Περιγραφή της εργασίας

Συνοπτικα:

Στην άσκηση αυτή θα δημιουργήσουμε ένα παράθυρο στο οποίο θα ζωγραφίζουμε τετράγωνα σε διάφορα σημεία του παραθύρου

Πιο αναλυτικά:

(i)

Φτιάχνουμε ένα πρόγραμμα που θα ανοίγει ένα βασικό παράθυρο 900x900 και δώσαμε τον τιτλο 'Πρώτη Άσκηση 2023'
προσθεσαμε το u8 για να αναγνωρισει τα ελληνικα γραμματα.

```
window = glfwCreateWindow(900, 900, u8"Πρώτη Ασκηση 2023", NULL, NULL);
//εδω ορισαμε το παραθυρο να ειναι 900x900 και δινουμε τιτλο στο παραθυρο δηλαδη το υποερωτημα 1..
```

το background color το καναμε μπλε βαζοντας 1 στην θεση B του RGB.

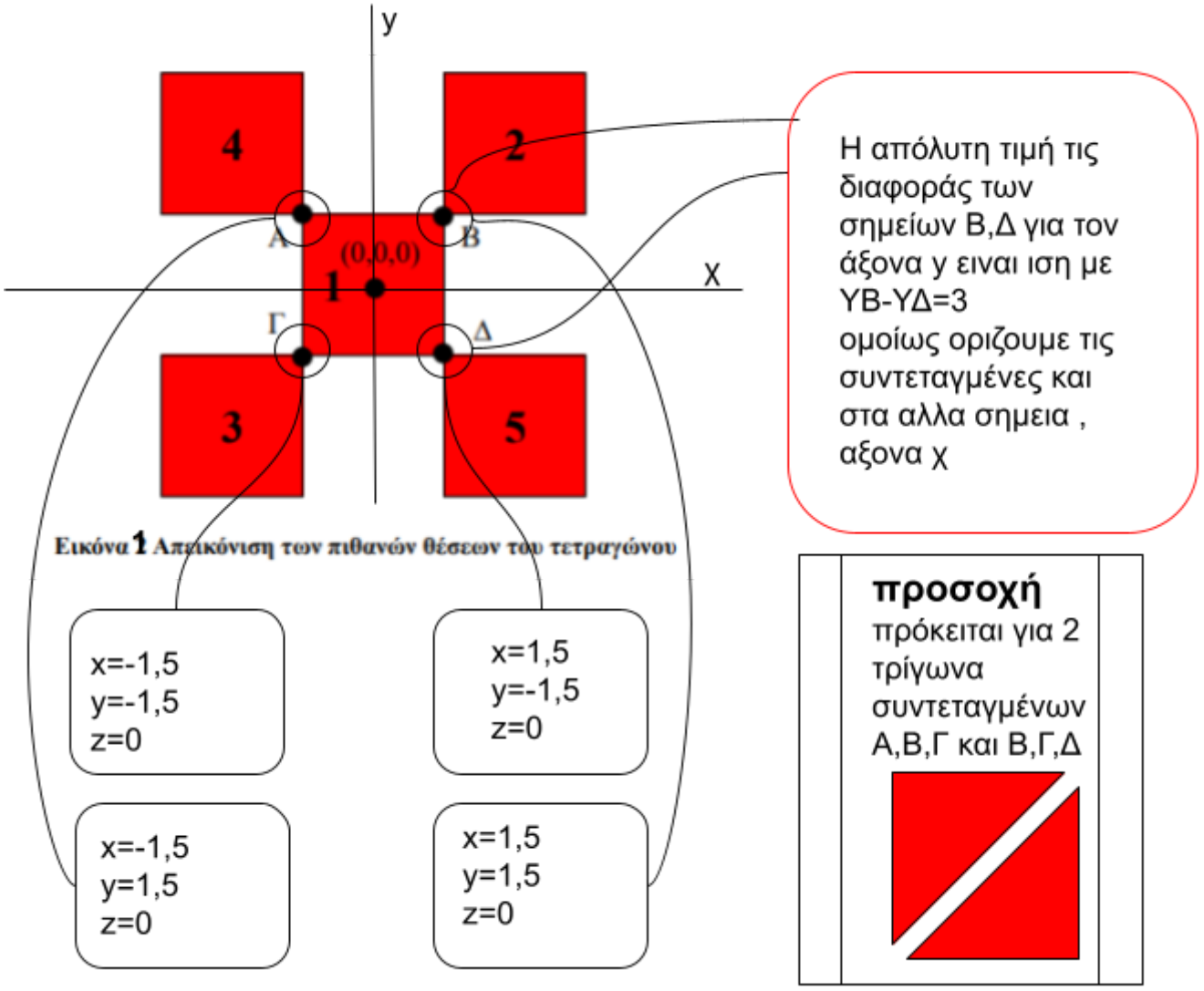
```
// blue background
glClearColor(0.0f, 0.0f, 1.0f, 0.0f);
```

Με το πλήκτρο C η εφαρμογή τερματίζει.

```
} while (glfwGetKey(window, GLFW_KEY_C) != GLFW_PRESS && glfwWindowShouldClose(window) == 0);
//αλλαξαμε το πληκτρο σε C οπως ζηταει το ερωτημα 1 για να κλεινει το παραθυρο
```

(ii)

ΞΕΚΙΝΆΜΕ ΖΩΓΡΑΦΊΖΟΝΤΑΣ ΈΝΑ ΤΕΤΡΆΓΩΝΟ, ΜΕ ΚΈΝΤΡΟ ΤΟ ΣΗΜΕΊΟ (0,0,0) ΤΟΥ ΕΠΙΠΈΔΟΥ ΚΑΙ ΠΛΕΥΡΆ Α, ΜΕ ΜΈΓΕΘΟΣ Α=3 (ΕΙΚΌΝΑ 1)



Εικόνα 1 Απεικόνιση των πιθανών θέσεων του τετραγώνου

Ο ΠΡΟΣΔΙΟΡΙΣΜΌΣ ΤΩΝ ΣΥΝΤΕΤΑΓΜΈΝΩΝ ΦΑΊΝΕΤΑΙ ΣΤΟ ΠΑΡΑΠΆΝΩ ΣΧΉΜΑ ΑΝΑΛΥΤΙΚΆ ΣΤΟ ΚΟΚΚΙΝΟ ΚΟΥΤΙ .Ο ΣΧΕΤΙΚΟΣ ΚΏΔΙΚΑΣ ΦΑΊΝΕΤΕ ΠΑΡΑΚΆΤΩ

```
static const GLfloat shape_1_buffer[] = {
    -1.5f, -1.5f, 0.0f,
    1.5f,  1.5f, 0.0f,
    1.5f ,-1.5f,0.0f
};

GLuint vertexbuffer;
glGenBuffers(1, &vertexbuffer);
glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer);
glBufferData(GL_ARRAY_BUFFER, sizeof(shape_1_buffer), shape_1_buffer, GL_STATIC_DRAW)

static const GLfloat shape_2_buffer[] = {
    -1.5f, -1.5f, 0.0f,
   1.5f,  1.5f, 0.0f,
   -1.5f,1.5f,0.0f
};

GLuint vertexbuffer2;
glGenBuffers(1, &vertexbuffer2);
glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer2);
glBufferData(GL_ARRAY_BUFFER, sizeof(shape_2_buffer), shape_2_buffer, GL_STATIC_DRAW)
```

ΠΑΡΑΚΆΤΩ ΦΑΊΝΕΤΑΙ Ο ΚΏΔΙΚΑΣ ΓΙΑ ΤΗΝ ΑΠΕΙΚΌΝΙΣΗ ΤΟΝ ΤΕΤΡΑΓΏΝΩΝ
ΔΗΜΙΟΥΡΓΉΣΑΜΕ ΑΥΤΉ ΤΗ ΣΥΝΆΡΤΗΣΗ ΓΙΑ ΝΑ ΑΠΛΟΠΟΙΉΣΟΥΜΕ ΤΟΝ ΚΏΔΙΚΑ

```
void square(GLuint vb, GLuint vb2) {
    // 1rst attribute buffer : vertices
    glEnableVertexAttribArray(0);
    glBindBuffer(GL_ARRAY_BUFFER, vb);
    glVertexAttribPointer(
        0,                  // attribute 0. No particular reason for 0, but must match the layout in the shader.
        3,                  // size
        GL_FLOAT,           // type
        GL_FALSE,           // normalized?
        0,                  // stride
        (void*)0            // array buffer offset
    );

    // Draw the triangle !
    glDrawArrays(GL_TRIANGLES, 0, 3); // 3 indices starting at 0 -> 1 triangle

    glBindBuffer(GL_ARRAY_BUFFER, vb2);
    glVertexAttribPointer(
        0,                  // attribute 0. No particular reason for 0, but must match the layout in the shader.
        3,                  // size
        GL_FLOAT,           // type
        GL_FALSE,           // normalized?
        0,                  // stride
        (void*)0            // array buffer offset
    );

    // Draw the triangle !
    glDrawArrays(GL_TRIANGLES, 0, 3); // 3 indices starting at 0 -> 1 triangle
    glDisableVertexAttribArray(0);

    // Swap buffers
    glfwSwapBuffers(window);
    glfwPollEvents();

    for (int i = 0; i < 6; i++) {
        cout << i << "\n";
    }
}
```

(iii)

Για αρχή θα ορίσουμε τις συντεταγμένες των τριγώνων τα οποία ανά δύο μας δημιουργούν το επιθυμητό τετράγωνο. Οι οποιες εχουν επιλεχθει με βαση την επεξήγηση που παρουσιάζεται στην εικονα1 (κοκκινο κουτι)

```
static const GLfloat shape_1_buffer[] = {
    -1.5f, -1.5f, 0.0f,
    1.5f,  1.5f, 0.0f,
    1.5f ,-1.5f,0.0f
};

GLuint vertexbuffer;
glGenBuffers(1, &vertexbuffer);
glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer);
glBufferData(GL_ARRAY_BUFFER, sizeof(shape_1_buffer), shape_1_buffer, GL_STATIC_DRAW);

static const GLfloat shape_2_buffer[] = {
    -1.5f, -1.5f, 0.0f,
   1.5f,  1.5f, 0.0f,
   -1.5f,1.5f,0.0f
};

GLuint vertexbuffer2;
glGenBuffers(1, &vertexbuffer2);
glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer2);
glBufferData(GL_ARRAY_BUFFER, sizeof(shape_2_buffer), shape_2_buffer, GL_STATIC_DRAW);

static const GLfloat shape_3_buffer[] = {
1.5f, 1.5f, 0.0f,
4.5f,  4.5f, 0.0f,
1.5f ,4.5f,0.0f
};

GLuint vertexbuffer3;
glGenBuffers(1, &vertexbuffer3);
glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer3);
glBufferData(GL_ARRAY_BUFFER, sizeof(shape_3_buffer), shape_3_buffer, GL_STATIC_DRAW);

static const GLfloat shape_4_buffer[] = {
    1.5f, 1.5f, 0.0f,
   4.5f,  4.5f, 0.0f,
   4.5f,1.5f,0.0f
};

GLuint vertexbuffer4;
glGenBuffers(1, &vertexbuffer4);
glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer4);
glBufferData(GL_ARRAY_BUFFER, sizeof(shape_4_buffer), shape_4_buffer, GL_STATIC_DRAW);

static const GLfloat shape_5_buffer[] = {
   -1.5f, -1.5f, 0.0f,
   -4.5f,  -4.5f, 0.0f,
   -1.5f ,-4.5f,0.0f
};

GLuint vertexbuffer5;
glGenBuffers(1, &vertexbuffer5);
glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer5);
glBufferData(GL_ARRAY_BUFFER, sizeof(shape_5_buffer), shape_5_buffer, GL_STATIC_DRAW);
```

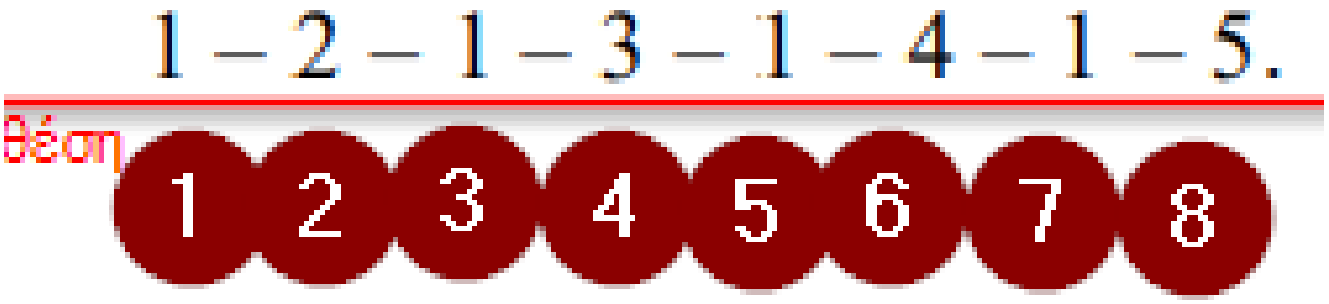
```
static const GLfloat shape_6_buffer[] = {
    -1.5f, -1.5f, 0.0f,
    -4.5f,  -4.5f, 0.0f,
    -4.5f,-1.5f,0.0f
};

GLuint vertexbuffer6;
glGenBuffers(1, &vertexbuffer6);
glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer6);
glBufferData(GL_ARRAY_BUFFER, sizeof(shape_6_buffer), shape_6_buffer, GL_STATIC_DRAW);

static const GLfloat shape_7_buffer[] = {
    -1.5f, 4.5f, 0.0f,
    -4.5f,  1.5f, 0.0f,
    -1.5f ,1.5f,0.0f
};

GLuint vertexbuffer7;
glGenBuffers(1, &vertexbuffer7);
glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer7);
glBufferData(GL_ARRAY_BUFFER, sizeof(shape_7_buffer), shape_7_buffer, GL_STATIC_DRAW);

static const GLfloat shape_8_buffer[] = {
    -1.5f, 4.5f, 0.0f,
    -4.5f,  1.5f, 0.0f,
    -4.5f,4.5f,0.0f
};
GLuint vertexbuffer8;
glGenBuffers(1, &vertexbuffer8);
glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer8);
glBufferData(GL_ARRAY_BUFFER, sizeof(shape_8_buffer), shape_8_buffer, GL_STATIC_DRAW);

static const GLfloat shape_9_buffer[] = {
    4.5f, -1.5f, 0.0f,
    1.5f,  -4.5f, 0.0f,
    1.5f ,-1.5f,0.0f
};

GLuint vertexbuffer9;
glGenBuffers(1, &vertexbuffer9);
glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer9);
glBufferData(GL_ARRAY_BUFFER, sizeof(shape_9_buffer), shape_9_buffer, GL_STATIC_DRAW);

static const GLfloat shape_10_buffer[] = {
    4.5f, -1.5f, 0.0f,
    1.5f,  -4.5f, 0.0f,
    4.5f,-4.5f,0.0f
};
GLuint vertexbuffer10;
glGenBuffers(1, &vertexbuffer10);
glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer10);
glBufferData(GL_ARRAY_BUFFER, sizeof(shape_10_buffer), shape_10_buffer, GL_STATIC_DRAW);
```

Το τετράγωνο θα εμφανίζεται στο χώρο, ακολουθώντας ένα συγκεκριμένο μοτίβο κίνησης: 1 – 2 – 1 – 3 – 1 – 4 – 1 – 5.
Ο χρόνος ανάμεσα σε κάθε εναλλαγή θέσης να είναι σταθερός και αρκετός ώστε να φανεί η απεικόνιση του τετραγώνου (int time).

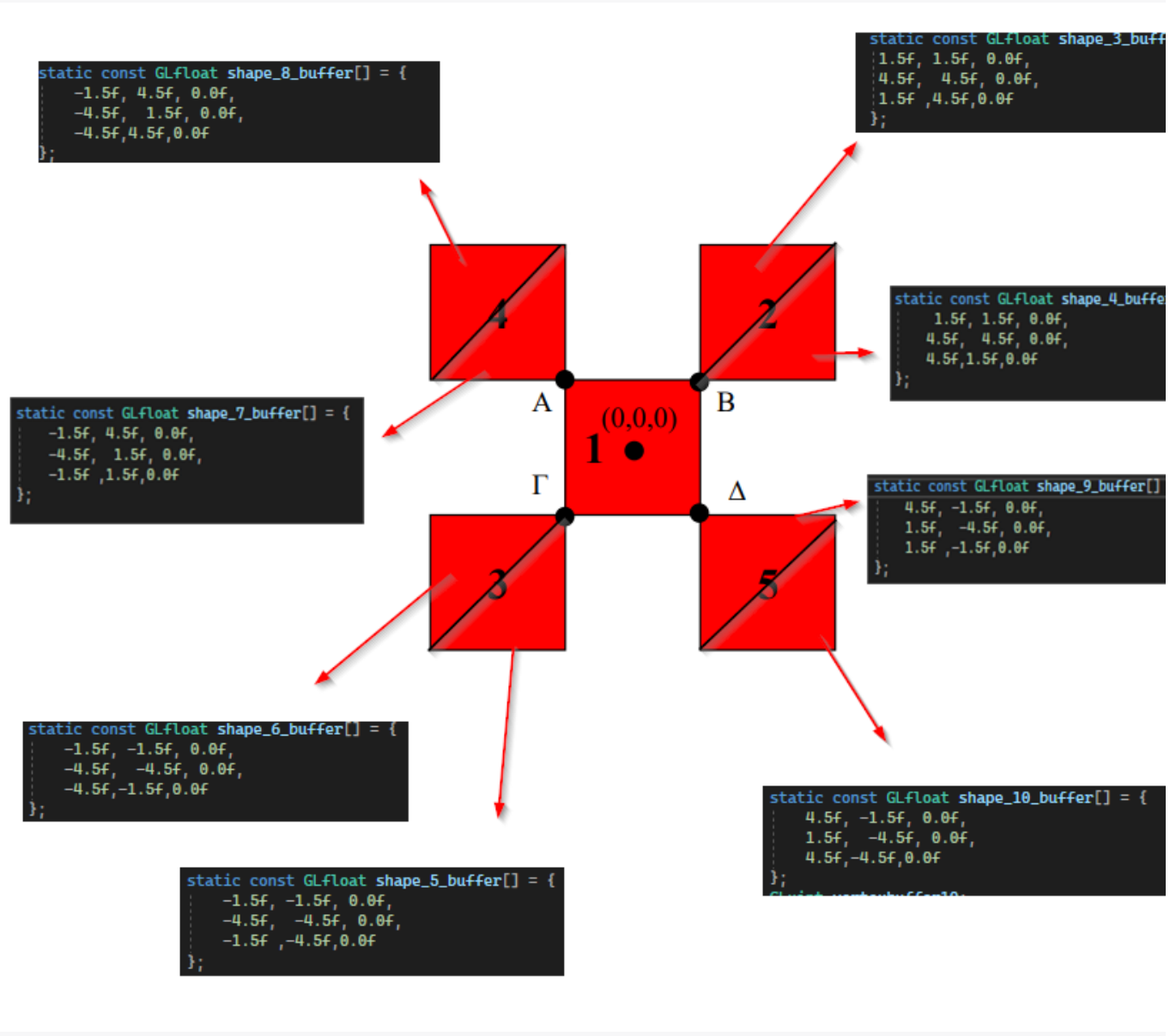
```
int j = 1; //ο μετρητης για το μοτιβο

int time = 300; //ο χρονος για την αναλλαγει του μοτιβου
```

```
if (j == 1 or j == 3 or j == 5 or j == 7) {
    square(vertexbuffer, vertexbuffer2);
    Sleep(time);
    j++;
}
else if (j == 2 ) {
    square(vertexbuffer3, vertexbuffer4);
    Sleep(time);
    j++;
}
else if (j == 4) {
    square(vertexbuffer5, vertexbuffer6);
    Sleep(time);
    j++;
}
else if (j == 6) {
    square(vertexbuffer7, vertexbuffer8);
    Sleep(time);
    j++;
}
else if (j == 8) {
    square(vertexbuffer9, vertexbuffer10);
    Sleep(time);
    j = j - 7;
}
```

Έχουμε φτιάξει ένα μετρητή J ο οποίος κάθε φορά που δεν έχει πατηθεί το πλήκτρο C(τερματισμός προγράμματος),με συνθήκες if ελέγχει σε ποια θέση του J βρισκόμαστε ανάλογα τη θέση ζωγραφίζει το ανάλογο τετράγωνο και αυξάνει τον μετρητή j κατά ένα. Όταν ο μετρητής φτάσει στο τελευταίο τετράγωνο δηλαδή J=8 κάνουμε αρχικοποίηση τον μετρητή ώστε το μοτίβο να επαναλαμβάνεται.



Επεξήγηση συντεταγμένων των τριγώνων 2,3,4,5



(iv)

Αν πατηθεί το κουμπί U αυξάνεται ο ρυθμός εναλλαγής του μοτίβου ενώ όταν πατηθεί το κουμπί D ελαττώνεται ο ρυθμός εναλλαγής

```
if (glfwGetKey(window, GLFW_KEY_U) != GLFW_PRESS) {
    time = time + 20;
}if(glfwGetKey(window, GLFW_KEY_D) != GLFW_PRESS) {
    time = time - 20;
}
```

Πληροφορίες σχετικά με την υλοποίηση

- Λειτουργικό Σύστημα (Windows)
- Περιβάλλον/Editor (Visual Studio x64)

Αξιολόγηση της λειτουργίας της ομάδας

Το project υλοποιήθηκε Από την αρχή μέχρι το τέλος με την παρουσία και των δύο! υπήρχε άρτια συνεννόηση και συνεισφορά

Αναφορές

https://www.glfw.org/docs/3.3/window_guide.html
https://www.cs.uoi.gr/%7Efudos/cs-93-16.pdf

TR CS-93-16

A Guide to C++ for C Programmers

Thomas A. Anastasio

Computer Science Department
University of Maryland, Baltimore County
Baltimore, MD 21228

27 October 1993