ΠΡΩΤΗ ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΑΣΚΗΣΗ
ΓΡΑΦΙΚΑ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΑ ΑΛΛΗΛΕΠΙΔΡΑΣΗΣ

Πανεπιστήμιο Ιωαννίνων - Τμήμα Μηχ. Η/Υ και Πληροφορικής

2019-2020

ΒραΧαΨα III

Δερέκας Παναγιώτης ΑΜ: 4014
Θωμά Αθανάσιος ΑΜ: 2979

Ιδιαιτερότητες αλληλεπίδρασης με τον χρήστη:

- Για να τρέξουμε το παιχνίδι εκτελούμε στο τερματικό
  make
  και μετά ./vraxaya
  με make clean, καθαρίζουμε τον φάκελο από τα εκτελέσιμα.
- Οι κανόνες του παιχνιδιού, όπως περιγράφονται στην εκφώνηση.
- Για να ξεκινήσει το παιχνίδι, δεξί κλικ και την επιλογή Start Game.
- Για να μετακινηθεί η κάμερα, τα βελάκια του πληκτρολογίου.

Αρχιτεκτονική του προγράμματος:

τα αρχεία που πραγματοποιούν την λειτουργία του παιχνιδιού είναι:

- interfacetest.cpp :
  εδώ βρίσκονται η main, οι περισσότερες αρχικοποιήσεις, η συνάρτηση που κατασκευάζει και εμφανίζει την σκηνή (display()), η διαχείριση μενού, πληκτρολογίου και ποντικιού.
- GManager.cpp :
  αυτή η κλάση έχει ρόλο διαχειριστή της λειτουργίας του παιχνιδιού. Κρατάει στην μνήμη έναν πίνακα από κύβους και ανάλογα με τι θα ζητήσει ο χρήστης, ανταποκρίνεται.
- GCube.cpp :
  αυτή η κλάση έχει ρόλο τον εύκολο χειρισμό κύβων, ως αντικείμενα με τιμή. Συνεργάζεται με άλλες κλάσεις/αρχεία που φορτώνουν εικόνες για να κατασκευάσει όμορφα γραφικά (textures).

~ Να σημειώσουμε ότι τα περισσότερα conventions της γλώσσας C++ δεν τηρούνται, μιας και αυτή είναι η πρώτη μας επαφή με την γλώσσα.
Προσπαθήσαμε όμως το πρόγραμμά μας να ακολουθεί τεχνικές αντικειμενοστρέφειας για να είναι πιο κατανοητό και επεκτάσιμο.

Εξηγήσεις για ιδιαίτερα τμήματα του προγράμματος:

- Textures : η κλάση TextureLoader είναι υπεύθυνη για να φορτώνει από τα αρχεία .png τα textures. Έχουμε μόνο ένα αντικείμενο, τον singletonLoader για να μην καθυστερούμε ούτε σε χρόνο αλλά ούτε και σε μνήμη. Εξασφαλίζουμε ότι κάθε εικόνα θα φορτωθεί μία φορά μέσα στον constructor και χρησιμοποιούμε την δημόσια βιβλιοθήκη stb_image για εύκολη προσπέλαση εικόνων από το δίσκο.
  Οι βασικές συναρτήσεις που χρησιμοποιεί ο TextureLoader είναι αυτές του δοσμένου παραδείγματος Textures.zip από το φροντιστήριο 11/11/2019. Είναι κατάλληλα προσαρμοσμένες για να ανταποκρίνονται στο δικό μας πρόγραμμα και για εύκολη επέκταση του προγράμματος.

- Camera: Γίνεται με τη χρήση της gluLookAt. Οι δύο πρώτες μεταβλητές της συνάρτησης αλλάζουν, ώστε η κάμερα να κουνίεται μόνο στο επίπεδο (x,y). Έχει σταθερή την τρίτη μεταβλητή (z=5), και κατεύθυνση την (0,0,0). Η κίνηση γίνεται με τη χρήση των βελακίων ως εξής:
  1. το πάνω μετακινεί την θέση της κάμερας κατά +1 στον y.
  2. το κάτω μετακινεί την θέση της κάμερας κατά -1 στον y.
  3. το αριστερά μετακινεί την θέση της κάμερας κατά -1 στον x.
  4. το δεξιά μετακινεί την θέση της κάμερας κατά +1 στον x.

- drawText: Η δημιουργία της συνάρτησης στηρίχθηκε στη σελίδα https://videogameslab.wordpress.com/category/opengl/ από το παράδειγμα 5. Παίρνει 3 ορίσματα. Τα x,y είναι η θέση του πρώτου γράμματος (pixel) στην οθόνη και το τρίτο είναι δείκτης σε πίνακα από char (το κείμενό μας).
  Στην αρχή γίνεται απενεργοποίηση του φωτισμού για να μην επιρεάζεται ο χρωματισμός των λέξεων. Έπειτα γίνεται προβολή του κειμένου στις δυο διαστάσεις έτσι ώστε να μην επιρεάζεται το κείμενο από την προοπτική της κάμερας.
  Αφού γίνει η τύπωση του κειμένου ενεργοποιούμε πάλι το φωτισμό.

- Reshape: Η δημιουργία της συνάρτησης στηρίχθηκε στη σελίδα http://www.swiftless.com/opengltuts.html από το δεύτερο tutorial. Η συνάρτηση παίρνει δυο ορίσματα ύψος και το πλάτος του παραθύρου.
  Αρχικά καλούμε την glViewPort με τα δυο πρώτα ορίσματα (0,0) για την κάτω αριστερή γωνιά και για την πάνω δεξιά τις άλλες δυο ((Glsizei)width, (Glsizei)height).
  Έπειτα χρησιμοποιούμε την glMatrixMode(GL_PROJECTION) για να σετάρουμε την σκηνή μας στο νέο παράθυρο.
  Μετά καλούμε την glLoadIdentity() για να καθαρίσουμε τα σκουπίδια από τη προηγούμενη κατασκευή της σκηνής.
  Θέτουμε την προοπτική να έχει άνοιγμα 45 μοίρες και να διατηρεί τις αναλογίες των σχημάτων.
  Τέλος αλλάζουμε πάλι πίσω με την glMatrixMode(GL_MODELVIEW) για να μπορούμε να σχηματίσουμε σχήματα σωστά.

- Ποντίκι : οι συναρτήσεις που σχετίζονται με το ποντίκι είναι οι
  - playerMove: καλείται σε κάθε αριστερό κλικ και με τη βοήθεια μιας boolean μεταβλητής, αναγνωρίζει αν το κλικ είναι το πρώτο ή το δεύτερο του παίκτη. Αν είναι το δεύτερο, στέλνει τους δύο κύβους στον Gmanager που αναλαμβάνει την διαχείριση της κίνησης .
  - PlayerMoveOnClick: δέχεται ως ορίσματα τις συντεταγμένες του ποντικιού και βρίσκει αν ο χρήστης πάτησε πάνω σε κύβο ή όχι. Επιστρέφει τη σχετική θέση του κύβου (board[i][j]).
    Ο αλγόριθμος επιλογής κύβου είναι:
    - Για όλους τους κύβους ,
    - Κάνε προβολή τις συντεταγμένες του ποντικιού από το παράθυρο θέασης στον χώρο των αντικειμένων. (χρήση της gluUnProject) και όρισε μία ακτίνα.
    - Βρες την απόσταση του κέντρου του κύβου από την ακτίνα.
    - Αν η απόσταση αυτή είναι μικρότερη του $(\frac{(\mu\varepsilon\gamma\acute{\varepsilon}\theta ους\ του\ κύβου)}{2})*\sqrt{2}$, επέστρεψε τον κύβο.

  - Για την υλοποίηση αυτού του αλγορίθμου, χρήσιμη θα ήταν η βιβλιοθήκη glm που υποστηρίζει διανύσματα. Επιλέξαμε να ορίσουμε τα διανύσματα του τρισδιάστατου χώρου με πίνακες μεγέθους 3.
  - Ο αλγόριθμος αυτός είναι εμπνευσμένος από το post https://stackoverflow.com/questions/16962891/mouse-picking-in-opengl-using-gluunproject

---

Δεν υλοποιήσαμε:

- την αναπλήρωση κατεστραμμένων κύβων, μετά από έκρηξη όμοιας τριάδας. Σε αυτό το σημείο η εκφώνηση είναι δυσνόητη, αφού υπονοεί αλληλεπίδραση με τον χρήστη : *“Όταν καταστρέφονται κυβάκια ο παίκτης μπορεί να μετακινήσει άλλα γειτονικά κυβάκια στη θέση αυτή.”*
  Ένας απλός τρόπος υλοποίησης αυτού του μηχανισμού είναι μετά από κάθε έκρηξη να καλείται μία συνάρτηση που:
  - Για όλους τους κύβους:
    - Αν ο κύβος έχει τιμή κενή (αποτέλεσμα της καταστροφής)
    - δώσε μια τυχαία τιμή στον κύβο, μέσω γεννήτριας rand().

  Ο λόγος όμως που επιλέξαμε να μην το υλοποιήσουμε είναι για να φαίνεται πιο εύκολα το αποτέλεσμα της έκρηξης.
  Αν πάρουμε κατά γράμμα την εκφώνηση, τότε το πρόγραμμά μας καλύπτει και αυτή την επιλογή. Δηλαδή, μετά από μια έκρηξη οι κύβοι δεν “χάνονται”, απλά χρωματίζονται όπως οι αρχικοί κύβοι (μπεζ ανοιχτό) και λειτουργούν κανονικά σε αλληλεπιδράσεις όπως η ανταλλαγή με τους γειτονικούς.

Bonus:

- υλοποίηση κουτιών βομβών (TNT). Τα κουτιά αυτά έχουν μικρότερη πιθανότητα να εμφανιστούν. Τα κουτιά αυτά καταστρέφονται μόνο αν βρίσκονται σε απόσταση 1 από μια “έκρηξη”. (Θεωρήσαμε ότι “έκρηξη” εμφανίζεται μόνο όταν η τριάδα κύβων αποτελείται από ΒραΧαΨα κύβους. Δηλαδή μια βόμβα δεν θα καταστραφεί αν σχηματιστεί τριάδα από μπλε ή κόκκινους κύβους δίπλα της.)
- η επιλογή και ανταλλαγή κύβων μέσω ποντικιού, λειτουργεί και με την κάμερα σε διαφορετική θέση. Να σημειωθεί ότι η ευστοχία του παίκτη θα παίξει ρόλο για την επιλογή του επιθυμητού κύβου.
- drawOutsideBox : κατασκευάζει ένα καφέ κουτί γύρω από τους κύβους, ανοιχτό από επάνω. Αν δεν θέλετε να εμφανίζεται το κουτί απλά βάλτε σε σχόλιο την εντολή << drawOutsideBox() >> που βρίσκεται μέσα στην display().