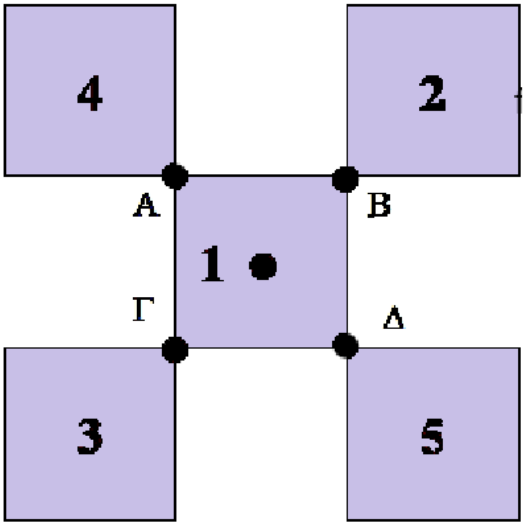
UNIVERSITY of IOANNINA

ΕΡΓΑΣΙΑ 1Α

Σκοπός της άσκησης είναι η απεικόνιση ενός τετράγωνου σε διάφορα σημεία του παραθύρου.



1. Περιγραφή της εργασίας

i. Στην γραμμή 149 του κώδικα με την εντολή αυτή καθορίζονται τα χαρακτηριστικά του παραθύρου ,όπως ζητείται και από την άσκηση με διαστάσεις 900X900 και με τίτλο <<Πρώτη άσκηση 2023>>.

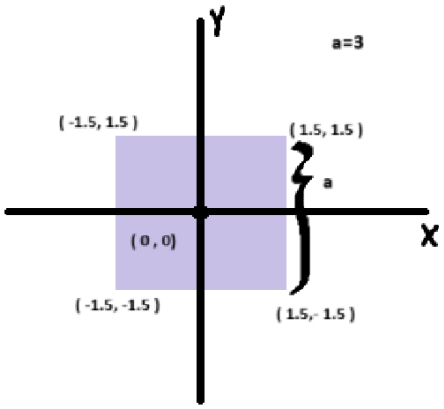
```
148     // Open a window and create its OpenGL context
149     window = glfwCreateWindow(900, 900, "Πρώτη άσκηση 2023", NULL, NULL);
150
151
```

Η εντολή στην γραμμή 173 παίρνει σαν ορίσματα 4 νούμερα, εκ των οποίων τα τρία πρώτα που μας ενδιαφέρουν είναι τα rgb. Συνεπώς για την απόχρωση του μπλε background του παραθύρου δίνουμε τις τιμές 0, 0,1,1 .

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

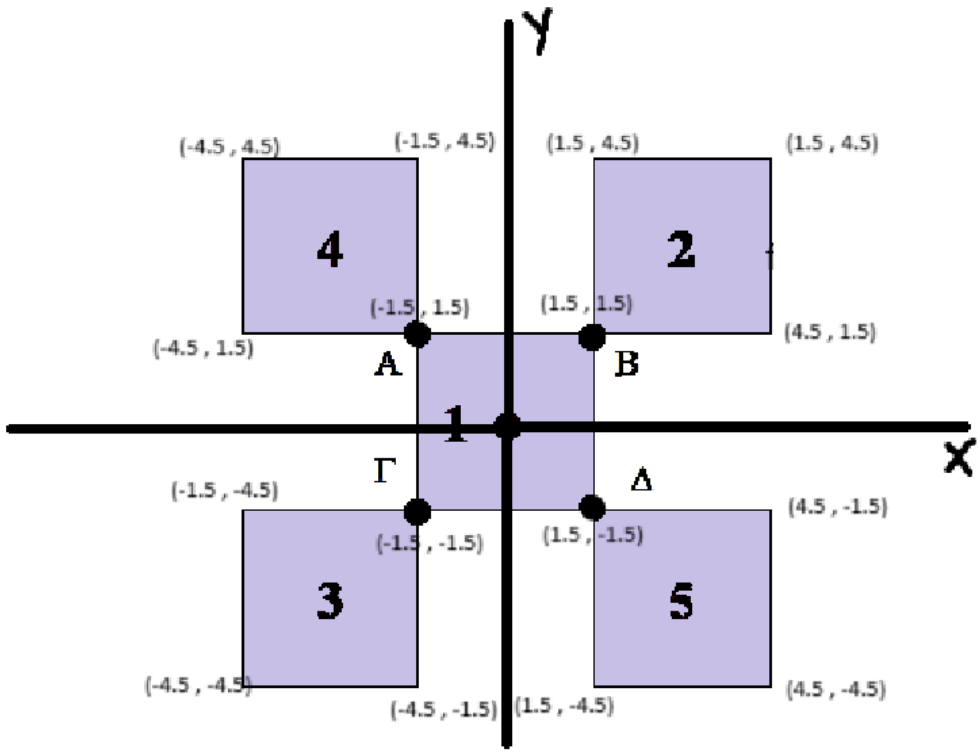
Με την παρακάτω εντολή στην γραμμή 333, πατώντας το πλήκτρο C η εφαρμογή τερματίζεται

```
333 } while (glfwGetKey(window, GLFW_KEY_C) != GLFW_PRESS && glfwWindowShouldClose(window) == 0);
334
```



ii.

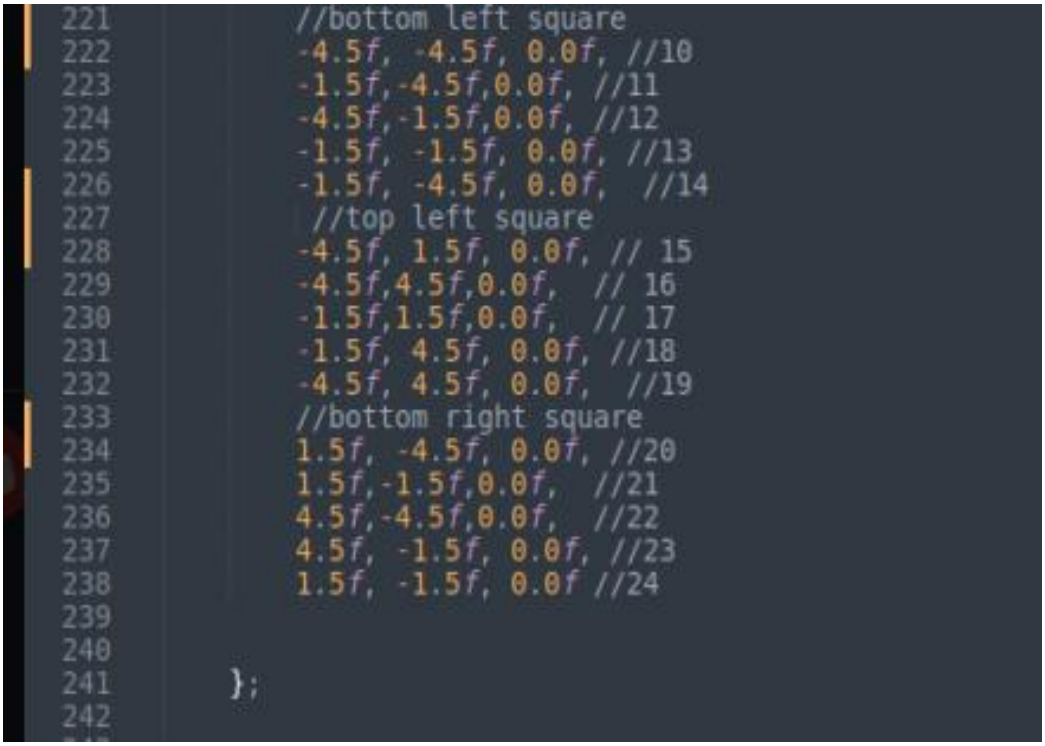
Στην παραπάνω εικόνα αναπαριστούμε την μορφή του κεντρικού τετραγώνου με σκοπό την εύρεση των συντεταγμένων **x,y** , ώστε να τις εντάξουμε στον (πίνακα) **buffer** του κώδικα μας .Πιο αναλυτικά, εφόσον το κέντρο του τετραγώνου είναι στην αρχή των αξόνων (0,0) και η πλευρά του είναι **a=3** προκύπτουν οι παραπάνω συντεταγμένες .



iii. Αναπαριστούμε την τελική μορφή του σχεδίου με τα συνολικά τετράγωνα και τις συντεταγμένες τους .

Στην συνέχεια ακολουθεί ο πίνακας στον οποίο αποθηκεύουμε τις συντεταγμένες(xyz) κάθε κορυφής όλων των τετραγώνων.

```
205
206
207     static const GLfloat shape_1_buffer[] = {
208
209         //center square
210         -1.5f, -1.5f, 0.0f, //  0
211         -1.5f, 1.5f, 0.0f, //   1
212         1.5f, -1.5f, 0.0f, // 2
213         1.5f, 1.5f, 0.0f, //3
214         -1.5f, 1.5f, 0.0f, //4
215         //top right square
216         1.5f, 1.5f, 0.0f, //5
217         1.5f,4.5f,0.0f,   //6
218         4.5f,1.5f,0.0f, // 7
219         4.5f, 4.5f, 0.0f, //8
220         1.5f, 4.5f, 0.0f, //9
```



```
221         //bottom left square
222         -4.5f, -4.5f, 0.0f, //10
223         -1.5f,-4.5f,0.0f, //11
224         -4.5f,-1.5f,0.0f, //12
225         -1.5f, -1.5f, 0.0f, //13
226         -1.5f, -4.5f, 0.0f,  //14
227          //top left square
228         -4.5f, 1.5f, 0.0f, // 15
229         -4.5f,4.5f,0.0f,  // 16
230         -1.5f,1.5f,0.0f,  // 17
231         -1.5f, 4.5f, 0.0f, //18
232         -4.5f, 4.5f, 0.0f,  //19
233         //bottom right square
234         1.5f, -4.5f, 0.0f, //20
235         1.5f,-1.5f,0.0f,  //21
236         4.5f,-4.5f,0.0f,  //22
237         4.5f, -1.5f, 0.0f, //23
238         1.5f, -1.5f, 0.0f //24
239
240
241     };
242
```

Στον κώδικα προσθέσαμε τα παρακάτω **if** με σκοπό την διαδοχική απεικόνιση των τετραγώνων. Με βάση το μονοπάτι που μας δόθηκε από την εκφώνηση **(1-2-1-3-1-4-1-5)** η τιμή του i αντιστοιχεί σε κάθε τετράγωνο. Συγκεκριμένα όταν το i=1,3,5,7 εμφανίζεται το *τετράγωνο 1*,όταν i=2 εμφανίζεται το *τετράγωνο 2* , όταν i=4 εμφανίζεται το *τετράγωνο 3,* όταν i=6 εμφανίζεται το *τετράγωνο 4* και όταν i=8 εμφανίζεται το *τετράγωνο 5* . Στο τέλος κάθε if το i αυξάνεται κατά ένα (γραμμή κώδικα 321) . Αν η τιμή του i είναι μεγαλύτερη του 8 τότε επανέρχεται στην τιμή 1 . Αυτό διασφαλίζει ότι θα εκτελεστεί κάθε τμήμα του κώδικα για κάθε τιμή του i από 1 έως 8 (γραμμή κώδικα 324).

Μέσα σε κάθε **if** τυπώνεται το αντίστοιχο τετράγωνο με τις συντεταγμένες του πίνακα buffer.Συγκεκριμένα η **#glDrawArrays(…, …, …)** δέχεται σαν δεύτερο όρισμα τη θέση του στοιχείου στον πίνακα και σαν τρίτο όρισμα το πλήθος των κορυφών.

Για να αποφύγουμε την ταυτόχρονη εμφάνιση των τετραγώνων χρησιμοποιήσαμε την συνάρτηση **sleep**. Με την **Sleep(1)** επιτυγχάνουμε την καθυστέρηση του καθενός τετράγωνου κατά 1 δευτερόλεπτο .

```
272
273         if (i == 1 || i == 3 || i == 5 || i == 7) {
274             // Draw the triangle !
275             glDrawArrays(GL_TRIANGLES, 0, 3);
276             glDrawArrays(GL_TRIANGLES, 2, 3);
277             sleep(1);
278         }
279
280         if (i == 2) {
281             // 2nd attribute buffer : vertices
282             glEnableVertexAttribArray(0);
283
284             // Draw the triangle !
285             glDrawArrays(GL_TRIANGLES, 5, 3);
286             glDrawArrays(GL_TRIANGLES, 7, 3);
287             sleep(1);
288         }
289
290         if (i == 4) {
291             // 3rd attribute buffer : vertices
292             glEnableVertexAttribArray(0);
293
294             // Draw the triangle !
295             glDrawArrays(GL_TRIANGLES, 10, 3);
296             glDrawArrays(GL_TRIANGLES, 12, 3);
297             sleep(1);
298         }
299
```

```
300         if (i == 6) {
301             // 4th attribute buffer : vertices
302             glEnableVertexAttribArray(0);
303
304             // Draw the triangle !
305             glDrawArrays(GL_TRIANGLES, 15, 3);
306             glDrawArrays(GL_TRIANGLES, 17, 3);
307             sleep(1);
308         }
309
310         if (i == 8) {
311             // 5th attribute buffer : vertices
312             glEnableVertexAttribArray(0);
313
314             // Draw the triangle !
315             glDrawArrays(GL_TRIANGLES, 20, 3);
316             glDrawArrays(GL_TRIANGLES, 22, 3);
317             sleep(1);
318         }
319
320         // increase i
321         i++;
322
323
324         if (i > 8) {
325             i = 1;
326         }
327
328         glDisableVertexAttribArray(0);
329
330         // Swap buffers
331         glfwSwapBuffers(window);
332         glfwPollEvents();
333     } while (glfwGetKey(window, GLFW_KEY_C) != GLFW_PRESS && glfwWin
334
```

```
242
243     int i = 1;
244     float patternSwitchRate = 1.0f; //set the initial switching rate
245
```

Αρχικά, ορίζουμε τον αρχικό ρυθμό εναλλαγής.

```
266
267         if (glfwGetKey(window, GLFW_KEY_U) == GLFW_PRESS) {
268             patternSwitchRate += 0.2f; //increase switching rate when U is pressed
269         }
270
271         if (glfwGetKey(window, GLFW_KEY_D) == GLFW_PRESS) {
272             patternSwitchRate -= 0.2f; //decrease switching rate when D is pressed
273         }
274
275         //Make sure the switching rate is not negative or too small
276         if (patternSwitchRate < 0.1f) {
277             patternSwitchRate = 0.1f;
278         }
279
280         usleep((int)(patternSwitchRate * 1000000)); //convert the switching rate to microseconds
281
```

Προσθέτουμε στον βρόχο do-while τα δύο if στα οποία με το αντίστοιχο πλήκτρο αυξάνεται/μειώνεται ο ρυθμός εναλλαγής του μοτίβου.Επίσης, χρησιμοποιείται έλεγχος για να αποφευχθούν αρνητικές τιμές και τέλος επειδή το sleep είναι σε microseconds, πολλαπλασιάζουμε τον ρυθμό εναλλαγής με 1000000.

2. Πληροφορίες σχετικά με την υλοποίηση:

Λειτουργικό Σύστημα: Linux

Text Editor : Sublime Text

3. Αξιολόγηση Ομάδας:

Η εργασία εκπονήθηκε από δύο άτομα, όπου ο καθένας έδινε ιδέες για την σχεδίαση της εφαρμογής. Η συνεργασία ήταν άψογη και παρουσιάστηκε συνέπεια ως προς την υλοποίηση της άσκησης.

4. Αναφορές – Πηγές που χρησιμοποιήθηκαν κατά την εκπόνηση της εργασίας.

Δεν χρειάστηκε η χρήση κάποιας πηγής