Πανεπιστήμιο Ιωαννίνων | Γραφικά Υπολογιστών και Συστήματα Αλληλεπίδρασης | Χειμ 19-20

Αναφορά πρώτης προγραμματιστικής άσκησης

ΑΓΓΕΛΟΥΛΗΣ ΓΕΩΡΓΙΟΣ 1995
ΑΝΤΩΝΙΟΥ ΣΕΡΑΦΕΙΜ ΗΛΙΑΣ 2640

Περιγραφή της άσκησης:

Το πρόγραμμα ξεκινά από τη main, αρχικοποιείται το παράθυρο με βάση την εκφώνηση καθώς και ορίζονται οι βασικές συναρτήσεις της glut.
Καλείται η **initializeScene()** η οποία αρχικοποιεί τη τρισδιάστατη σκηνή μας, η **initializeMenu()** η οποία ορίζει το menu το οποίο θα καλείται με το δεξί κλικ, η **initializeCubes()** η οποία αρχικοποιεί τους κύβους και η **initializeTextures()** η οποία φορτώνει τα τρία textures που θα χρησιμοποιηθούν.

Ο τρόπος με τον οποίο έχει οριστεί ένας κύβος είναι ένα stracture στο οποίο κρατιούνται οι συντεταγμένες του κύβου, ο τύπος του (μεταξύ των 6 που ορίζει η εκφώνηση), καθώς και ένα ψεύτικο χρώμα που χαρακτηρίζει μοναδικά τον κύβο (θα εξηγηθεί παρακάτω).

```
struct Cube {
    int x;
    int y;
    int type;
    float fakeColor;
};
```

Έχουν γίνει define οι εξής αριθμοί, για την ευκολία της ανάθεσης τύπου σε κάθε κύβο.

```
#define C_WHITE 0
#define C_SCISSORS 1
#define C_ROCK 2
#define C_PAPER 3
#define C_RED 4
#define C_BLUE 5
#define C_NONE 6
```

Επίσης, κρατάμε σε global μεταβλητές το πλήθος των κινήσεων που έχουν γίνει, το score του παίκτη και ένα δισδιάστατο πίνακα 15x15 όπου κρατάμε τους κύβους του παιχνιδιού.

Κατά την αρχικοποίηση των κύβων όλα είναι τύπου C_WHITE. Σε αυτό το σημείο περιμένουμε ώστε ο χρήστης να πατήσει την επιλογή Start Game από το menu ωστε να ξεκινήσει το παιχνίδι.

SCREENSHOT HERE w/ white cubes

Μόλις το παιχνίδι ξεκινήσει, σε όλους τους κύβους ανατίθεται ένας τυχαίος τύπος (τυχαίος αριθμός μεταξύ του 1 και του 5). Η συνάρτηση renderScene() η οποία έχει οριστεί ως βασική συνάρτηση display της glut, με βάση αυτούς τους αριθμούς κάνει render αντιστοίχως τους κύβους.

SCREENSHOT HERE w/ colored cubes

Πιο συγκεκριμένα, στη **renderScene**() ορίζουμε ένα φως λίγο μπροστά από το grid μας (συγκεκριμένα στη θέση (7, 7, 5)), ορίζουμε την gluLookAt ώστε να κοιτάζει στο κέντρο του grid και να είναι αρχικά στη θέση (7, 7, 20) η οποία θέση της κάμερας είναι παραμετροποιήσιμη ώστε να μπορεί να μεταβληθεί με τα βελάκια, και καλούμε την **drawCubes**().

Η drawCubes() με τη σειρά της, για κάθε ένα τυπο κύβου που έχουμε, φορτώνει το αντίστοιχο texture ή χρώμα και καλεί για κάθε ένα κύβο την drawCube() η οποία φτιάχνει ένα κύβο (6 rectangles) γύρω από τις συντεταγμένες του εκάστοτε κύβου.

Για τα textures χρησιμοποιήθηκε η βιβλιοθήκη libpng η οποία είναι προεγκατεστημένη στα εργαστήρια του τμήματος. Με τη βοήθεια αυτής της βιβλιοθήκης, διαβάσαμε τρία αρχεία εικόνας png για κάθε ένα από τα στοιχεία που χρειάζεται τα οποία και σώσαμε σε τρεις global μεταβλητές και τα οποία φορτώνουμε κάθε φορά που θέλουμε να χρησιμοποιήσουμε.

```
GLubyte *textureScissors;
GLubyte *texturePaper;
GLubyte *textureRock;
```

Η συνάρτηση **initializeTextures**() είναι υπεύθυνη για το διάβασμα των αρχείων εικόνας και σώσιμο των textures και η loadTexture() καλείται κάθε φορά που θέλουμε να φορτώσουμε κάποιο.

Αφού το παιχνίδι έχει ξεκινήσει, περιμένουμε ώσπου ο χρήστης να ενεργήσει. Συγκεκριμένα, περιμένουμε από τον χρήστη να πατήσει σε κάποιο κύβο και να σύρει με το ποντίκι του τον επιλεγμένο κύβο σε κάποιο γειτονικό. Για την αναγνώριση του κύβου που πατάει ο χρήστης χρησιμοποιήθηκε η τεχνική Color Picking. Στη συγκεκριμένη τεχνική, ζωγραφίζουμε ξανα τη σκηνή μας αλλά αυτή τη φορά κάθε κύβος έχει ένα μοναδικό χρώμα (χωρίς textures). Η σκηνή δε γίνεται ποτέ render αλλά εμείς μπορούμε να διαβάσουμε το χρώμα του συγκεκριμένου pixel που έχει κάνει κλικ ο χρήστης μέσω του buffer της σκηνής ακόμα και όταν αυτός δεν έχει γίνει flush.
Κατά την αρχικοποίηση των κύβων αναθέσαμε το πεδίο fakeColor όπου για το 15x15 grid μας, αναθέσαμε έναν αριθμό σε κάθε κύβο από το 0 ως το 225.
Μέσω της συνάρτησης **renderFakeScene**(), γεμίζουμε τον buffer με μια πανομοιότυπη σκηνή όσον αφορά το σύστημα συντεταγμένων και την θέση της κάμερας αλλά κάθε κύβος ζωγραφίζεται με μία μοναδική απόχρωση του γκρι μεταξύ των (0, 0, 0) και (225, 225, 225) σε σύστημα rgb. Επίσης, επειδή ξεκινάμε από το 0, κάναμε το clearColor (background color) λευκό (255, 255, 255) ώστε να αναγνωρίσουμε όταν ο χρήστης δεν πάτησε κάποιο κύβο.

Με αυτή τη λογική κατασκευάστηκε η συνάρτηση που διαβάζει τα κλικ του ποντικιού mouseFunc() η οποία κατά το πάτημα κάτω του αριστερού κλικ κρατάει τον κύβο που πατήθηκε και με το γεγονός αριστερό κλικ πάνω, καλεί τη συνάρτηση **checkAndSwapCubes**() περνώντας ως όρισμα τους δύο κύβους με τους οποίους ο χρήστης αλληλεπίδρασε.

Η συνάρτηση **checkAndSwapCubes**() καλύπτει την λειτουργικότητα που θέλουμε να δώσουμε όσον αφορά τη λογική του παιχνιδιού. Συγκεκριμένα, για τους δύο κύβους που δέχεται ως όρισμα καλεί την **areNeighbours**() η οποία επιστρέφει True/False για το αν οι δύο κύβοι συνορεύουν, και εφόσον αυτό ισχύει καλεί την swapCubes() η οποία κάνει το swap στους δύο τύπους των κύβων. Εάν ο ένας κύβος είναι τύπου None δηλαδή έχει καταστραφεί και ουσιαστικά έχουμε απλά μετακίνηση κύβου και όχι swap, τότε καλείται η **destroyEatingNeighbours**() με όρισμα τον κύβο που μετακινήθηκε και ελέγχει τους γείτονες του κύβου. Εάν “τρώει” κάποιο από αυτούς, τότε μετατρέπει τον αντίστοιχο κύβο σε τύπου None. Στην περίπτωση που έχω swap και όχι μετακίνηση, καλείται δύο φορές η συνάρτηση **checkTriad**() μία φορά για κάθε ένα από τους κύβους που έγιναν swap.

Η **checkTriad**() ελέγχει για πιθανό σχηματισμό τριάδας ίδιου τύπου στον κύβο που έχει ως ορισμα. Εάν βρει ίδιου τύπου καλεί την **destroyTriad**() για τους τρεις κύβους που βρήκε ότι έχουν ίδιο τύπο.

Η **destroyTriad**() αφού καταστρέψει την τριάδα (θέσει τύπου None τους κύβους που την αποτελούν), ελέγχει αν έχω οριζόντια ή κατακόρυφη τριάδα και καλεί τις **doArea1()** και **doArea2and3**() με όρισμα τον κεντρικό κύβο και ένα boolean για το αν έχω οριζόντια ή κατακόρυφη τριάδα.

Η **doArea1**() και **doArea2and3**() ελέγχουν για τους κανόνες που αναφέρονται στην εκφώνηση όσον αφορά τις περιοχές 1, 2, 3 και εκτελεί τις αντίστοιχες ενέργειες.

Στις παραπάνω ενέργειες το score του παίκτη ανανεώνεται με βάση την εκφώνηση με τη χρήση της μεταβλητής **points**. Επίσης, όταν γίνεται μια κίνηση, ο μετρητής **movesMade** αυξανει κατα ενα.

Στις global μεταβλητές υπάρχουν επίσης οι **camerax**, **cameray**, **cameraz**. Αυτές χρησιμοποιούνται από την gluLookAt() ώστε να μπορεί ο χρήστης να μετακινήσει την κάμερα στον χώρο ώστε να δει την σκηνή από μια άλλη γωνία. Συγκεκριμένα, με τα βελάκια (πλήκτρα) στη συνάρτηση **keyboardFunc**() αυξομειώνουμε τις **camerax** και **cameray**, και με τη ροδέλα του ποντικιού στη **mouseFunc**() αυξομειώνουμε την **cameraz** ωστε να μπορεί να κάνει zoom in και zoom out.

Οι πόντοι με κάθε μεταβολή τους τυπώνονται στο τερματικό καθώς και το πόσες κινήσεις έχουν μείνει. Υπήρξαν κάποιες δυσκολίες στο να τοποθετήσουμε το κείμενο μέσα στη σκηνή μας και το κάναμε με αυτό τον τρόπο.

Παράθεση και σύντομη περιγραφή των συναρτήσεων:

- `bool loadPngImage(char *name, int &outWidth, int &outHeight, bool &outHasAlpha, GLubyte **outData)`: φορτώνει και αρχικοποιεί ένα texture σε μία μεταβλητή τύπου GLubyte*.
- **void loadTexture(int type)**: φορτώνει ένα από τα τρία texture με βάση τον τύπο (κύβου) που παιρνει ως όρισμα.
- **void initializeTextures()**: Καλεί την loadPngImage για τα τρια texture που θέλουμε.
- **void initializeCubes()**: Αρχικοποιεί τους κύβους. Τους κάνει τύπου White και τους δίνει μοναδικό fakeColor.
- **void startGame()**: Καλείται με την επιλογη Start Game από το μενού. Δίνει τυχαία τύπους στους κύβους. Μηδενίζει τις κινήσεις που έγιναν και το score.
- **void drawCube(Cube *cb, bool texture)**: κανει Draw το κύβο που δέχεται ως ορισμα. Αν το bool texture ειναι true, προσδίδει και το texture που έχει φορτωθεί.
- **void drawCubeFakeColor(Cube *cb)**: κάνει Draw τον κύβο που δέχεται ως όρισμα με επιλεγμένο χρώμα το fakeColor του συγκεκριμένου κύβου.
- **void drawCubes()**: Για κάθε τύπο κύβου, φορτώνει το texture ή το χρώμα και καλεί την drawCube για κάθε έναν κύβο.
- **void drawCubesFake()**: Για κάθε κύβο, καλεί την drawCubeFakeColor.
- **static void renderFakeScene(void)**: Κατασκευάζει τη ψεύτικη σκηνή (με τα fakeColors), απενεργοποιεί το φωτισμό (για να μην έχουμε αλλοίωση με την αναγνώρηση του χρώματος), αλλά δεν καλεί glFlush ή glutSwapBuffers ώστε να μην εμφανιστεί η σκηνή.
- **static void renderScene(void)**: Κατασκευάζει την πραγματική σκηνή που ο χρήστης βλέπει.
- **bool areNeighbours(Cube cube1, Cube cube2)**: Επιστρέφει true/false για το αν οι δύο κύβοι που δέχεται ως όρισμα είναι γείτονες.
- **bool eats(Cube cube1, Cube cube2)**: Επιστρέφει true/false για το αν ο cube1 “τρώει” τον cube2 (ορίσματα).
- **void destroyEatingNeighbours(Cube cube)**: Καταστρέφει (κάνει τύπου None) τους γείτονες τους οποίους “τρώει”.
- **void doArea1(bool isHorizontal, Cube middle)**: Εκτελεί τον έλεγχο και τις ενέργειες που απαιτούνται στην περιοχή 1.
- **void doArea2and3(bool isHorizontal**, Cube **middle)**: Εκτελεί τον έλεγχο και τις ενέργειες που απαιτούνται στις περιοχές 2 και 3.
- **void destroyTriad(Cube *cube1, Cube *cube2, Cube *cube3)**: Καταστρέφει την τριάδα κύβων που παίρνει ως όρισμα και εκτελεί τον έλεγχο για τις περιοχές 1, 2, 3.
- **void checkTriad(Cube cube)**: Ελέγχει αν ο κύβος που δέχεται ως όρισμα ανήκει σε κάποια τριάδα ίδιων τύπων.
- **void swapCubes(Cube *cube1, Cube *cube2)**: Εκτελεί το swapping στους δύο κύβους που δέχεται ως όρισμα. (swap των τύπων τους)
- **void checkAndSwapCubes(Cube *cube1, Cube *cube2)**: Ελέγχει για το αν οι δύο κύβοι που δέχεται ως όρισμα συνορεύουν και αν ναι καλεί την swapCubes. Επίσης, καλεί τον έλεγχο τριάδας ή τον έλεγχο destroyEatingNeighbours για τις αντίστοιχες περιπτώσεις.

- **void keyboardFunc(int key, int x, int y)**: Βασική συνάρτηση πληκτρολογίου. Κατά την εισαγωγή από κάποιο από τα βελάκια μεταβάλει τη θέση της κάμερας.
- **void menuOptions(int i)**: Έλεγχος και εκτέλεση κάποιας από τις ενέργειες του menu.
- **void initializeMenu(void)**: Αρχικοποίηση του menu επιλογών με το δεξί κλικ.
- **void mouseFunc(int btn, int state, int x, int y)**: Βασική συνάρτηση ποντικιού. Ελέγχει για το αν πατήθηκε κάποιος κύβος και εκτελεί την αντίστοιχη ενέργεια.
- **void initializeScene()**: Αρχικοποίηση της 3δ σκηνής.

Παρατήρηση:

Κατά την εκτέλεση του προγράμματος στον υπολογιστή στον οποίο αναπτύχθηκε όλα κυλούν ομαλά. Στη δοκιμή που έγινε σε έναν από τους υπολογιστές του εργαστηρίου παρατηρήσαμε ότι η σκηνή “τρεμοπαίζει”.
Είχαμε την **renderScene,** δηλαδή τη βασική συνάρτηση σκηνής μας, ορισμένη και ως **glutDisplayFunc** αλλά και ως **glutIdleFunc**, συνεπώς καλείται συνέχεια και δεν προλαβαίνουν να γίνουν render τα πάντα σε κάθε frame. Σε αυτό ίσως οφείλεται το φόρτωμα των textures που ίσως καθυστερεί παραπάνω.
Για να αντιμετωπιστεί αυτό, σταματήσαμε να ορίζουμε την **renderScene** ως idle function και την καλούμε κάθε φορά που συμβαίνει μια αλλαγή στη σκηνή μας (μετακίνηση κύβου, καταστροφή τριάδας, μετακίνηση κάμερας κλπ.).
Επίσης, όταν πατιέται το κλικ και ψευτο-ζωγραφίζεται η ψεύτικη σκηνή, ενώ δεν καλούμε **glFlush**() η **glutSwapBuffers**(), η σκηνή τρεμοπαίζει για ένα frame. Αυτό δεν συνέβαινε στον υπολογιστή που αναπτύχθηκε αλλά στον υπολογιστή του εργαστηρίου συμβαίνει.