ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ.

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ.

ΓΡΑΦΙΚΑ ΥΠΟΛΟΓΙΣΤΩΝ.

ΧΕΙΜΕΡΙΝΟ ΕΞΑΜΗΝΟ 15-16.

ΣΤΕΡΝΙΩΤΗΣ ΕΛΕΥΘΕΡΙΟΣ | Α.Μ: 2363.

ΑΣΗΜΑΚΟΠΟΥΛΟΣ ΓΕΩΡΓΙΟΣ | Α.Μ: 2216.

2η ΣΕΙΡΑ ΑΣΚΗΣΕΩΝ.

ΤΕΛΙΚΗ ΑΝΑΦΟΡΑ ΕΡΓΑΣΙΑΣ.

ΕΚΤΕΛΕΣΗ ΠΡΟΓΡΑΜΜΑΤΟΣ:

Η δημιουργία και η εκτέλεση της τρέχουσας εργαστηριακής άσκησης πραγματοποιήθηκε αποκλειστικά σε λειτουργικό σύστημα Windows 7.Το πρόγραμμα που χρησιμοποιήθηκε για την εκτέλεση ήταν το Dev-C++ (έκδοση 5.4.1) με την εισαγωγή των απαραίτητων βιβλιοθηκών για την εκτέλεση προγραμμάτων σε OpenGL περιβάλλον. Για την εκτέλεση της εργασίας σε λειτουργικό σύστημα Ubuntu (όπως των εργαστηρίων) ανοίγουμε ένα νέο τερματικό στο φάκελο που περιέχει το αρχείο .cpp,το αρχείο file.maz όπως και τις 3 εικόνες PNG αλλά και την βιβλιοθήκη SOIL για το άνοιγμα των τριών εικόνων. Επίσης, για την εκτέλεση της εργασίας είναι απαραίτητη και η παρουσία του Makefile αν η εκτέλεση του γίνει όπως παρακάτω, διαφορετικά θα πρέπει να περάσουμε τις παραμέτρους που απαταίτουνται στο τερματικό μαζί με την εντολή εκτέλεσης.Τελικά, ανοίγουμε το τερματικό και εκτελούμε την παρακάτω εντολή:

```
make PROG=project2
```

Έπειτα αφού το compile του προγράμματος γίνει χωρίς κάποιο λάθος εκτελούμε την παρακάτω εντολή στο τρέχων τερματικό:

```
./project2 file.maz
```

ή γενικότερα την εντολή:

```
./project2 <file name>
```

Έχουμε προσθέσει την λειτουργία η μέθοδος main() να δέχεται ως όρισμα το όνομα του αρχείου που θέλουμε να διαβάσει έτσι ώστε να φορτωθεί σωστά ο λαβύρινθος, όπως φαίνεται παραπάνω στη δεύτερη κατά σειρά εντολή.

Η τρέχουσα εργαστηριακή άσκηση δοκιμάστηκε επίσης και στα εργαστήρια του τμήματος μας, όπου με τις απαραίτητες αλλαγές σε βιβλιοθήκες και εντολές τρέχει κανονικά εμφανίζοντας τον λαβύρινθο, μαζί με όλες τις επιπλέον λειτουργίες όπου αυτή απαιτεί.

ΕΠΕΞΗΓΗΣΗ ΛΕΙΤΟΥΡΓΙΑΣ:

Θα αρχίσουμε την επεξήγηση της τρέχουσας εργαστηριακής άσκησης με το να αναλύσουμε την κάθε μέθοδο ξεχωριστά και τι ρόλο επιτελεί στο τελικό κώδικα του προγράμματος:

1.void readMazeFile()

Η τρέχουσα μέθοδος είναι μια από τις πιο βασικές του προγράμματος μας. Σε αυτή τη μέθοδο το πρόγραμμα μας διαβάζει το αρχείο του λαβυρίνθου file.maz και το αποθηκεύει σε ένα πίνακα τριών διαστάσεων maze. Επίσης, μέσω αυτής της μεθόδου διαβάζονται και όλες οι επιπλέον πληροφορίες που μας παρέχει το παραπάνω αρχείο όπως πλήθος σφυριών του παίκτη, πλήθος επιπέδων αλλά και μέγεθος του λαβυρίνθου. Αρχικά ανοίγουμε το αρχείο ως ένα απλό txt αρχείο διαβάζοντας και αποθηκεύοντας σε μεταβλητές τα πρώτα τρία στοιχεία του αρχείου. Στη συνέχεια διαβάζουμε και αποθηκεύουμε στο πίνακα του λαβυρίνθου όλα τα στοιχεία του (τρείς εικόνες, κόκκινο, μπλε, πράσινο,κενό,μαύρο για μεταφορά). Τα οκτώ στοιχεία του λαβυρίνθου έχουν γίνει define στην αρχή του κώδικα για μεγαλύτερη ευκολία στην χρήση τους μέσα στο πρόγραμμα όπου χρειάζεται.

2.void loadTextures()

Η μέθοδος αυτή μας βοηθά να εισάγουμε τις τρείς εικόνες που δίδονται στην εκφώνηση της άσκησης μέσα στο πρόγραμμα μας. Εκτελούμε τρείς φορές το ίδιο μπλοκ εντολών για κάθε μια απο τις εικόνες. Αποθηκεύουμε τις πληροφορίες κάθε εικόνας σε ένα πίνακα τριών διαστάσεων GLuint textures[3] όπου σε κάθε στοιχείο του πίνακα εκχωρούμε την πληροφορία για κάθε εικόνα. Ο παραπάνω πίνακας χρησιμοποιείται μέσα στη μέθοδο renderScene() με την βοήθεια της μεθόδου drawCubeTexture() όπου σε κάθε πλευρά του κύβου "κολλάμε" το αντίστοιχο texture για να δημιουργηθεί ορθά το κυβάκι. Οι παραπάνω μέθοδοι θα εξηγηθούν σε παρακάτω παραγράφους.

3.void drawCubeColor(float r,float g,float b)

Η μέθοδος αυτή μας βοηθά στην δημιουργία ενός και μόνο κύβου. Όπως φαίνεται και από τον κώδικα της τρέχουσας μεθόδου δημιουργούμε την κάθε πλευρά του κύβου ξεχωριστά αφού είναι ένα 3D αντικείμενο θα έχει 6 πλευρές. Η κάθε μια από τις πλευρές τοποθετείται με διαφορετικές συντεταγμένες από τις άλλες, επιπλέον κάθε πλευρά του κύβου μπορεί να πάρει διαφορετικό χρώμα αν εμείς το επιθυμούμε. Στην άσκηση αυτή θέλουμε κάθε κύβος να έχει ένα συγκεκριμένο χρώμα (κόκκινο, μπλε, πράσινο,μαύρο). Η μέθοδος αυτή καλείται μέσα στην διαδικασία renderScene() για να δημιουργήσει τους κύβους ανάλογα με τον τύπο και το χρώμα που διαβάζει από τον πίνακα maze που πήραμε από το αρχείο file.maz. Σύμφωνα με τον τύπο κυβακίου που θα διαβαστεί απο το παραπάνω αρχείο καλούμε τη τρέχουσα μέθοδο με τα σωστά ορίσματα για να βάψει το αντίστοιχο χρώμα που θέλουμε. Η μέθοδος αυτή χρησιμοποείται μόνο για να δημιουργήσει κυβάκια τα οποία έχουν τα τρία χρώματα που αναφέραμε παραπάνω και όχι για να "κολλήσει" τις εικόνες. Η ενέργεια αυτή θα πραγματοποιηθεί απο την επόμενη μέθοδο.

4.void drawCubeTexture(float r,float g,float b,GLuint texture)

Η τρέχουσα μέθοδος λειτουργεί ακριβώς με τον ίδιο τρόπο όπως η παραπάνω. Δημιουργούμε τις 6 πλευρές του κύβου ξεχωριστά η κάθε μια σε διαφορετικές συντεταγμένες. Η βασικότερη διαφορά με την παραπάνω συνάρτηση είναι ότι στην αρχή και στο τέλος της μεθόδου καλούμε δύο νέες εντολές έτσι ώστε να μπορέσουμε να ενεργοποιήσουμε αρχικά και τέλος να απενεργοποίησουμε την διαδικασία φόρτωσης των εικόνων png. Επίσης, σε κάθε πλευρά καλούμε και την εντολή glTexCoord2i() για να την ορθή φόρτωση των εικόνων στα τέσσερα σημεία που ορίζουν πλευρά ενός κύβου. Η μέθοδος δέχεται τέσσερα ορίσματα, τα πρώτα τρία αναφέρονται στα διαφορετικά χρώματα όπως στη προηγούμενη μέθοδος και το τελευταίο όρισμα αναφέρεται στην εικόνα που θέλουμε να φορτώσουμε κάθε φορά για κάθε εικόνα δίδουμε και το αντίστοιχο όρισμα σύμφωνα με τα στοιχεία του πίνακα που είναι αποθηκευμένες οι πληροφορίες τους.

5.void processNormalKeys(unsigned char key,int xx,int yy)

Η μέθοδος αυτή λειτουργεί αποκλειστικά με την βοήθεια του πληκτρολογίου. Ο χρήστης πατώντας τα απαραίτητα πλήκτρα εκτελεί και την αντίστοιχη λειτουργία, όπως αυτή περιγράφεται από την εκφώνηση της εργασίας. Με το πλήκτρο ESC το παιχνίδι τερματίζει την λειτουργία του και το παράθυρου κλείνει αυτόματα, με το πλήκτρο X όπου και αν είναι ο παίκτης το παιχνίδι θα τερματίσει και θα εμφανίσει το τελικό σκορ. Με τα πλήκτρα W και S ο

παίκτης κινείται εμπρός και πίσω αντίστοιχα, με το πλήκτρο H καλείται η μέθοδος destroyCube() η οποία καταστρέφει τον κύβο όπου ο παίκτης κοιτά. Με το πλήκτρο R η κάμερα περιστρέφεται γύρω από τον λαβύρινθο, με το πλήκτρο V αν η κάμερα είναι εκτός του λαβυρίνθου εστιάζει στο εσωτερικό του και αν είναι εντός συμβαίνει η αντίθετη λειτουργία. Με το πλήκτρο Spacebar αν δεν υπάρχει κάποιος κύβος πάνω από τον παίκτη τότε αυτός μεταβαίνει στο αμέσως από πάνω επίπεδο από αυτό που βρισκόταν αρχικά. Με το πλήκτρο E αν ο παίκτης βρίσκεται στο τελευταίο επίπεδο του λαβυρίνθου (επίπεδο L) τότε το παιχνίδι τερματίζει και εμφανίζεται προς στον χρήστη το τελικό σκορ. Έχουμε επίσης προσθέσει και μια επιπλέον λειτουργία πάνω σε αυτή τη μέθοδο πατώντας το πλήκτρο A ο χρήστης μπορεί να κάνει επανακκίνηση του παιχνιδίου αφού πρώτα έχει πατηθεί απο αυτόν το πλήκτρο X.

6.void mouseMove(int x, int y)

Η μέθοδος αυτή μας βοηθά να μετακινούμε την κάμερα με την βοήθεια του ποντικιού και όχι των πλήκτρων. Η κάμερα κινείται γύρω από τον παίκτη χωρίς φυσικά να τον μετακινεί από την θέση του. Η πορεία όπου κοιτά είναι η επόμενη κίνηση του παίκτη. Η κάμερα κινείται ως προς όλες τις κατευθύνσεις ανάλογα με την κίνηση του ποντικιού (πάνω, αριστερά, κάτω, δεξιά).

7.void destroyCube()

Η συνάρτηση αυτή όπως αναφέρει και το όνομα της μας βοηθά να καταστρέψουμε ένα κύβο μέσα στο λαβύρινθο. Ο κύβος ο οποίος καταστρέφεται είναι εκείνος όπου ο παίκτης κοίτα μέσα στο λαβύρινθο. Αφού ο παίκτης έχει εστιάσει σε ένα συγκεκριμένο κύβο πατώντας το πλήκτρο H όπως περιγράφηκε παραπάνω καταστρέφει τον αντίστοιχο κύβο, μειώνοντας επίσης και τον αριθμό των σφυριών που έχει στην διάθεση του. Η δράση αυτή πραγματοποιείται μέσα στο σώμα της εντολής if, όπου αν ο παίκτης κοιτά σε ένα μη κενό τετράγωνο τότε πατώντας το αντίστοιχο πλήκτρο H αυτό γίνεται αυτόματα κενό, οπότε με αυτό τον τρόπο καταστρέφεται, αφαιρώντας ταυτόχρονα και ένα σφυρί από την διάθεση του παίκτη. Τέλος καλείται ξανά η μέθοδος renderScene() για να δημιουργήσει την σκηνή του λαβυρίνθου ξανά από την αρχή χωρίς τον κύβο που καταστράφηκε.

8.void changeSize(int w, int h)

Η μέθοδος αυτή χρησιμοποιείτε όπως ακριβώς είχε δοθεί και στα παραδείγματα του φροντιστηρίου, μας βοηθά όταν αλλάζουμε τις διαστάσεις του παραθύρου όπου εμφανίζεται ο λαβύρινθος, τα πίξελς να μην παραμορφώνονται, έτσι ώστε και μετά την αλλαγή του παραθύρου να εμφανίζεται ακριβώς η ίδια εικόνα της εφαρμογής μας όπως ήταν αρχικά, απλά με μεγαλύτερο μέγεθος σχημάτων.

9.draw_cylinder(GLfloat radius,GLfloat height)

Η μέθοδος αυτή όπως αναφέρει και το όνομα της, μας βοηθά να δημιουργήσουμε έναν κύλινδρο. Ο κύλινδρος θα εμφανίζεται μέσα στο λαβύρινθο όταν η κάμερα βρίσκεται από ψηλά είτε στην αρχή του προγράμματος είτε όταν έχουμε πατήσει το πλήκτρο V άρτιο αριθμό φορών. Ο κύλινδρος θα αντιπροσωπεύει το σώμα του παίκτη έτσι ώστε να γνωρίζουμε που ακριβώς βρίσκεται κάθε φορά μέσα στο λαβύρινθο, όταν ο παίκτης κινείται θα μετακινείται ταυτόχρονα και ο κύλινδρος. Η συνάρτηση αυτή δέχεται δύο ορίσματα, το πρώτο αναφέρεται στην ακτίνα που επιθυμούμε να έχεις ο κύλινδρος μας και το δεύτερο στο ύψος του.

10.void renderBitmapString(float x,float y,float z,void *font,char *string)

Η μέθοδος αυτή μας βοηθά να γράψουμε κείμενο στην οθόνη του παραθύρου. Δέχεται πέντε ορίσματα συνολικά. Τα τρία πρώτα αρχικά ορίσματα προσδιορίζουν το σημείο που θα τοποθετηθεί το κείμενο στους τρείς άξονες. Το τέταρτο κατα σειρά όρισμα δέχεται και προσδιορίζει την γραμματοσειρά που θα χρησιμοποιηθεί για την προβολή του κειμένου. Τέλος,το πέμπτο όρισμα προσδιορίζει το κείμενο που θα προβληθεί στην οθόνη.

11.void computePos(float deltaMove)

Η μέθοδος αυτή χρησιμοποείται για υπολογισμό του νέου σημείου που θα προβάλει η κάμερα μετά απο την κίνηση του παίκτη. Παίρνει ως μοναδικό όρισμα την μεταβολή της κίνησης και υπολογίζει τα νέα σημεία (μεταβλητές xx και zz). Μέσα στο σώμα της εντολής if, εκτελούμε την κίνηση του παίκτη μέσα στο λαβύρινθο αν και μόνο αν είναι εντός ορίων του λαβυρίνθου και το κυβάκι το οποίο κοιτά η κάμερα είναι κενό ή μαύρο, έτσι ώστε να μπορέσει να κινηθεί γιατί διαφορετικά θα συγκρουστεί με κάποιο υπαρκτό κυβάκι. Αν το κυβάκι το οποίο κοιτά είναι μαύρο και κινηθεί προς αυτό τότε όπως αναφέρεται και στο μπόνους ερώτημα ο παίκτης μπορεί να τηλεμεταφερθεί

σε ένα αντίστοιχο μαύρο κυβάκι του ίδιου επιπέδου και μόνο. Επίσης, μέσα στο σώμα της εντολής if υπολογίζουμε και την περίπτωση που ο παίκτης βρίσκεται σε ανώτερο επίπεδο, δηλαδή ο χρήστης έχει πατήσει ήδη το Space bar. Σε αυτή την περίπτωση καλείται ο βρόχος while, όπου όσο ο παίκτης βρίσκεται σε ανώτερο επίπεδο απο το πρώτο και το επόμενο κυβάκι που θέλει να κινηθεί,το ακριβώς απο κάτω του έχει κενό τότε θα πέσει στο απο κάτω ακριβώς επίπεδο γιατί δεν θα μπορεί να κινείται στον αέρα, μόνο στη περίπτωση όπου υπάρχουν κανονικά κυβάκια στο κάτω επίπεδο ο παίκτης που θα βρίσκεται σε ανώτερο θα μπορεί να κινείται επάνω τους.

12.void renderScene(void)

Η μέθοδος αυτή στηρίζει ολόκληρο το πρόγραμμα και βοηθά σε ένα μεγάλο ποσοστό της λειτουργίας του. Αρχικά, μέσω της μεθόδου αυτής δημιουργούμε το έδαφος όπου πάνω σε αυτό θα τοποθετηθεί ο λαβύρινθος που διαβάζεται απο το αρχείο. Επίσης, μέσω της μεθόδου δημιουργείται και ο λαβύρινθος, παίρνωντας τον τριών διαστάσεων πίνακα maze και χρησιμοποιώντας τρείς βρόχους for μέσα στα όρια που απαιτούνται για την κατασκευη του. Ανάλογα με τον τύπο αρχείου που θα συναντήσει το πρόγραμμα στο πίνακα δημιουργεί και το αντίστοιχο κύβο καλώντας την μέθοδο drawCube(...) για να χρωματισεί απλά ένα κύβο με τα τρία χρώματα και την μέθοδο drawCubeTexture(...) όπως περιγράφηκε παραπάνω όταν θέλουμε να κολλήσουμε στις πλευρές των κύβων εικόνες ,τα στοιχεία από τα όποια περιέχεται ο πίνακας είναι επτά,μαζί με το κενό. Τα εν λόγω στοιχεία έχουν γίνει define στην αρχή του προγράμματος για περισσότερη ευκολία στην χρήση τους. Επίσης, μέσα στο σώμα της συνάρτησης αυτής υπολογίζεται το τελικό σκορ που θα έχει ο χρήστης με βάση τον τύπο που δίδεται στην εκφώνηση της άσκησης. Η μεταβλητή moves στον τύπο αναφέρεται στις κινήσεις του παίκτη μέσα στο λαβύρινθο και η μεταβλητή initialK αναφέρεται στα αρχικά σφυριά που είχε ο παίκτης όταν άρχισε το παιχνίδι. Χρησιμοποιώντας την διαδικασία void renderBitmapString() όπως περιγράφηκε παραπάνω εκχωρούμε σε μια μεταβλητή τύπου char το κείμενο που επιθυμούμε να γράψουμε στην οθόνη και παιρνώντας ως όρισμα στην παραπάνω συνάρτηση εκτυπώνεται στην οθόνη. Επιπλέον, η συνάρτηση αυτή χρησιμοποιήτε και για τον υπολογισμό της οπτικής γωνίας της κάμερας του χρήστη ανάλογα με την πλευρά που κοιτά μέσα στο λαβύρινθο με την βοήθεια της συνάρτησης computePos(). Επίσης, με τη βοήθεια της μεθόδου μπορούμε και περιστρέφουμε την κάμερα γύρω-γύρω απο το λαβύρινθο αφού ο χρήστης έχει πατήσει το πλήκτρο R. Όσο ο χρήστης πατά το πλήκτρο αυτό η κάμερα κινείται δείχνοντας το λαβύρινθο απέξω και απο διαφορετική οπτική γωνία κάθε φορά. Για αυτό το λόγο χρησιμοποιούνται και οι πολλές συνθήκες if - else if για να μπορέσουμε να καλύψουμε όλες τις περιπτώσεις

όπου θα ο χρήστης θα πατήσει το συγκεκριμένο πλήκτρο. Η τελευταία συνθήκη else καλύπτει την περίπτωση όπου ο χρήστης θα πατήσει το πλήκτρο V και η κάμερα θα κινηθεί στο εσωτερικό του λαβυρίνθου, εκείνη την στιγμή απενεργοποείται η λειτουργία του πλήκτρου R, όταν πατηθεί ξανά το πλήκτρο V τότε ενεργοποείται ξανά.

13. setOrthographicProjection()

Η μέθοδος αυτή μετατρέπει το 3D περιβάλλον σε 2D έτσι ώστε να μπορέσουμε να γράψουμε στην οθόνη του παραθύρου κείμενο χρησιμοποιώντας την μέθοδο void renderBitmapString(...) όπως περιγράφηκε παραπάνω για να μην υπάρχει πρόβλημα στην εμφάνιση του κειμένου. Πρέπει όμως να εκτελέσουμε και την αντίθετη διαδικασία για να επαναφέρουμε το περιβάλλον όπως αρχικά ήταν.

14.restorePerspectiveProjection()

Η μέθοδος αυτή εκτελεί την ακριβώς αντίθετη διαδικασία απο την παραπάνω. Αφού έχει γίνει η εγγραφή του κειμένου στο παράθυρο επαναφέρουμε το περιβάλλον απο 2D σε 3D όπως ήταν αρχικά για να μπορέσουν να εκτελέστουν οι υπόλοιπες λειτουργίες.

15.int main(int argc, char **argv)

Η κεντρική μέθοδος του προγράμματος μας, όπου μέσω αυτής καλούνται όλες συναρτήσεις που περιγράφηκαν στις παραπάνω παραγράφους, με σκοπό την εκτέλεση όλων των επιμέρους λειτουργιών του παιχνιδιού. Αρχικά, η μέθοδος αυτή διαβάζει το αρχείο του λαβυρίνθου και τοποθετεί τον παίκτη σε ένα τυχαίο σημείο στους τρείς άξονες αν και μόνο αν το σημείο αυτό περιέχει κενό κυβάκι. Στη συνέχεια καλούμε τις τυπικές συναρτήσεις της βιβλιοθήκης Glut για την δημιουργία του παραθύρου, το όνομα, τις διαστάσεις του αλλά και τις συναρτήσεις που μας βοηθούν να διαχειριστούμε τις εισόδους απο πληκτρολόγιο και ποντίκι αντίστοιχα. Επίσης, καλούμε και συνάρτησεις για την κίνηση του ποντικιού, δηλαδή της κάμερας του παίκτη. Η κλήση της μεθόδου loadTextures() μας βοηθά στη φόρτωση των τριών PNG εικόνων όπως περιγράφηκε στη παράγραφο 2. Τέλος, καλούμε την έτοιμη συνάρτηση της βιβλιοθήκης Glut, την glutMainLoop() έτσι ώστε το παράθυρο μας να μείνει μέχρι εμείς να το κλείσουμε στην οθόνη.