Computer graphics and multimedia
UDL MINF 20-21

Package 3

Danillo Lange Souza Borges dos Santos

Bugs fixed: fixed pacman and ghosts movimentation
Ghost's movement are now independent of the player

Task 1. Inclusion of 3D graphics to the game

Considerations:

- The game now are rendered in 3D, the maze cells are 3D Squares with textures and the player, ghosts and food are Spheres.
- The Sphere code is located in Sphere.cpp, in the main we just inicialize and control the figures.
- You can rotate the maze with the keys 'i' 'j' 'k' 'l' and press 'q' to reset it to the inicial position
- You can zoom-in with 'x' and zoom-out with 'z' key, press 'c' to reset zoom values to the default ones.
- The code for the maze cell is in displayMaze() function

```
void displayMaze() {

    glClearColor(0.0,0.0,0.0,0.0);
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);

    for (int x = 0; x < maze.maze_width; x++) {
        for (int y = 0; y < maze.maze_heigth; y++) {
            if (maze.maze_grid[maze.translateXY(x,y)] == '#' ||
maze.maze_grid[maze.translateXY(x,y)] == '1') {

                /*
                P4 <-- P3
                       ^
                       |
                P1 --> P2
```

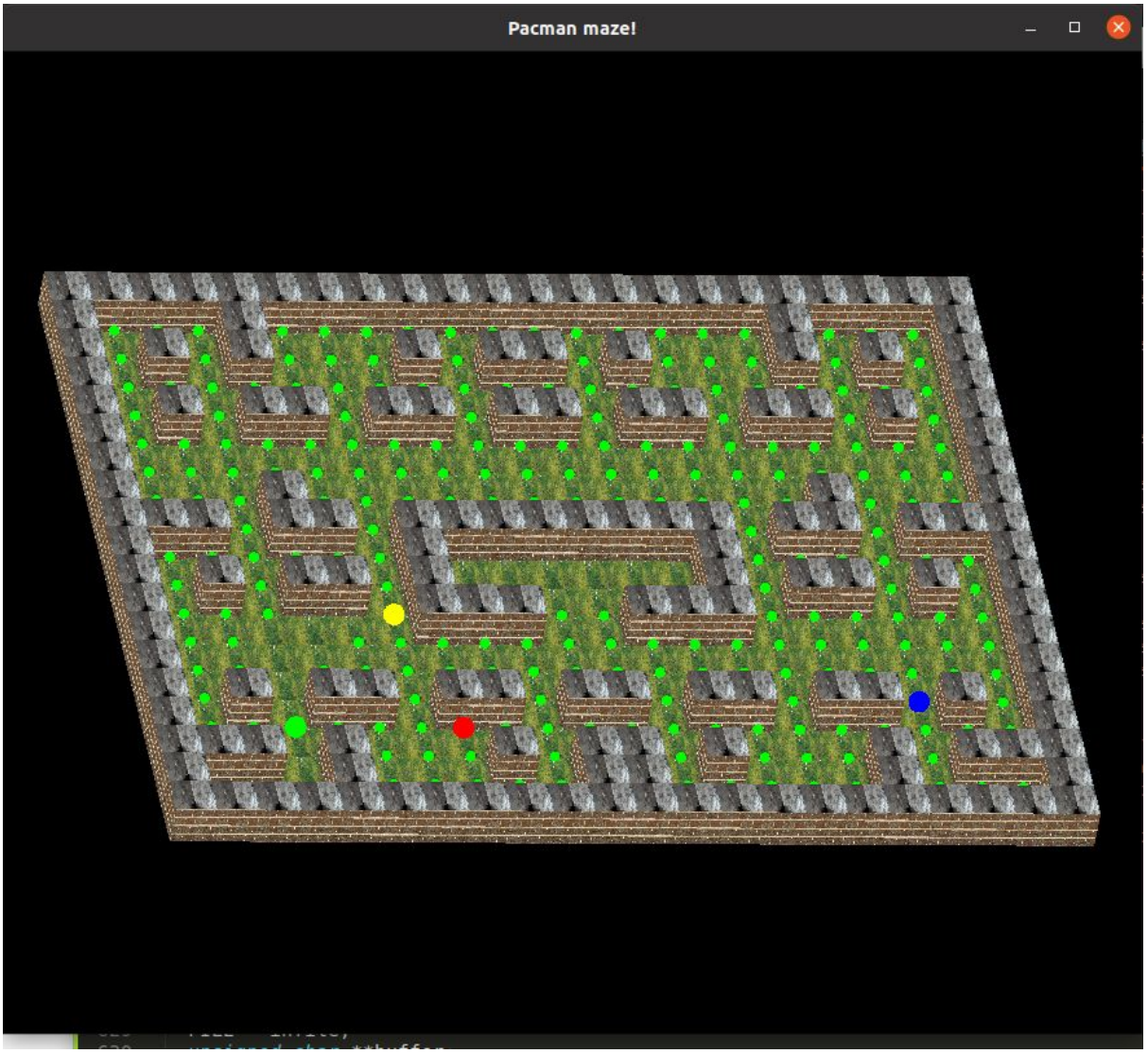
```
                */


                //glColor3f(0.0,0.0,255.0); // Blue
                glEnable(GL_TEXTURE_2D);
                glBindTexture(GL_TEXTURE_2D,2);
                glBegin(GL_QUADS);
                    glTexCoord2f(-0.25,0.0); glVertex3i(x * (cell_width), y *
(cell_width),cell_width); // P1 bottom left
                    glTexCoord2f(0.25,0.0); glVertex3i(x * (cell_width) + cell_width, y *
(cell_width), cell_width); // P2 bottom right
                    glTexCoord2f(0.25,0.25); glVertex3i(x * (cell_width) + cell_width, y *
(cell_width) + cell_width, cell_width); // P3 top right
                    glTexCoord2f(-0.25,0.25); glVertex3i(x * (cell_width), y * (cell_width) +
cell_width, cell_width); // P4 top left
                glEnd();
                glDisable(GL_TEXTURE_2D);

                //glColor3f(255.0,0.0,0.0); // Red
                glEnable(GL_TEXTURE_2D);
                glBindTexture(GL_TEXTURE_2D,0);
                glBegin(GL_QUADS);
                    glTexCoord2f(-1.0,0.0); glVertex3i(x * (cell_width), y * (cell_width),0); //
P1 bottom left
                    glTexCoord2f(1.0,0.0); glVertex3i(x * (cell_width) + cell_width, y *
(cell_width), 0); // P2 bottom right
                    glTexCoord2f(1.0,1.0); glVertex3i(x * (cell_width) + cell_width, y *
(cell_width) + cell_width, 0); // P3 top right
                    glTexCoord2f(-1.0,1.0); glVertex3i(x * (cell_width), y * (cell_width) +
cell_width, 0); // P4 top left
                glEnd();
                glDisable(GL_TEXTURE_2D);

                glEnable(GL_TEXTURE_2D);
                glBindTexture(GL_TEXTURE_2D,0);
                //glColor3f(0.0,255.0,0.0); // Green
                glBegin(GL_QUADS);
                    glTexCoord2f(-1.0,0.0);  glVertex3i(x * (cell_width) + cell_width, y *
(cell_width), 0); // P1
                    glTexCoord2f(1.0,0.0); glVertex3i(x * (cell_width) + cell_width, y *
(cell_width) + cell_width, 0); // P2
                    glTexCoord2f(1.0,1.0); glVertex3i(x * (cell_width) + cell_width, y *
(cell_width) + cell_width, cell_width); // P3
                    glTexCoord2f(-1.0,1.0); glVertex3i(x * (cell_width) + cell_width, y *
(cell_width), cell_width); // P4
                glEnd();
                glDisable(GL_TEXTURE_2D);

                glEnable(GL_TEXTURE_2D);
                glBindTexture(GL_TEXTURE_2D,0);
                //glColor3f(255.0,255.0,0.0); // Yellow
                glBegin(GL_QUADS);
                    glTexCoord2f(-1.0,0.0); glVertex3i(x * (cell_width), y * (cell_width),0); //
P1
                    glTexCoord2f(1.0,0.0); glVertex3i(x * (cell_width) + cell_width, y *
(cell_width), 0); // P2
                    glTexCoord2f(1.0,1.0); glVertex3i(x * (cell_width) + cell_width, y *
(cell_width), cell_width); // P3
                    glTexCoord2f(-1.0,1.0); glVertex3i(x * (cell_width), y *
```

```
(cell_width),cell_width); // P4
                glEnd();
                glDisable(GL_TEXTURE_2D);

                glEnable(GL_TEXTURE_2D);
                glBindTexture(GL_TEXTURE_2D,0);
                //glColor3f(255.0,0.0,255.0); // Pink
                glBegin(GL_QUADS);
                    glTexCoord2f(-1.0,0.0); glVertex3i(x * (cell_width), y * (cell_width) +
cell_width, 0); // P1
                    glTexCoord2f(1.0,0.0); glVertex3i(x * (cell_width), y * (cell_width),0); //
P2
                    glTexCoord2f(1.0,1.0); glVertex3i(x * (cell_width), y *
(cell_width),cell_width); // P3
                    glTexCoord2f(-1.0,1.0); glVertex3i(x * (cell_width), y * (cell_width) +
cell_width, cell_width); // P4
                glEnd();
                glDisable(GL_TEXTURE_2D);

                glEnable(GL_TEXTURE_2D);
                glBindTexture(GL_TEXTURE_2D,0);
                //glColor3f(255.0,0.0,255.0); // Pink
                glBegin(GL_QUADS);
                    glTexCoord2f(1.0,0.0); glVertex3i(x * (cell_width) + cell_width, y *
(cell_width) + cell_width, 0); // P1
                    glTexCoord2f(-1.0,0.0); glVertex3i(x * (cell_width), y * (cell_width) +
cell_width, 0); // P2
                    glTexCoord2f(-0.5,0.5); glVertex3i(x * (cell_width), y * (cell_width) +
cell_width, cell_width); // P3
                    glTexCoord2f(0.5,0.5); glVertex3i(x * (cell_width) + cell_width, y *
(cell_width) + cell_width, cell_width); // P4
                glEnd();
                glDisable(GL_TEXTURE_2D);

            }
            else if (maze.maze_grid[maze.translateXY(x,y)] == ' ' ||
maze.maze_grid[maze.translateXY(x,y)] == '0') {
                //glColor3f(0.0,0.0,0.0); // Black
                glBindTexture(GL_TEXTURE_2D,1);
                glEnable(GL_TEXTURE_2D);
                glBegin(GL_QUADS);
                    glTexCoord2f(-0.5,0.0); glVertex2i(x * (cell_width), y * (cell_width)); // P1
bottom left
                    glTexCoord2f(0.5,0.0); glVertex2i(x * (cell_width) + cell_width, y *
(cell_width)); // P2 bottom right
                    glTexCoord2f(0.5,0.5); glVertex2i(x * (cell_width) + cell_width, y *
(cell_width) + cell_width); // P3 top right
                    glTexCoord2f(-0.5,0.5); glVertex2i(x * (cell_width), y * (cell_width) +
cell_width); // P4 top left
                glEnd();
                glDisable(GL_TEXTURE_2D);
            }
            else if (maze.maze_grid[maze.translateXY(x,y)] == 'L')
                glColor3f(0.0,255.0,0.0); // Green

        }
    }
}
```

Task 2. Inclusion of texture mapping

Considerations:

- We are using 3 textures, one for the ceiling, one for the floor, and one for the walls, all files are included in the package
- All textures are properly scaled to low-resolution



Bugs to be fixed:

- The program is not yet prepared for all sizes of maze, if the values are too high it may distort the graphics a little, please use 22x19 for optimal results.