ΓΡΑΦΙΚΑ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΑ ΑΛΛΗΛΕΠΙΔΡΑΣΗΣ

Προγραμματιστική Άσκηση 1-Α.

Team

Γκόβαρης Χρήστος-Γρηγόριος
Σπανού Μαρία

Χρονοδιάγραμμα Εργαστηριακής Άσκησης 1-Α

Έκδοση	Ημερομηνία	Progress
1.0	21/10/2024	Ανακοίνωση εργαστηριακής άσκησης 1-Α
2.0	24/10/2024	Υλοποίηση των ερωτημάτων (i), (ii) και (iii)
2.1	25/10/2024	Υλοποίηση του ερωτήματος (iv)
3.0	25/10/2024	Υλοποίηση Report εργαστηριακής άσκησης

Ανάλυση Χαρακτηριστικών

Η εργασία υλοποιήθηκε σε ένα μηχάνημα (Lenovo Ideapad 3), με τα εξής χαρακτηριστικά πυρήνα:

- 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)

Επιπλέον, η **εργασία 1-Α** υλοποιήθηκε σε **Windows 11 Home**, χρησιμοποιώντας το **Visual Studio 2022 (x64)**.

Λειτουργία Ομάδας / Ανάλυση Ερωτημάτων

Η συνεργασία της ομάδας ήταν υποδειγματική, εξασφαλίζοντας την επιτυχή υλοποίηση της Άσκησης 1-Β. Και τα δύο μέλη συνέβαλαν ουσιαστικά στον αλγοριθμικό σχεδιασμό. Στο ερώτημα (i), καταλήξαμε επιτυχώς στο ζητούμενο αποτέλεσμα. Στο ερώτημα (ii), το μέλος 5203 ανέλαβε την αναπαράσταση του λαβύρινθου σε χαρτί, ενώ το μέλος 5351 διαμόρφωσε τον πίνακα shape_1_buffer[] με τις συντεταγμένες των 330 τριγώνων. Στο ερώτημα (iii), δημιουργήσαμε τον χαρακτήρα Α και προσδιορίσαμε τις συντεταγμένες του ώστε να είναι κεντραρισμένος στο πλέγμα. Στο ερώτημα (iv), αντιμετωπίσαμε πρόβλημα με την εντολή glfwGetKey, το οποίο λύσαμε με αλγοριθμική βελτιστοποίηση. Τέλος, στο ερώτημα (ii), ένα διπλότυπο τρίγωνο οδήγησε σε 333 κορυφές αντί για τις 330 που έπρεπε.

2.0 Υλοποίηση των ερωτημάτων (i), (ii) και (iii)

Για το ερώτημα (i):

Στο παρόν ερώτημα, μας ζητήθηκε να υλοποιήσουμε ένα πρόγραμμα που ανοίγει ένα βασικό παράθυρο 750x750, καθώς και να αλλάξουμε το χρώμα του background σε μαύρο. Πιο συγκεκριμένα, έγιναν οι εξής αλλαγές:

Στο αρχείο **Source-1A.cpp**, εντοπίσαμε την εντολή

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

η οποία καθορίζει το μέγεθος και τον τίτλο του παραθύρου (όπου αργότερα θα απεικονιστεί ο λαβύρινθος). Προχωρήσαμε στην αλλαγή των δύο πρώτων ορισμάτων της συνάρτησης **glfwCreateWindow** σε 750, ορίζοντας έτσι το μέγεθος του παραθύρου στις διαστάσεις 750x750.

Επιπροσθέτως, τροποποιήσαμε το τρίτο όρισμα της συνάρτησης, το οποίο αντιστοιχεί στον τίτλο του παραθύρου, αλλάζοντάς το σε «Άσκηση 1Α - 2024». Κατά το άνοιγμα του παραθύρου, αντιμετωπίσαμε ένα ζήτημα καθώς δεν υποστηρίζονταν ελληνικοί χαρακτήρες. Για την επίλυση του προβλήματος αυτού, προσθέσαμε το πρόθεμα «u8» πριν από τον επιθυμητό τίτλο του παραθύρου. Τέλος, προσθέσαμε τη δυνατότητα κλεισίματος του παραθύρου με το πλήκτρο Q.

Τέλος, όσον αφορά το χρώμα του υποβάθρου, εντοπίσαμε την εντολή

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

της οποίας τροποποιήσαμε κατάλληλα τα ορίσματα, ώστε να δημιουργηθεί σωστά το «Μαύρο Υπόβαθρο».

Για το ερώτημα (ii):

Στο παρόν ερώτημα, μας ζητήθηκε να σχεδιάσουμε τον λαβύρινθο όπως απεικονίζεται στην Εικόνα 1 της εκφώνησης. Πιο συγκεκριμένα, έγιναν οι εξής αλλαγές:

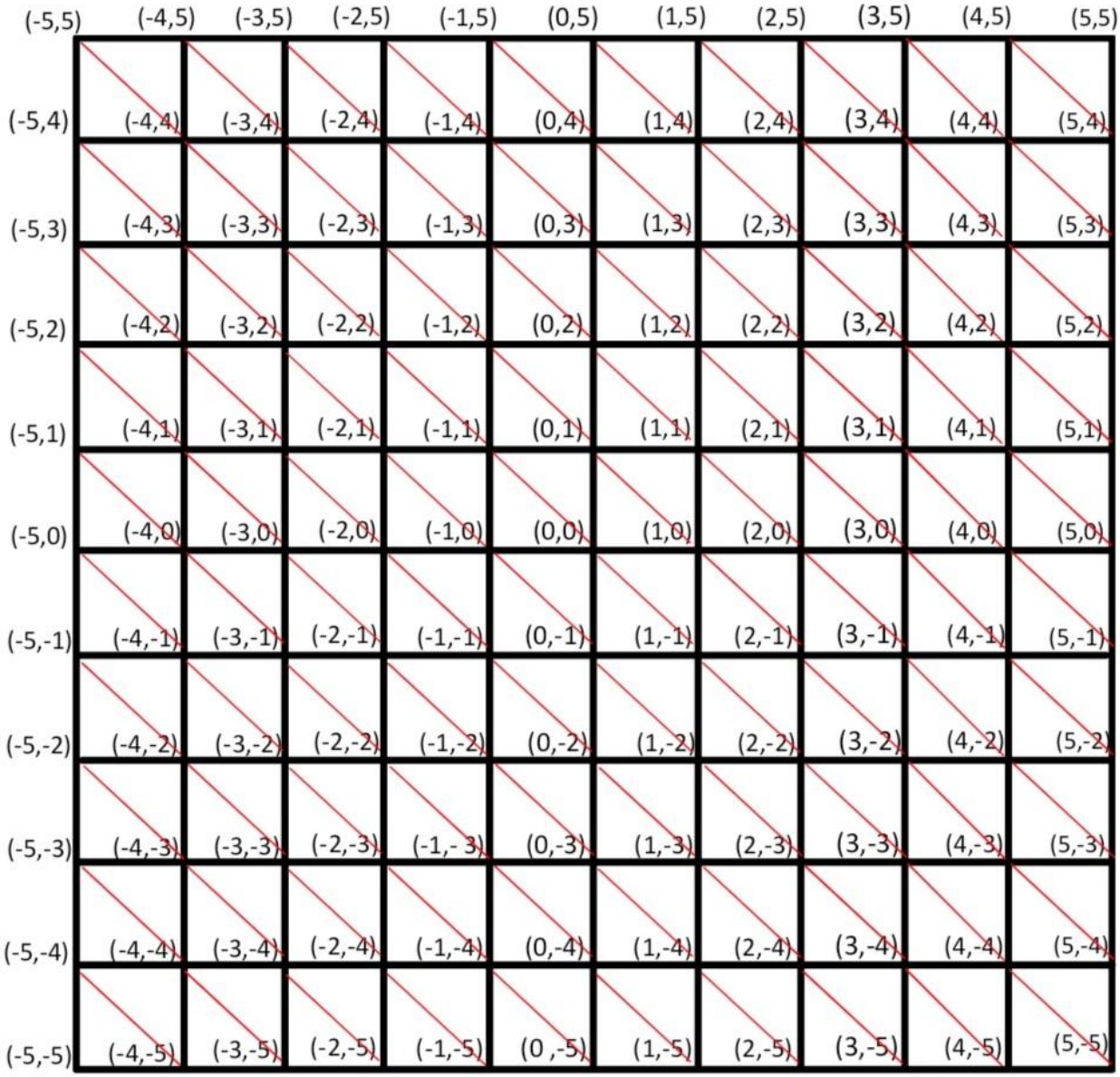
Στο αρχείο **Source-1A.cpp**, υλοποιήσαμε την εντολή

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, 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 };

στην οποία περιλαμβάνονται όλες οι συντεταγμένες των τριγώνων που συνθέτουν τον τελικό λαβύρινθο. Επιπλέον, στην εντολή **glDrawArrays(GL_TRIANGLES, 0, 330)**, καθορίσαμε τις 330 κορυφές των συγκεκριμένων τριγώνων (αρχικά, το τρίτο όρισμα της συνάρτησης είχε διαφορετική τιμή).

Στο αρχείο ProjectFragmentShader.fragmentshader, τροποποιήσαμε την εντολή **color=vec3(1, 0, 0)** σε **color=vec3(0, 0, 1)**, προκειμένου να αποδοθεί μπλε χρώμα.

Πιο αναλυτικά, ο τρόπος εύρεσης των συντεταγμένων, έγινε σύμφωνα με το παρακάτω σχήμα:



Για το ερώτημα (iii):

Στο παρόν ερώτημα, μας ζητήθηκε να σχεδιάσουμε τον χαρακτήρα Α, όπως απεικονίζεται στην Εικόνα 1 της εκφώνησης. Ο χαρακτήρας πρέπει να έχει συγκεκριμένες διαστάσεις και να απεικονίζεται στο κέντρο του τετραγώνου όπου βρίσκεται. Πιο συγκεκριμένα, έγιναν οι εξής αλλαγές:

Στο αρχείο **Source-1A.cpp**, υλοποιήσαμε τις εντολές

```
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);
```

σύμφωνα με τις οποίες, αρχικοποιήσαμε τις συντεταγμένες x και y (πρώτο κελί του παραπάνω πίνακα), ώστε να δημιουργήσουμε τις συντεταγμένες του χαρακτήρα Α (δεύτερο κελί του παραπάνω πίνακα). Στην συνέχεια, όπως και στο προηγούμενο ερώτημα, δημιουργήσαμε VAOs και VBOs (τρίτο κελί του παραπάνω πίνακα) και τα τυπώσαμε στο παράθυρο (τέταρτο κελί του παραπάνω πίνακα).

2.1 Υλοποίηση του ερωτήματος (iv)

Για το ερώτημα (iv):

Στο παρόν ερώτημα, μας ζητήθηκε να υλοποιήσουμε τις κινήσεις για τον χαρακτήρα Α, όπως αναφέρεται στην εκφώνηση. Ο χαρακτήρας θα πρέπει να μπορεί να μετακινείται μέσα στον λαβύρινθο, με περιορισμούς τα τοιχώματα και με την πίεση συγκεκριμένων πλήκτρων. Πιο συγκεκριμένα, έγιναν οι εξής αλλαγές:

Στο αρχείο **Source-1A.cpp**, υλοποιήσαμε τις εντολές

```
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);
```

σύμφωνα με τις οποίες, στο πρώτο κελί του πίνακα αρχικοποιήσαμε έναν πίνακα **maze 10x10**, ο οποίος αποτελείται από τιμές 0 και 1, όπως ορίζει η εκφώνηση. Στη συνέχεια, θέσαμε τέσσερις **boolean** μεταβλητές σε **false**, ώστε να παρέχουν πληροφορία (feedback) για την πίεση των πλήκτρων. Κατόπιν, υλοποιήσαμε μία συνάρτηση που επιστρέφει boolean τιμή για να ελέγχει εάν ο χαρακτήρας Α επιχειρεί να κινηθεί σε θέση που περιέχει τοίχο (**isWall**). Η συνάρτηση δέχεται ως ορίσματα δύο **float τιμές x** και **y**, που αντιστοιχούν στις νέες συντεταγμένες κίνησης, ενώ υπολογίζει τις τιμές στήλης και γραμμής με βάση αυτές τις συντεταγμένες. Συγκεκριμένα, η x συντεταγμένη μετατοπίζεται κατά +5 (**x+5.0f**) ώστε να ταιριάζει με το κέντρο του πλέγματος του λαβύρινθου, όπου το κέντρο θεωρείται ως η αρχή συντεταγμένων (0,0). Η **static_cast<int>** μετατρέπει την δεκαδική τιμή (**x+5.0f**) σε ακέραιο, απαραίτητο για τον υπολογισμό της στήλης. Αντίστοιχα, η y συντεταγμένη μετατοπίζεται κατά +5 και αφαιρείται από το 5 (**5.0f-y**) για να αντιστραφεί η κατεύθυνση και να ταιριάζει με την αρίθμηση των σειρών του λαβύρινθου. Η **static_cast<int>** μετατρέπει και εδώ το αποτέλεσμα σε ακέραιο, όπως και για τη στήλη. Η συνάρτηση στη συνέχεια ελέγχει αν οι τιμές στήλης και γραμμής βρίσκονται εντός των ορίων του πίνακα maze και επιστρέφει true αν η θέση είναι τοίχος ή αν βρίσκεται εκτός ορίων του λαβυρίνθου. Στη συνάρτηση **moveChar()**, η οποία δέχεται ως ορίσματα δύο float pointers x και y, έναν πίνακα **GLfloat character_vertices[]**, έναν δείκτη **GLFWwindow *window**, και τον **buffer GLuint vertexbuffer2**, αρχικοποιούμε δύο νέες float μεταβλητές, **new_x** και **new_y**, στις οποίες αποδίδουμε τις τιμές των pointers x και y. Ελέγχουμε μέσω της συνάρτησης **glfwGetKey()** αν πιέστηκε κάποιο από τα πλήκτρα κίνησης, και σε περίπτωση θετικού αποτελέσματος ενημερώνουμε ότι πιέστηκε, αυξομειώνουμε τις τιμές των x και y αναλόγως, και θέτουμε την αντίστοιχη boolean μεταβλητή σε true (διαφορετικά σε false). Σημειώνεται ότι για την αριστερή κίνηση ελέγχουμε επιπλέον αν το x pointer είναι μεγαλύτερο του -4.75 πριν μειώσουμε το new_x κατά 1, ώστε να μην υπερβεί τα όρια του λαβυρίνθου. Στη συνέχεια, εάν το σημείο δεν είναι τοίχος, επιστρέφουμε τις νέες τιμές στους pointers x και y, ενημερώνουμε το **character_vertices** με τις νέες συντεταγμένες, πραγματοποιούμε bind στο **vertexbuffer2** και ενημερώνουμε τα δεδομένα.

Στο δεύτερο κελί του πίνακα, καλούμε τη **moveChar()** παρέχοντας τα απαραίτητα ορίσματα μέσα σε ένα **do-while loop**, και σχεδιάζουμε τον χαρακτήρα σε κάθε του μετακίνηση.

Αναφορές

- Ενδεικτικά video από την ιστοσελίδα του μαθήματος στο e-course (Βασιλική Σταμάτη).
- StackOverFlow-solution-for-u8