how difficult it is to reach the gem's postion.

5. When will i win

You win when you collect all the gems on the map. Before you lose all your lifes

If you do this, the third channel (win_screen) will be activated and contain a **congratulatory** message and two choices

- Play again
- Close

In the case of playing again, the game will be started from the beginning again and played and so on

6. When will i lose

You will lose when you have exhausted all your lifes.

Lifes = 0;

If you do this, the third channel (lose_screen) will be activated and contain a **Oops!** message and two choices

- Play again
- Close

In the case of playing again, the game will be started from the beginning again and played and so on.

7. Some of the functions used

✓ **Timer Function**

It is a function used to periodically repaint the scene after a specified period of time

It was used to activate the automatic movement of

- Monsters
- Gems

```
//---------- Timer Funcation
void timer(int value) {
    glutPostRedisplay();
    if (angle2 + 1 == 360) { ... }
    else
        angle2 += 1;

    if (angle1 + 10 == 360) { ... }
    else
        angle1 += 10;
    //-------------------------------- movment of right Side
    if (e_flag == 1) { ... }
    //-----------

    if (e_flag == 0) { ... }

    //-------------------------------- movment of left Side
    if (e2_flag == 1) { ... }
    //-----------

    if (e2_flag == 0) { ... }

    //-------------------------------- movment of top Side
    if (e3_flag == 1) { ... }
    //-----------

    if (e3_flag == 0) { ... }
    if (life == 0)
        chanal = 2;

    glutTimerFunc(1000 / 60, timer, 0);

}
//----------SpecialKeyboard Funcation
```

✓ **Reshape Function**

Used to reset the assigned drawing area if the display frame area changes

```
//----------- ReShape Funcation
void reshape(GLsizei width, GLsizei height) {  // GLsizei for non-negative integer
   // Compute aspect ratio of the new window
    if (height == 0) height = 1;                // To prevent divide by 0
    GLfloat aspect = (GLfloat)width / (GLfloat)height;

    // Set the viewport to cover the new window
    glViewport(0, 0, width, height);

    // Set the aspect ratio of the clipping volume to match the viewport
    glMatrixMode(GL_PROJECTION);  // To operate on the Projection matrix
    glLoadIdentity();             // Reset
    // Enable perspective projection with fovy, aspect, zNear and zFar
    gluPerspective(50.0f, aspect, 0.1f, 500.0f);
    glMatrixMode(GL_MODELVIEW);

}
//----------- Main Funcation
```

8. Events that have been handled

✓ Keyboard Events

- F1 : Used to activate and deactivate full screen mode.
- Up,Down,Left,Right Arrows : It is used to move the Pac-Man.
- 0 : Used to return to the start page (menu).
- Enter : used to select the choice.
- ESC : Used to exit the game.

```
//-----------SpecialKeyboard Funcation
void Keyboard(unsigned char key, int x, int y) {
    if (key == 13)
    {
        if(chanal!= 1 )
            score = 0;
        //std::cout << "enter pressing " << std::endl;
        if (option1 > option2)
        {
            chanal = 1;

        }
        else
            exit(0);
    }
    if (key == 27)
        exit(0);
    if (key == 48)
        chanal = - 1;

}
```

```
void specialKeyboard(int key, int x, int y) {
    if (key == GLUT_KEY_F1) {
        fullScreen = !fullScreen;
        if (fullScreen)
            glutFullScreen();
        else {
            glutPositionWindow(0,25);
            glutReshapeWindow(width_screen, height_screen-66);
        }
```

- ✓ Mouse Events
  - ➢ Left button : Rotate the game map by 10 degrees counterclockwise.
  - ➢ Right button : Rotate the game map by 10 degrees clockwise.

```
void mouse(int button, int state, int x, int y)
{

    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
    {
        flag = 0;
        angle += 1;
    }
    else if (button == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
    {
        flag = 1;
        angle -= 1;

    }

}
//------------ Load Funcation
```

……