Computer graphics and multimedia
UDL MINF 20-21

Universitat de Lleida

Package 2

Danillo Lange Souza Borges dos Santos

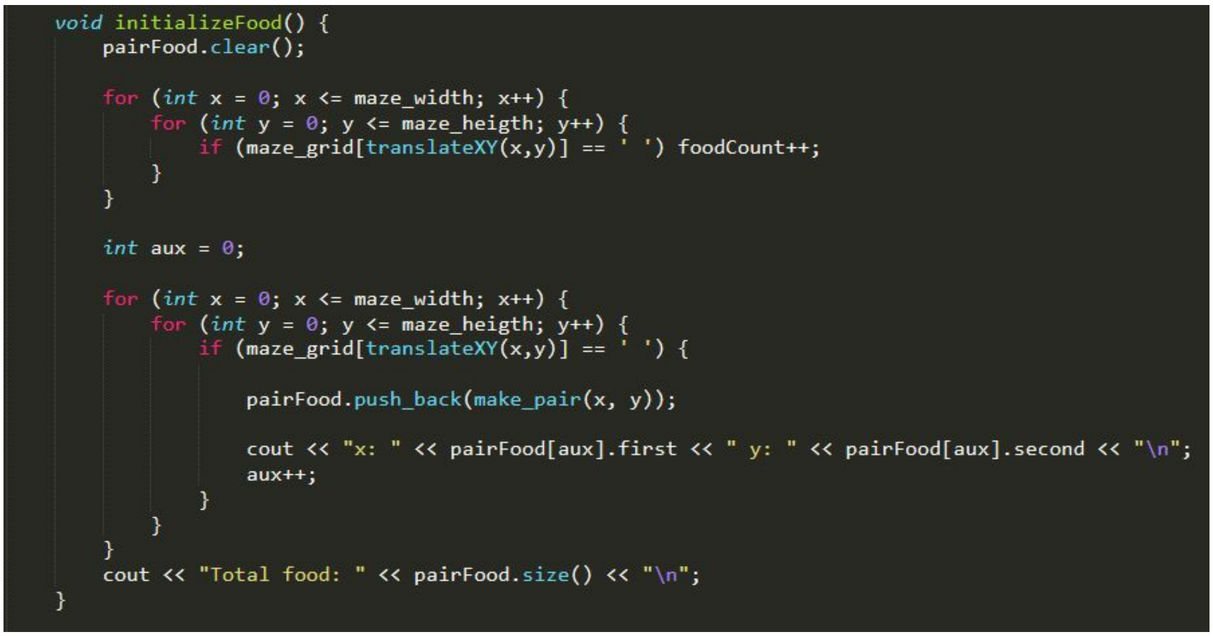
Bugs fixed: removed dead-ends from the maze

Task 1. Inclusion of food elements to the map

Considerations:

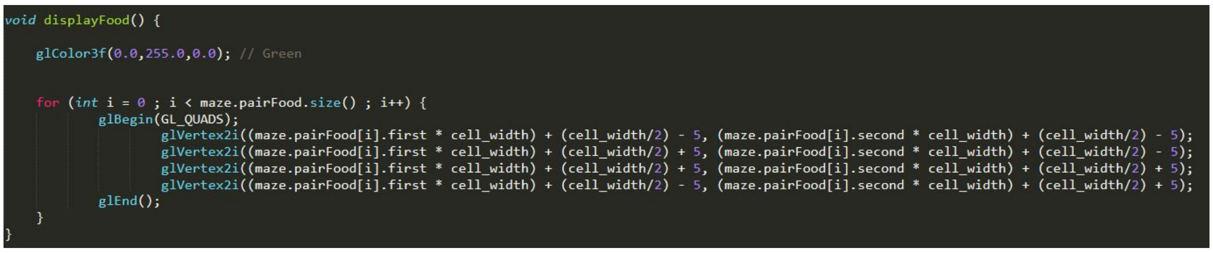
- The food meet all the criterias, when you eat one, your "score" goes up and the fruit disappear, the score is shown on the screen (just for debug reasons, will be changed later)

  The food declaration and initialization are on Maze.cpp

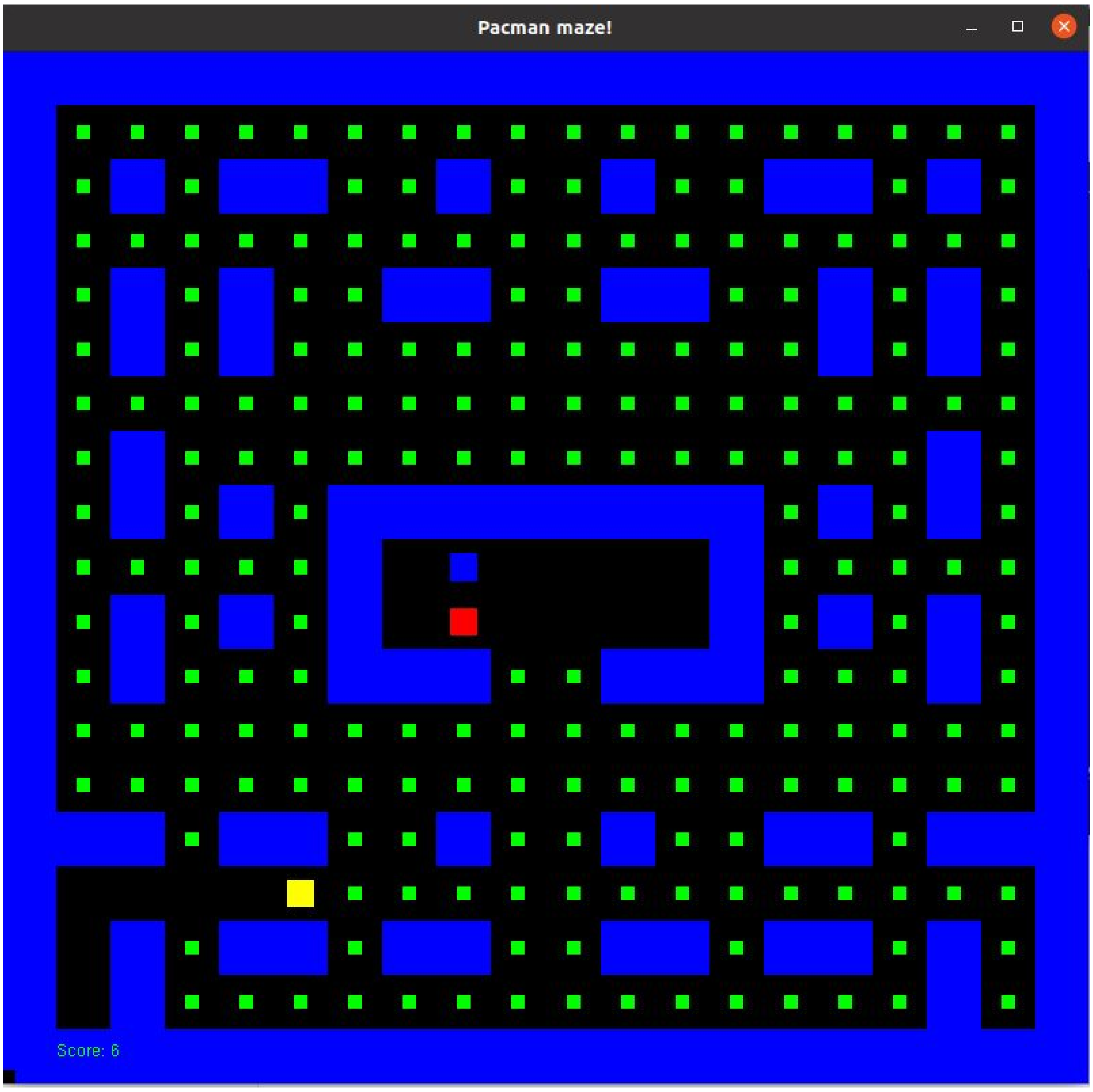


```
void initializeFood() {
    pairFood.clear();

    for (int x = 0; x <= maze_width; x++) {
        for (int y = 0; y <= maze_heigth; y++) {
            if (maze_grid[translateXY(x,y)] == ' ') foodCount++;
        }
    }

    int aux = 0;

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

                pairFood.push_back(make_pair(x, y));

                cout << "x: " << pairFood[aux].first << " y: " << pairFood[aux].second << "\n";
                aux++;
            }
        }
    }
    cout << "Total food: " << pairFood.size() << "\n";
}
```

main.cpp Printing



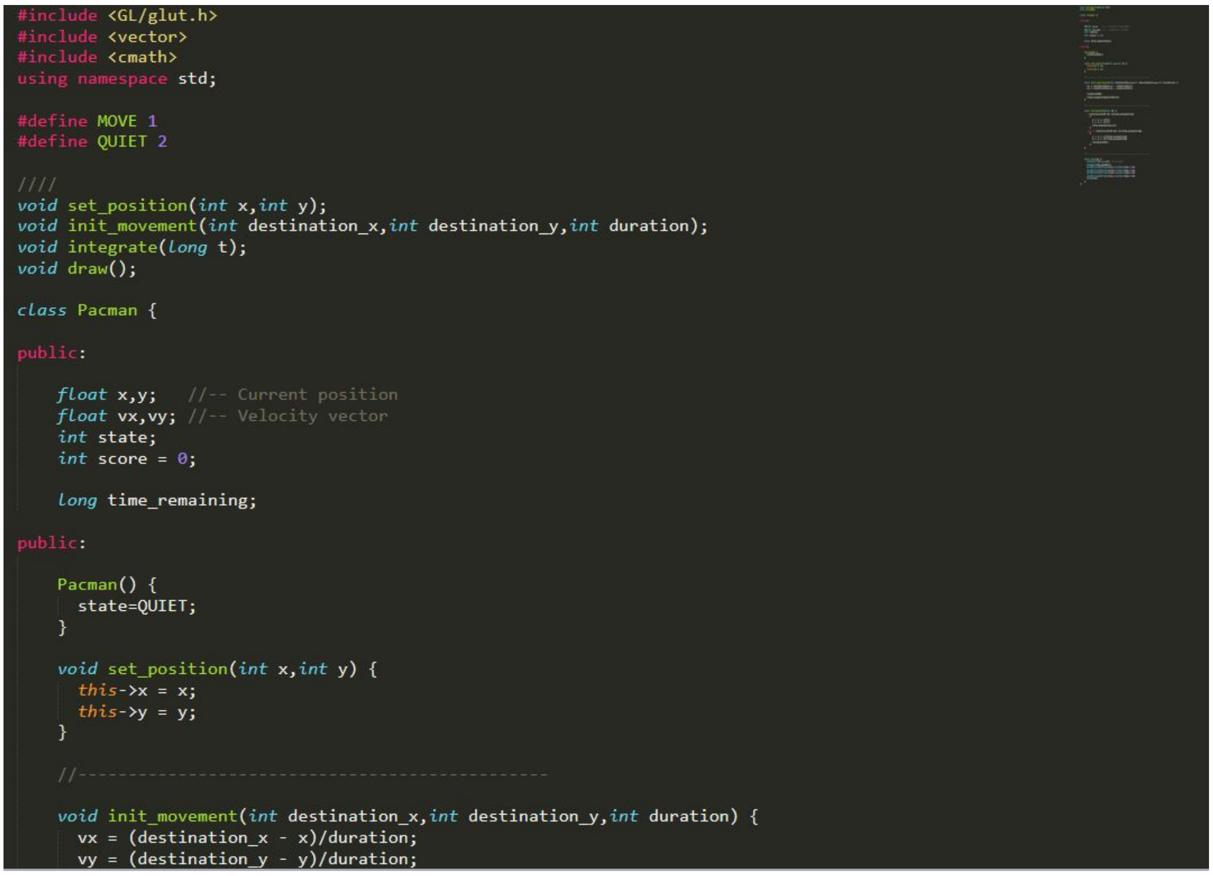
```
void displayFood() {

    glColor3f(0.0,255.0,0.0); // Green


    for (int i = 0 ; i < maze.pairFood.size() ; i++) {
            glBegin(GL_QUADS);
                    glVertex2i((maze.pairFood[i].first * cell_width) + (cell_width/2) - 5, (maze.pairFood[i].second * cell_width) + (cell_width/2) - 5);
                    glVertex2i((maze.pairFood[i].first * cell_width) + (cell_width/2) + 5, (maze.pairFood[i].second * cell_width) + (cell_width/2) - 5);
                    glVertex2i((maze.pairFood[i].first * cell_width) + (cell_width/2) + 5, (maze.pairFood[i].second * cell_width) + (cell_width/2) + 5);
                    glVertex2i((maze.pairFood[i].first * cell_width) + (cell_width/2) - 5, (maze.pairFood[i].second * cell_width) + (cell_width/2) + 5);
            glEnd();
    }
}
```



Task 2. Inclusion and animation of the main character

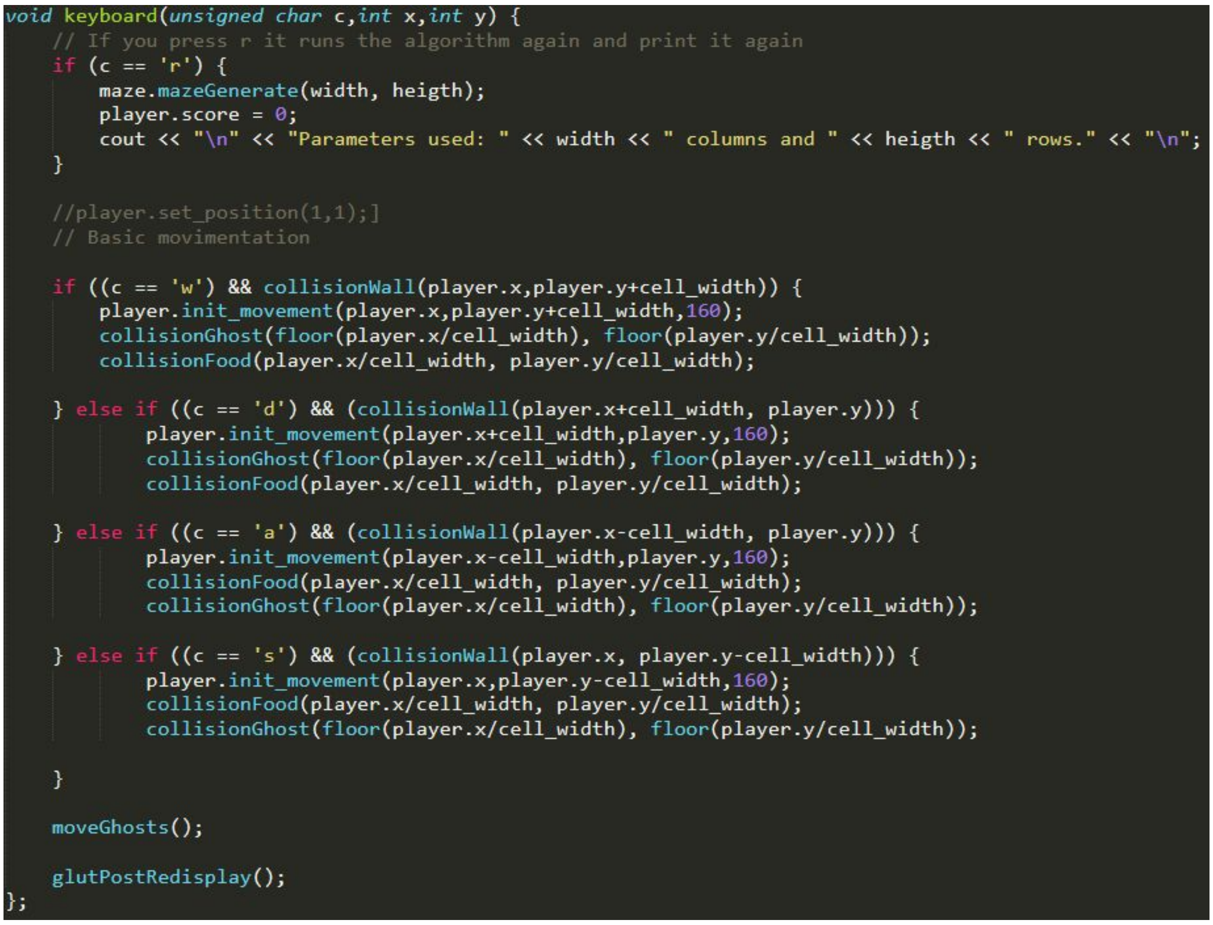
Considerations:

- In file Pacman.cpp are all the attributes and functions regarding the pacman movement and animation.
- You can choose pacmans position at the start of the game (if you begin in a wall dont worry, its a bug but you can walk out of it).
- When you press ‘W’,’A’,’S’,’D’ the pacman moves in the desired direction and check for collisions.



```
#include <GL/glut.h>
#include <vector>
#include <cmath>
using namespace std;

#define MOVE 1
#define QUIET 2

////
void set_position(int x,int y);
void init_movement(int destination_x,int destination_y,int duration);
void integrate(long t);
void draw();

class Pacman {

public:

    float x,y;   //-- Current position
    float vx,vy; //-- Velocity vector
    int state;
    int score = 0;

    long time_remaining;

public:

    Pacman() {
      state=QUIET;
    }

    void set_position(int x,int y) {
      this->x = x;
      this->y = y;
    }

    //----------------------------------------------

    void init_movement(int destination_x,int destination_y,int duration) {
      vx = (destination_x - x)/duration;
      vy = (destination_y - y)/duration;
```

main.cpp :



```cpp
void keyboard(unsigned char c,int x,int y) {
    // If you press r it runs the algorithm again and print it again
    if (c == 'r') {
        maze.mazeGenerate(width, heigth);
        player.score = 0;
        cout << "\n" << "Parameters used: " << width << " columns and " << heigth << " rows." << "\n";
    }

    //player.set_position(1,1);]
    // Basic movimentation

    if ((c == 'w') && collisionWall(player.x,player.y+cell_width)) {
        player.init_movement(player.x,player.y+cell_width,160);
        collisionGhost(floor(player.x/cell_width), floor(player.y/cell_width));
        collisionFood(player.x/cell_width, player.y/cell_width);

    } else if ((c == 'd') && (collisionWall(player.x+cell_width, player.y))) {
            player.init_movement(player.x+cell_width,player.y,160);
            collisionGhost(floor(player.x/cell_width), floor(player.y/cell_width));
            collisionFood(player.x/cell_width, player.y/cell_width);

    } else if ((c == 'a') && (collisionWall(player.x-cell_width, player.y))) {
            player.init_movement(player.x-cell_width,player.y,160);
            collisionFood(player.x/cell_width, player.y/cell_width);
            collisionGhost(floor(player.x/cell_width), floor(player.y/cell_width));

    } else if ((c == 's') && (collisionWall(player.x, player.y-cell_width))) {
            player.init_movement(player.x,player.y-cell_width,160);
            collisionFood(player.x/cell_width, player.y/cell_width);
            collisionGhost(floor(player.x/cell_width), floor(player.y/cell_width));

    }

    moveGhosts();

    glutPostRedisplay();
};
```

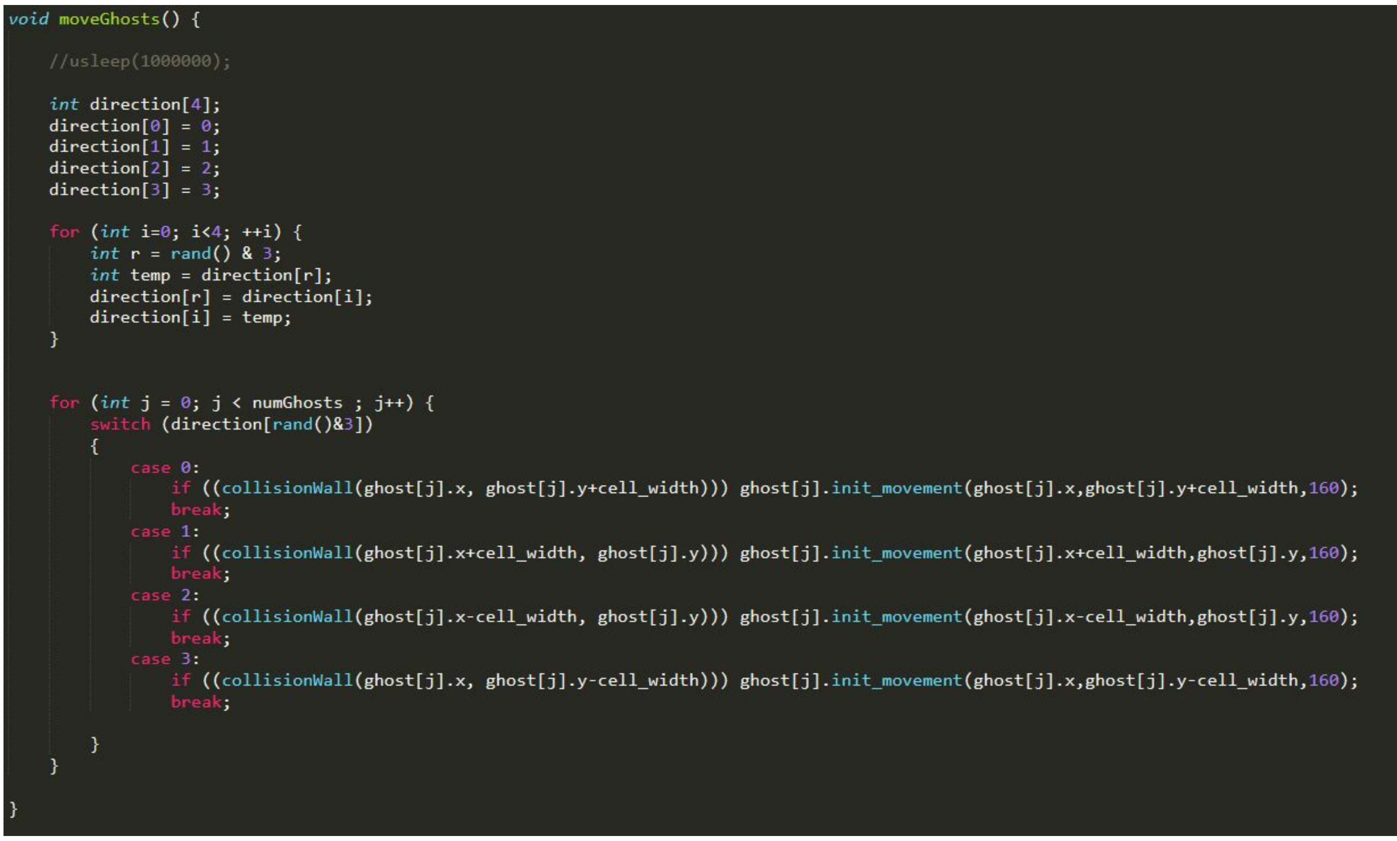
- The animation happens with a movement of cellWidth pixels (defined in the game by default 40) to the desired location.

Task 3. Inclusion and animation of enemy characters

Considerations:

- The enemies attributes and functions are located in Ghost.cpp
- The enemies were included, being represented as squares with 3 pre-defined colors, you can choose from 1-3 ghost at the start of the game.
- Their movement is random and independent from each other.
- When you are in the same position as one of the ghosts, “Game Over” is printed on the terminal, saying that you collided with one of them.
- However, their movement only happens when the player moves, due to difficulties during adding their behaviour to the idle function.
- They spawn at the center room.
- Their animation is the same as the main character.

main.cpp:



```cpp
void moveGhosts() {

    //usleep(1000000);

    int direction[4];
    direction[0] = 0;
    direction[1] = 1;
    direction[2] = 2;
    direction[3] = 3;

    for (int i=0; i<4; ++i) {
        int r = rand() & 3;
        int temp = direction[r];
        direction[r] = direction[i];
        direction[i] = temp;
    }


    for (int j = 0; j < numGhosts ; j++) {
        switch (direction[rand()&3])
        {
            case 0:
                if ((collisionWall(ghost[j].x, ghost[j].y+cell_width))) ghost[j].init_movement(ghost[j].x,ghost[j].y+cell_width,160);
                break;
            case 1:
                if ((collisionWall(ghost[j].x+cell_width, ghost[j].y))) ghost[j].init_movement(ghost[j].x+cell_width,ghost[j].y,160);
                break;
            case 2:
                if ((collisionWall(ghost[j].x-cell_width, ghost[j].y))) ghost[j].init_movement(ghost[j].x-cell_width,ghost[j].y,160);
                break;
            case 3:
                if ((collisionWall(ghost[j].x, ghost[j].y-cell_width))) ghost[j].init_movement(ghost[j].x,ghost[j].y-cell_width,160);
                break;

        }
    }

}
```

Problems to be fixed:

- Ghost's movement to be independent of the player
- Players movement to match the original game
- Exact collision between the player and the ghosts are still to be finished
- Starting position of player still is in progress