COMPUTER GRAPHICS AND INTERACTION SYSTEMS

Programming Assignment 1-A.

Team

Gkovaris Christos-Grigorios
Spanou Maria

Timetable of Assignment 1A

Version	Date	Progress
1.0	21/10/2024	Lab Exercise 1-A Announcement
2.0	24/10/2024	Implementation of questions (i), (ii) και (iii)
2.1	25/10/2024	Implementation of question (iv)
3.0	25/10/2024	Report preparation for the lab exercise

Feature Analysis

The assignment was implemented on a machine (Lenovo Ideapad 3), with the following specifications:

- AMD Ryzen 7 (7730U)
- 8 CPU cores
- 16 Threads
- Boost Clock up to 4.5GHz
- Base Clock 2.0GHz
- CPU Socket FP6
- Intergraded Graphics (Radeon Graphics)

Furthermore, the **assignment 1-A** was implemented on **Windows 11 Home**, using the **Visual Studio 2022 (x64)**.

Team Functionality / Query Analysis

The teamwork was exemplary, ensuring flawless implementation and optimal functionality for Exercise 1-A. Both members contributed significantly to the design of the algorithmic portion of the task. In question (i), the collaboration was complete, and we successfully achieved the desired result. In question (ii), member 5203 undertook the representation of the maze with the triangle coordinates on paper, while member 5351 created the shape_1_buffer[] array with the coordinates of the 330 triangles. In question (iii), we created character A based on the coordinate array, carefully positioning the coordinates to ensure it was centered within the grid and accurately sized. In question (iv), an initial issue we encountered was that using the glfwGetKey command registered multiple executions of key presses, even with a brief press. We identified this issue through debugging prints and resolved it by optimizing the algorithm we followed. Finally, in question (ii), there is a duplicate triangle that we were unable to locate, resulting in an increase of three vertices (a total of 333) instead of the expected 330.

2.0 Implementation of questions (i), (ii) και (iii)

For question (i):

In this question, we were asked to implement a program that opens a basic window of size 750x750 and changes the background color to black. Specifically, the following changes were made:

In the file **Source-1A.cpp**, we located the command

window = glfwCreateWindow(750, 750, u8"Άσκηση 1A - 2024", NULL, NULL);

which defines the size and title of the window (where the maze will later be displayed). We proceeded to change the first two arguments of the **glfwCreateWindow** function to 750, thus setting the window size to 750x750.

Additionally, we modified the third argument of the function, which corresponds to the window title, changing it to "Άσκηση 1A - 2024". Upon opening the window, we encountered an issue as Greek characters were not supported. To resolve this problem, we added the prefix "u8" before the desired window title.

Finally, regarding the background color, we located the command

glClearColor(0.0f, 0.0f, 0.0f, 0.0f);

of which we appropriately modified the arguments to correctly create the "Black Background."

For question (ii):

In this question, we were asked to design the maze as depicted in Image 1 of the prompt. Specifically, the following changes were made:

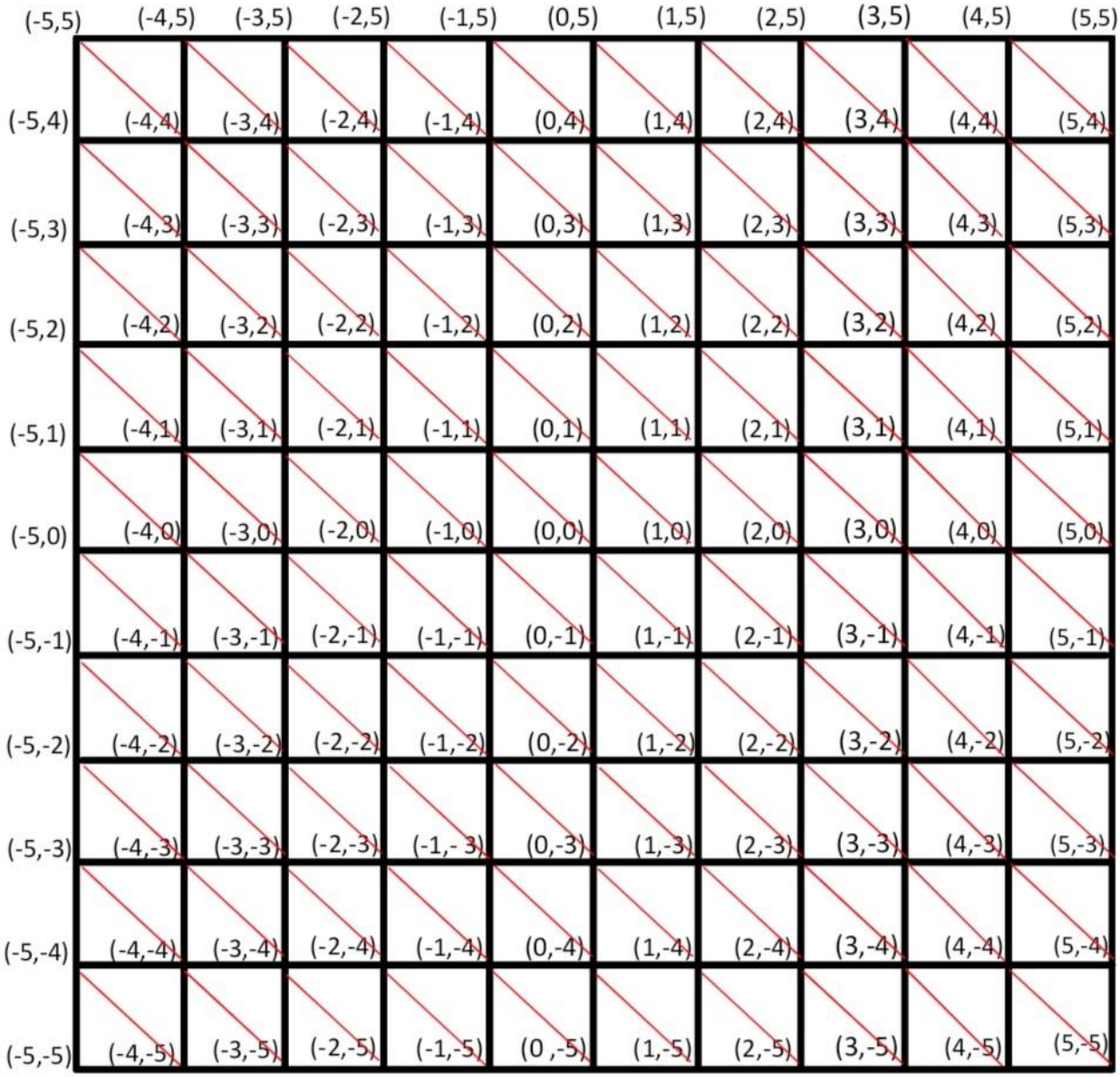
In the file **Source-1A.cpp**, we implemented the command

```
static const GLfloat shape_1_buffer[] = {  -5.0f , 5.0f,   -5.0f , 4.0f,   -4.0f , 4.0f,  -4.0f , 4.0f,   -4.0f , 5.0f,   -5.0f , 5.0f,  -
4.0f , 5.0f,   -4.0f , 4.0f,   -3.0f , 4.0f, -3.0f , 4.0f,   -4.0f , 5.0f,   -3.0f , 5.0f, -3.0f , 5.0f,   -3.0f , 4.0f,   -2.0f , 4.0f, -3.0f ,
5.0f,   -2.0f , 5.0f,   -2.0f , 4.0f, 2.0f , 5.0f,   -2.0f , 4.0f,   -1.0f , 4.0f, -2.0f , 5.0f,   -1.0f , 4.0f,   -1.0f , 5.0f, 1.0f , 5.0f,   -1.0f ,
4.0f,   0.0f , 4.0f, -1.0f , 5.0f,   0.0f , 4.0f,   0.0f , 5.0f, 0.0f , 4.0f,   0.0f , 5.0f,   1.0f , 4.0f, 1.0f , 5.0f,   1.0f , 4.0f,   0.0f ,
5.0f, 1.0f , 5.0f,   1.0f , 4.0f,   2.0f , 4.0f, 1.0f , 5.0f,   2.0f , 4.0f,   2.0f , 5.0f, 2.0f , 5.0f,   2.0f , 4.0f,   3.0f , 4.0f, 2.0f ,
5.0f,   3.0f , 5.0f,   3.0f , 4.0f, 3.0f , 5.0f,   3.0f , 4.0f,   4.0f , 4.0f, 3.0f , 5.0f,   4.0f , 4.0f,   4.0f , 5.0f, 4.0f , 5.0f,   4.0f ,
4.0f,   5.0f , 4.0f, 4.0f , 5.0f,   5.0f , 4.0f,   5.0f , 5.0f, -5.0f , 4.0f,   -5.0f , 3.0f,   -4.0f , 3.0f, -5.0f , 4.0f,   -4.0f , 4.0f,   -
4.0f , 3.0f, 4.0f , 4.0f,   4.0f , 3.0f,   5.0f , 3.0f, 4.0f , 4.0f,   5.0f , 4.0f,   5.0f , 3.0f, -3.0f , 3.0f,   -3.0f , 2.0f,   -2.0f , 2.0f, -
3.0f , 3.0f,   -2.0f , 3.0f,   -2.0f , 2.0f, -2.0f , 3.0f,   -2.0f , 2.0f,   -1.0f , 2.0f, -2.0f , 3.0f,   -1.0f , 3.0f,   -1.0f , 2.0f, 0.0f ,
2.0f,   -1.0f , 3.0f,   -1.0f , 2.0f, 0.0f , 2.0f,   -1.0f , 3.0f,   0.0f , 3.0f, 0.0f , 2.0f,   1.0f , 2.0f,   0.0f , 3.0f,.0f , 3.0f,   1.0f ,
2.0f,   0.0f , 3.0f, 2.0f , 3.0f,   2.0f , 2.0f,   3.0f , 2.0f, 2.0f , 3.0f,   3.0f , 3.0f,   3.0f , 2.0f, 4.0f , 3.0f,   4.0f , 2.0f,   5.0f ,
2.0f, 4.0f , 3.0f,   5.0f , 3.0f,   5.0f , 2.0f,- 5.0f , 2.0f,   -5.0f , 1.0f,   -4.0f , 1.0f, -5.0f , 2.0f,   -4.0f , 2.0f,   -4.0f , 1.0f, -3.0f ,
2.0f,   -3.0f , 1.0f,   -2.0f , 1.0f, -3.0f , 2.0f,   -2.0f , 2.0f,   -2.0f , 1.0f,2.0f , 2.0f,   2.0f , 1.0f,   3.0f , 1.0f, 2.0f , 2.0f,   3.0f ,
2.0f,   3.0f , 1.0f, 4.0f , 2.0f,   4.0f , 1.0f,   5.0f , 1.0f,4.0f , 2.0f,   5.0f , 2.0f,   5.0f , 1.0f,-5.0f , 1.0f,   -5.0f , 0.0f,   -4.0f ,
0.0f, -3.0f , 1.0f,   -3.0f , 0.0f,   -2.0f , 0.0f, -3.0f , 1.0f,   -2.0f , 1.0f,   -2.0f , 0.0f, -1.0f , 1.0f,   -1.0f , 0.0f,   0.0f , 0.0f, -1.0f ,
1.0f,   0.0f , 1.0f,   0.0f , 0.0f,0.0f , 1.0f,   0.0f , 0.0f,   1.0f , 0.0f, 0.0f , 1.0f,   1.0f , 1.0f,   1.0f , 0.0f, 2.0f , 1.0f,   2.0f ,
0.0f,   3.0f , 0.0f, 2.0f , 1.0f,   3.0f , 1.0f,   3.0f , 0.0f, 4.0f , 1.0f,   4.0f , 0.0f,   5.0f , 0.0f, 4.0f , 1.0f,   5.0f , 1.0f,   5.0f , 0.0f,
-5.0f , 0.0f,   -5.0f , -1.0f,  -4.0f , -1.0f, -5.0f , 0.0f,   -4.0f , 0.0f,   -4.0f , -1.0f, 0.0f , 0.0f,   0.0f , -1.0f,   1.0f , -1.0f, 0.0f ,
0.0f,   1.0f , 0.0f,   1.0f , -1.0f, 4.0f , 0.0f,   4.0f , -1.0f,   5.0f , -1.0f, 4.0f , 0.0f,   5.0f , 0.0f,   5.0f , -1.0f, -5.0f , 1.0f,   -4.0f ,
1.0f,   -4.0f , 0.0f,-5.0f , -1.0f,   -5.0f , -2.0f,   -4.0f , -2.0f, -5.0f , -1.0f,   -4.0f , -1.0f,   -4.0f , -2.0f, -3.0f , -1.0f,   -3.0f , -
2.0f,   -2.0f , -2.0f, -3.0f , -1.0f,   -2.0f , -1.0f,   -2.0f , -2.0f,-2.0f , -1.0f,   -2.0f , -2.0f,   -1.0f , -2.0f, -2.0f , -1.0f,   -1.0f , -
1.0f,  -1.0f , -2.0f, 0.0f , -1.0f,   0.0f , -2.0f,   1.0f , -2.0f, 0.0f , -1.0f,   1.0f , -1.0f,   1.0f , -2.0f, 1.0f , -1.0f,   1.0f , -2.0f,   2.0f , -
2.0f, 1.0f , -1.0f,   2.0f , -1.0f,   2.0f , -2.0f, 2.0f , -1.0f,   2.0f , -2.0f,   3.0f , -2.0f, 2.0f , -1.0f,   3.0f , -2.0f,   3.0f , -2.0f, 4.0f ,
-1.0f,   4.0f , -2.0f,   5.0f , -2.0f,   -5.0f , -2.0f,   -5.0f , -3.0f,   -4.0f , -3.0f, 2.0f , -2.0f,   2.0f , -3.0f,   3.0f , -3.0f, 2.0f , -
2.0f,   3.0f , -2.0f,   3.0f , -3.0f, -5.0f , -3.0f,   -5.0f , -4.0f,   -4.0f , -4.0f, -5.0f , -3.0f,   -4.0f , -3.0f,   -4.0f , -4.0f, -3.0f , -
3.0f,   -3.0f , -4.0f,   -2.0f , -4.0f, -3.0f , -3.0f,   -2.0f , -3.0f,   -2.0f , -4.0f, -1.0f , -3.0f,   -1.0f , -4.0f,   0.0f , -4.0f, -1.0f , -
3.0f,   0.0f , -3.0f,   0.0f , -4.0f, 0.0f , -3.0f,   0.0f , -4.0f,   1.0f , -4.0f, 0.0f , -3.0f,   1.0f , -3.0f,   1.0f , -4.0f, 4.0f , -
3.0f,   4.0f , -4.0f,   5.0f , -4.0f, 4.0f , -3.0f,   5.0f , -3.0f,   5.0f , -4.0f, 2.0f , -1.0f,   3.0f , -1.0f,   3.0f , -2.0f, 4.0f , -
1.0f,   5.0f , -1.0f,   5.0f , -2.0f, -5.0f , -2.0f,   -4.0f , -2.0f,   -4.0f , -3.0f, -5.0f , -4.0f,   -5.0f , -5.0f,   -4.0f , -5.0f, -5.0f , -
4.0f,   -4.0f , -4.0f,   -4.0f , -5.0f, -4.0f , -4.0f,   -4.0f , -5.0f,   -3.0f , -5.0f,-4.0f , -4.0f,   -3.0f , -4.0f,   -3.0f , -5.0f, -3.0f , -
4.0f,   -3.0f , -5.0f,   -2.0f , -5.0f, -3.0f , -4.0f,   -2.0f , -4.0f,   -2.0f , -5.0f, -2.0f , -4.0f,   -2.0f , -5.0f,   -1.0f , -5.0f, -2.0f , -
4.0f,   -1.0f , -4.0f,   -1.0f , -5.0f, -1.0f , -4.0f,   -1.0f , -5.0f,   0.0f , -5.0f, -1.0f , -4.0f,   0.0f , -4.0f,   0.0f , -5.0f, 0.0f , -
4.0f,   1.0f , -4.0f,   1.0f , -5.0f, 1.0f , -4.0f,   1.0f , -5.0f,   2.0f , -5.0f, 1.0f , -4.0f,   2.0f , -4.0f,   2.0f , -5.0f, 2.0f , -
4.0f,   2.0f , -5.0f,   3.0f , -5.0f, 2.0f , -4.0f,   3.0f , -4.0f,   3.0f , -5.0f, 3.0f , -4.0f,   3.0f , -5.0f,   4.0f , -5.0f, 3.0f , -
4.0f,   4.0f , -4.0f,   4.0f , -5.0f,4.0f , -4.0f,   4.0f , -5.0f,   5.0f , -5.0f,4.0f , -4.0f,   5.0f , -4.0f,   5.0f , -5.0f,0.0f , -
4.0f,   0.0f , -5.0f,   1.0f , -5.0f  };
```

which includes all the coordinates of the triangles that make up the final maze. Additionally, in the command **glDrawArrays(GL_TRIANGLES, 0, 330)**, we specified the 330 vertices of these triangles (initially, the third argument of the function had a different value).

In the file **ProjectFragmentShader.fragmentshader**, we modified the command **color=vec3(1, 0, 0)** to **color=vec3(0, 0, 1)** in order to render the color blue.

More specifically, the method for finding the coordinates was done according to the following diagram:



For question (iii):

In this question, we were asked to design character A, as depicted in Image 1 of the prompt. The character must have specific dimensions and be displayed at the center of the square where it is located. Specifically, the following changes were made:

In the file **Source-1A.cpp**, we implemented the commands

```
float x = -4.75;
float y = 2.25;
```

```
GLfloat character_vertices[] = {
    x , (y + 0.5f), x , y, (x + 0.5f) , y,
    x , (y + 0.5f), (x + 0.5f) , (y + 0.5f), (x + 0.5f) , y
};
```

```
GLuint vertexbuffer2; glGenBuffers(1, &vertexbuffer2);

glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer2);

glBufferData(GL_ARRAY_BUFFER, sizeof(character_vertices),
character_vertices, GL_STATIC_DRAW);
```

```
glEnableVertexAttribArray(0);

glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer2);

glVertexAttribPointer( 0, // attribute 0, must match the layout
in the shader. 2, // size GL_FLOAT, // type GL_FALSE, //
normalized? 0, // stride (void*)0 // array buffer offset );

glDrawArrays(GL_TRIANGLES, 0, 6);
```

according to which we initialized the x and y coordinates (first cell of the above array) to create the coordinates for character A (second cell of the above array). Then, as in the previous question, we created VAOs and VBOs (third cell of the above array) and rendered them in the window (fourth cell of the above array).

2.1 Implementation of question (iv)

For question (iv):

In this question, we were asked to implement the movements for character A, as mentioned in the prompt. The character should be able to move within the maze, with constraints imposed by the walls and the pressing of specific keys. Specifically, the following changes were made:

In the file **Source-1A.cpp**, we implemented the commands

```
bool rightKeyPressed = false;
bool leftKeyPressed = false;
bool downKeyPressed = false;
bool upKeyPressed = false;

int maze[10][10] = { {1,1,1,1,1,1,1,1,1,1},
                     {1,0,0,0,0,0,0,0,0,1},
                     {0,0,1,1,1,1,0,1,0,1},
                     {1,0,1,0,0,0,0,1,0,1},
                     {1,0,1,0,1,1,0,1,0,1},
                     {1,0,0,0,0,1,0,0,0,1},
                     {1,0,1,1,0,1,1,1,0,1},
                     {1,0,0,0,0,0,0,1,0,0},
                     {1,0,1,0,1,1,0,0,0,1},
                     {1,1,1,1,1,1,1,1,1,1} };

bool isWall(float x, float y) {
    int col = static_cast<int>(x + 5.0f);
    int row = static_cast<int>(5.0f - y);
    if (row >= 0 && row < 10 && col >= 0 && col < 10) {
        return maze[row][col] == 1;  // Return true if it's a wall
    }
    return true;
}

void moveChar(float* x, float* y, GLfloat character_vertices[], GLFWwindow* window, GLuint vertexbuffer2) {
    float new_x = *x;
    float new_y = *y;
    if (glfwGetKey(window, GLFW_KEY_L) == GLFW_PRESS) {
        if (!rightKeyPressed) {
            std::cout << "Right key pressed\n";
            new_x += 1.0f;
            rightKeyPressed = true
        }
    } else {
        rightKeyPressed = false;
    }


if (glfwGetKey(window, GLFW_KEY_J) == GLFW_PRESS) {
    if (!leftKeyPressed) {
        std::cout << "Left key pressed\n";
        if (*x > 4.75f) { new_x -= 1.0f; }
        new_x -= 1.0f
        leftKeyPressed = true
    }
} else {
    leftKeyPressed = false;
}

if (glfwGetKey(window, GLFW_KEY_K) == GLFW_PRESS) {
    if (!downKeyPressed) {
        std::cout << "Down key pressed\n";
        new_y -= 1.0f;
        downKeyPressed = true;  // Set the flag to true
    }
} else {
    downKeyPressed = false;
}

if (glfwGetKey(window, GLFW_KEY_I) == GLFW_PRESS) {
    if (!upKeyPressed) {
        std::cout << "Up key pressed\n";
        new_y += 1.0f;
        upKeyPressed = true;
    }
} else {
    upKeyPressed = false;
}

if (!isWall(new_x, new_y)) {
    *x = new_x;
    *y = new_y;

    character_vertices[0] = *x;
    character_vertices[1] = *y + 0.5f;
    character_vertices[2] = *x;
    character_vertices[3] = *y;
    character_vertices[4] = *x + 0.5f;
    character_vertices[5] = *y;
    character_vertices[6] = *x;
    character_vertices[7] = *y + 0.5f;
    character_vertices[8] = *x + 0.5f;
    character_vertices[9] = *y + 0.5f;
    character_vertices[10] = *x + 0.5f;
    character_vertices[11] = *y;

    glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer2);
    glBufferData(GL_ARRAY_BUFFER, sizeof(float) * 12, character_vertices, GL_STATIC_DRAW);
    }
}
```

```
moveChar(&x, &y, character_vertices, window, vertexbuffer2);

                glEnableVertexAttribArray(0);

                glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer2);

                glVertexAttribPointer(0, 2, GL_FLOAT, GL_FALSE, 0, (void*)0);

                glDrawArrays(GL_TRIANGLES, 0, 6);

                glDisableVertexAttribArray(0);
```

according to which, in the first cell of the array, we initialized a **10x10 maze** array consisting of values 0 and 1, as defined in the prompt. We then set four **boolean** variables to **false** to provide feedback for key presses. Next, we implemented a function that returns a boolean value to check whether character A is attempting to move to a position that contains a wall (**isWall**). The function takes two **float values**, **x** and **y**, as arguments, corresponding to the new movement coordinates, and calculates the column and row values based on these coordinates. Specifically, the x coordinate is shifted by +5 (**x+5.0f**) to align with the center of the maze grid, where the center is considered the origin (0,0). The **static_cast<int>** converts the decimal value (**x+5.0f**) into an integer, necessary for calculating the column. Similarly, the y coordinate is shifted by +5 and subtracted from 5 (**5.0f-y**) to reverse the direction and align with the row numbering of the maze. The **static_cast<int>** also converts the result to an integer, just as for the column. The function then checks if the column and row values are within the bounds of the maze array and returns true if the position is a wall or if it is out of bounds of the maze. In the **moveChar()** function, which takes two float pointers, x and y, an array **GLfloat character_vertices[]**, a pointer **GLFWwindow *window**, and the buffer **GLuint vertexbuffer2**, we initialize two new float variables, **new_x** and **new_y**, assigning them the values of the pointers x and y. We check via the **glfwGetKey()** function if any of the movement keys are pressed, and if positive, we update the state of the corresponding boolean variable, adjusting the values of x and y accordingly (setting the boolean variable to true or false as appropriate). Notably, for left movement, we additionally check if the x pointer is greater than -4.75 before decreasing new_x by 1, to ensure it does not exceed the maze boundaries. If the point is not a wall, we return the new values to the pointers x and y, update the **character_vertices** with the new coordinates, bind **vertexbuffer2**, and update the data.

In the second cell of the array, we call **moveChar()** providing the necessary arguments within a **do-while loop** and render the character during each movement.

Referances

- Sample videos from the course website on e-course (Vasiliki Stamati).
- StackOverFlow-solution-for-u8