Εργασία στον Προγραμματισμό Υπολογιστών με C++

Ακαδ. Έτος 2021-22

Επιτραπέζιο παιχνίδι

Εκφώνηση

Ο στόχος της εργασίας είναι να δημιουργήσετε τη δική σας ηλεκτρονική εκδοχή ενός από τα κλασικά επιτραπέζια παιχνίδια ή κάποιο παιχνίδι με κάρτες (ηλεκτρονικό ή μη), βασιζόμενοι στη βιβλιοθήκη Simple Graphics Library (SGG) που έχει φτιαχτεί για το μάθημα.

Μπορείτε να επιλέξετε κάποιο από τα πολύ κλασικά και απλά παιχνίδια, να τα επεκτείνετε με περισσότερες δυνατότητες, να βάλετε δυναμικά γραφικά και ήχο, δυνατότητα να παίξουν 2 παίκτες στον ίδιο υπολογιστή (pass and play) ή ό,τι άλλο φανταστείτε! Θυμηθείτε, η δημιουργικότητα ανταμείβεται. Οι ελάχιστοι περιορισμοί για ένα παιχνίδι είναι:

- Να απαιτεί παραπάνω από ένα είδος κάρτας ή πιονιού. Π.χ. το Othello ή το Σκορ 4 (Connect 4) δεν είναι επιλέξιμα.
- Να απαιτεί την αλληλεπίδραση του παίκτη με τον καμβά του παιχνιδιού μέσω του ποντικιού ή των πλήκτρων (επιλογή, μετακίνηση, κλπ.), δηλαδή να μην έχει προκαθορισμένη συμπεριφορά εξαρτώμενη μόνο από το αποτέλεσμα ενός ζαριού.
- Να εκτελείται σε διακριτά χρονικά βήματα μη πραγματικού χρόνου, είτε για έναν είτε για δύο ή περισσότερους παίκτες.

Ενδεικτικές κατηγορίες παιχνιδιών:

Ηλεκτρονικά **παιχνίδια αντιστοίχισης** ψηφίδων, όπως το Candy Crush.

Παιχνίδια στρατηγικής με κάρτες, όπως το Hearthstone και παραλλαγές του (και βέβαια το φετινό demo).

Απλά επιτραπέζια **παιχνίδια εξερεύνησης και ρόλων**, όπως το HeroQuest ή αντίστοιχα ηλεκτρονικά, όπως το ArcaneQuest.

Παιχνίδια στρατηγικής, όπως το Stratego που παίζεται σε ταμπλό με κελιά ή κάποιο άλλο skirmish game με ελεύθερη μετακίνηση στο ταμπλό.

Σημείωση: Και πολύ απλούστερα παιχνίδια μπορούν να υλοποιηθούν, αν τους αλλάξετε τους κανόνες. Για παράδειγμα, το φιδάκι (snakes and ladders) δεν είναι επιλέξιμο στη βασική του μορφή καθώς η έκβασή του εξαρτάται μόνο από το ζάρι. Αν όμως διαλέγατε να ελέγχετε εσείς τη θέση από τα φιδάκια και τις σκάλες αντί για μια κανονική κίνησή σας; Αυτό θα έβαζε μια διαφορετική, πιο διαδραστική διάσταση στο παιχνίδι! Επίσης, σας παροτρύνουμε να εμπνευσθείτε από υπάρχοντα παιχνίδια και να δημιουργήσετε δικά σας, αντλώντας στοιχεία και ιδέες από υπάρχοντα παιχνίδια και φτιάχνοντας κάτι νέο.

Υλοποίηση

Κατά την υλοποίηση της εφαρμογής σας, καλείστε να συνδυάσετε γνώσεις που αποκομίσατε από τις διαλέξεις και να σκεφτείτε καλά την αρχιτεκτονική του κώδικά σας, προκειμένου να πετύχετε α) καλή επαναχρησιμοποίηση κώδικα, β) ενιαίο και πολυμορφικό τρόπο κλήσης μεθόδων, δ) αποδοτική εκμετάλλευση έτοιμων δομών της STL, γ) ταχύτητα.

Οι παρακάτω στόχοι είναι υποχρεωτικοί και βαθμολογούνται:

- **Χρήση της βιβλιοθήκης SGG.** Η ενσωμάτωση της βιβλιοθήκης SGG και χρήση των συναρτήσεων που παρέχονται είναι υποχρεωτική και αποκλειστική. Η εφαρμογή σας δε θα πρέπει να βασίζεται σε άλλη εξωτερική βιβλιοθήκη για τη διαχείριση του παραθύρου και των συμβάντων πληκτρολογίου και ποντικιού, τη σχεδίαση γραφικών και την αναπαραγωγή ήχου.

- **Χρήση δυναμικής μνήμης**. Στα παιχνίδια, πολλές οντότητες (assets) δημιουργούνται και «ζουν» για ένα περιορισμένο διάστημα κατά την εκτέλεση του κώδικα. Τέτοια παραδείγματα είναι πιόνια, στιγμιαία «εφέ» (λάμψεις, βελάκια και άλλες ενδείξεις, animations, κλπ.) ή άλλα δυναμικά στοιχεία του παιχνιδιού. Τέτοια στοιχεία πρέπει να δημιουργούνται δυναμικά στη μνήμη (με new ή malloc) και να καταστρέφονται όταν δε χρειάζονται.

- **Κληρονομικότητα και πολυμορφισμός**. Τα διάφορα στοιχεία του παιχνιδιού αυθόρμητα έχουν μια εσωτερική οντολογική ιεραρχική δομή. Για παράδειγμα, ενδεικτικά, οτιδήποτε αναπαράγεται με οποιονδήποτε τρόπο κατά την εκτέλεση του παιχνιδιού είναι ένα “asset”. Μπορούμε να έχουμε assets που «σχεδιάζονται» ή που «ακούγονται», όπως ένα “bitmap” (“sprite”) ή ένα “sound” αντικείμενο.
  Σε πιο υψηλό λειτουργικό επίπεδο, διαθέτουμε οντότητες “GameObject” που κρατάνε και ενημερώνουν τη δική τους κατάσταση, αλληλεπιδρούν με άλλες και σχεδιάζονται ή αναπαράγουν έναν ήχο. Αυτές τυπικά διαθέτουν κάποια μέθοδο “draw” και “update” που θα πρέπει να καλέσει η βασική λογική του παιχνιδιού μας σε ένα βρόγχο, όσο τρέχει. Από ένα GameObject, μπορώ να εξειδικεύσω οντότητες τύπου «κάρτα», «ζάρι», «πιόνι», «στατικού αντικειμένου» (περιβάλλοντος, background Κλπ.), “UI widget” κλπ.
  Στην εργασία σας καλείστε να οργανώσετε τις κλάσεις σας με έναν παρόμοιο ιεραρχικό τρόπο και να χρησιμοποιήσετε πολυμορφισμό για την κλήση μεθόδων στιγμιοτύπων αυτών των κλάσεων.

- **Συλλογές**. Σε ένα παιχνίδι, κατασκευάζουμε, αποθηκεύουμε και διαχειριζόμαστε μια πολλαπλότητα από αντικείμενα, είτε για τη λειτουργία του προγράμματος, είτε για τη σχεδίαση των γραφικών στην οθόνη. Καλείστε να χρησιμοποιήσετε τις καταλληλότερες για τη δουλειά που τις χρειάζεστε συλλογές της STL για τις ανάγκες αποθήκευσης, αναζήτησης και μαζικής εκτέλεσης μεθόδων. Προσοχή: για να δουλέψουν ορισμένες από τις παρεχόμενες συλλογές σωστά με δικές σας κλάσεις, θα πρέπει να προσδιορίσετε τους κατάλληλους τελεστές για την ταξινόμηση ή το hashing των αντικειμένων (βλ. διαφάνειες μαθήματος). Συστήνεται αυστηρά να μην υλοποιήσετε δικές σας συλλογές για πράγματα που ήδη σας παρέχει η STL.

Προαιρετικά χαρακτηριστικά. Θα εκτιμηθεί θετικά ο σωστός σχεδιασμός και δόμηση του κώδικα, η σχολαστική δήλωση μεθόδων (π.χ. σωστή χρήση αναφορών και const ορισμάτων ή μεθόδων), η εκμετάλλευση templated συναρτήσεων ή κλάσεων, όπου φαίνεται χρήσιμο. Υπενθυμίζεται ότι ορισμένα μονοπάτια κώδικα που εκτελούν ενδεχομένως βαριές διαδικασίες υπολογισμών μπορούν να εκτελεστούν σε ξεχωριστό(ά) thread(s)[1].

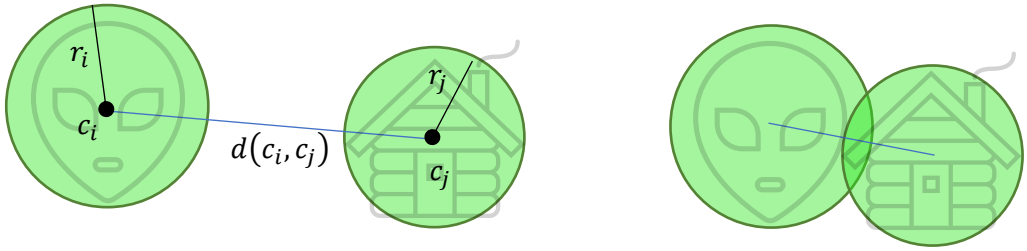
Συγκρούσεις. Αναπόφευκτα, ένα παιχνίδι στο οποίο πολλαπλές οντότητες επικαλύπτονται, κινούνται και πρέπει να ενεργοποιηθούν λειτουργίες όταν ένα στοιχείο του παιχνιδιού το ακουμπήσω πάνω σε κάποιο άλλο, πρέπει να ελέγχει για συγκρούσεις (collisions) μεταξύ ορισμένων από τα στοιχεία αυτά. Τέτοια παραδείγματα είναι η μετακίνηση μια κάρτας πάνω από κάποια άλλη, η τοποθέτηση ενός πιονιού σε συγκεκριμένη θέση στο ταμπλό ή η αποφυγή επικάλυψης μεταξύ δύο πιονιών. Η διαδικασία ελέγχου γίνεται αρκετά απλά αν θεωρήσουμε ότι για κάθε οντότητα i που διαθέτει δυνατότητα ανίχνευσης σύγκρουσης

[1] ΜΗΝ το κάνετε για τη σχεδίαση ή οτιδήποτε έχει να κάνει με γραφικά και context παραθύρου, αυτά πρέπει να καλούνται από το κύριο thread της εφαρμογής.

(collision detection) έχει προσδιοριστεί ένας κύκλος ακτίνας r_i μέσα στον οποίο αν βρεθεί η αντίστοιχη «ζώνη επιρροής» με ακτίνα r_j κάποιου άλλου στοιχείου j τότε αυτό πρέπει να σημάνει μια σύγκρουση και να ενεργοποιήσει μια αντίδραση σε αυτή. Ο έλεγχος για το αν δύο οντότητες συγκρούστηκαν μπορεί τότε να γίνει εύκολα με βάση τη συνθήκη:

$$d(c_i, c_j) - r_i - r_j < 0,$$

όπου c_i, c_j τα κέντρα των δύο κύκλων και $d(\alpha, \beta)$ η Ευκλείδεια απόσταση μεταξύ τους.



Ομάδες

Οι εργασία παραδίδεται από ομάδες 1-2 ατόμων. Σε περίπτωση που μια ομάδα επιλέξει να υλοποιήσει ένα αρκετά πιο σύνθετο παιχνίδι (μετά από συνεννόηση με το διδάσκοντα), τότε μπορεί να επεκταθεί μέχρι τα 3 μέλη. Δεν εμποδίζει τίποτα προφανώς μικρότερες ομάδες να αναλάβουν και να παραδώσουν μια πιο σύνθετη εργασία (με την ανάλογη επιβράβευση).

Σε κάθε περίπτωση, όλα τα μέλη της ομάδας θα πρέπει να έχουν ασχοληθεί με κάποια λειτουργικότητα της εφαρμογής και να έχουν συντελέσει στη συγγραφή του κώδικα. Δηλαδή δεν επιτρέπεται να επιμεριστεί ο φόρτος μεταξύ των μελών ώστε κάποιος να επιμεληθεί μόνο των εικαστικών ή των ήχων.

Οπτικοακουστικό Υλικό

Τα παιχνίδια που σας προτείνουμε να υλοποιήσετε μπορούν να υλοποιηθούν και με πολύ βασικά γραφικά και τις σχεδιαστικές δυνατότητες της SGG, οπότε είτε κατά τα αρχικά στάδια της υλοποίησής σας (που κυρίως ελέγχετε λειτουργικότητα) είτε αν δεν έχετε δυνατότητα ή χρόνο να ασχοληθείτε με την «παρουσίαση» του περιεχομένου, μπορείτε να χρησιμοποιήσετε τα έτοιμα primitives σχεδίασης που σας παρέχει η SGG για να δείξετε απλές μορφές και σχήματα στην οθόνη. Δε βαθμολογήστε αρνητικά για την αισθητική της εφαρμογής.

Αν πάλι θέλετε να βάλετε δικά σας στοιχεία ή έτοιμα bitmaps που κατεβάσατε από το διαδίκτυο, ο μορφότυπος PNG για τη φόρτωση και σχεδίαση εικόνων πάνω στα βασικά primitives είναι υπερ-αρκετός για τις ανάγκες σας, αφού υποστηρίζει και κανάλι διαφάνειας και όλα τα δημοφιλή και ελεύθερα προγράμματα επεξεργασίας και μετατροπής εικόνων το υποστηρίζουν. Τα αρχεία σας φέρτε τα στο κατάλληλο μέγεθος που να είναι συμβατό με τις ανάγκες της εφαρμογής. Για παράδειγμα, αν θέλετε να φορτώσετε ως background μια εικόνα που βρήκατε ανάλυσης 4400X3300 pixels, αυτή θα είναι πολύ μεγάλη, ξοδεύοντας άσκοπα α) μνήμη, β) χρόνο ανοίγματος (ειδικά σε debug mode), γ) χώρο στο δίσκο και στο zip που θα φτιάξετε στο τέλος (βλ. παράδοση εργασιών). Με κάποιο πρόγραμμα επεξεργασίας εικόνων, φέρτε τη σε διαστάσεις κατάλληλες για το παράθυρό σας. Π.χ. για ένα 1024X768 παράθυρο, στο παράδειγμά μας καλή είναι και η μετατροπή της εικόνας σε 1067X800, δηλαδή να υπερκαλύπτει την αναμενόμενη ανάλυση παραθύρου, διατηρώντας το λόγο πλάτους ύψους της αρχικής εικόνας. Σημείωση: αν έχετε φέρει τις εικόνες σας σε διαστάσεις που να είναι δυνάμεις του 2 (π.χ. 1024X512, 64X64), τότε η φόρτωσή τους από τη βιβλιοθήκη είναι γρηγορότερη, καθώς διαφορετικά πρέπει να τις μετατρέψει στις πλησιέστερες δυνάμεις του 2 η συνάρτηση φορτώματος.

Αντίπαλοι και Τεχνητή Νοημοσύνη

Στην υλοποίησή σας δε συνιστάται να χρησιμοποιήσετε κάποιο είδος τεχνητής νοημοσύνης, καθώς δεν απαιτείται. Οι κατηγορίες των παιχνιδιών που σας προτείνουμε παίζονται με 2 παίκτες εναλλάξ ή με έναν.

Παράδοση Εργασιών

Οι εργασία σας θα πρέπει να ανέβει στο eclass από ένα από τα μέλη της ομάδας σαν ένα zip αρχείο που θα πρέπει να περιλαμβάνει:

- Τον κώδικα της εργασίας. Προσοχή, από τη βιβλιοθήκη SGG να έχετε μόνο τα 2 απαραίτητα header files για τη μεταγλώττιση του δικού σας προγράμματος (graphics.h, scancodes.h), όχι όλο τον κώδικα της βιβλιοθήκης!
- Μην ανεβάσετε τη βιβλιοθήκη SGG που χτίσατε.
- Τα assets που χρησιμοποιεί η εφαρμογή σας στους κατάλληλους φακέλους έτσι ώστε το εκτελέσιμο που χτίζεται να μπορεί να τα βρει κατά την εκτέλεσή του.
- Τα απαραίτητα αρχεία για το χτίσιμο του κώδικα της εργασίας, π.χ. το solution (.sln) και Project file (.vcxproj) στους κατάλληλους υποφακέλους, αν χρειάζεται, ή κάποιο makefile ή κάποιο build script.

Δε χρειάζεται να έχετε ανεβασμένα τα αρχεία των βιβλιοθηκών δυναμικής σύνδεσης (DLLs. SOs).

Προσοχή: Πριν ανεβάσετε την εργασία σας, αντιγράψτε τη σε ένα χωριστό φάκελο και διαγράψτε από εκεί όλα τα προσωρινά αρχεία που δημιουργούνται κατά το χτίσιμο της εφαρμογής και τα οποία ενδέχεται (ειδικά για την περίπτωση του visual studio) να είναι αρκετά μεγάλα. Τέτοια είναι τα *.pdb, *.tmp, *.obj, *.ilk, καθώς και ο κρυφός φάκελος .vs.

Στο όνομα του αρχείου zip πρέπει να περιλαμβάνονται οι αριθμοί μητρώου όλων των μελών της ομάδας.

Η καταληκτική ημερομηνία και ώρα παράδοσης της εργασίας είναι 23/1/2021, 23:00. Δε θα δοθεί καμία παράταση, καθώς θα ξεκινήσει την εβδομάδα που ακολουθεί η εξέταση της εργασίας.