Εργασία στον Προγραμματισμό Υπολογιστών με C++

Ακαδ. Έτος 2020-21

Arcade Game

Εκφώνηση

Ο στόχος της εργασίας είναι να δημιουργήσετε τη δική σας εκδοχή από ένα από τα κλασικά παιχνίδια arcade που έφεραν την επανάσταση στην ηλεκτρονική ψυχαγωγία πριν από αρκετές δεκαετίες, βασιζόμενοι στη βιβλιοθήκη Simple Graphics Library (SGG) που έχει φτιαχτεί για το μάθημα.

Μπορείτε να επιλέξετε κάποιο από τα πολύ κλασικά και απλά παιχνίδια Pong, Space Invaders ή Asteroids (βλ. αναφορές [1,2,3] παρακάτω), να τα επεκτείνετε με περισσότερες δυνατότητες, καλύτερα γραφικά και ήχο, power-ups, δυνατότητα να παίξουν 2 παίκτες στον ίδιο υπολογιστή (με άλλα πλήκτρα του πληκτρολογίου) ή ό,τι άλλο φανταστείτε! Θυμηθείτε, η δημιουργικότητα ανταμείβεται.

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 «βολές» του παίκτη ή αντιπάλων, οι εχθρικές μονάδες, εφέ όπως εκρήξεις κλπ. 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 «παίχτη», «βολής», «στατικού αντικειμένου» (περιβάλλοντος, background Κλπ.), «εχθρού», “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.

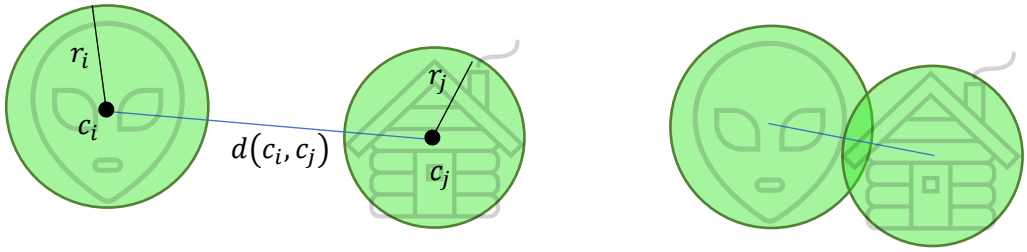
- **Συλλογές**. Σε ένα παιχνίδι, κατασκευάζουμε, αποθηκεύουμε και διαχειριζόμαστε μια πολλαπλότητα από αντικείμενα, είτε για τη λειτουργία του προγράμματος, είτε για τη σχεδίαση των γραφικών στην οθόνη. Καλείστε να χρησιμοποιήσετε τις καταλληλότερες για τη δουλειά που τις χρειάζεστε συλλογές της STL 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].

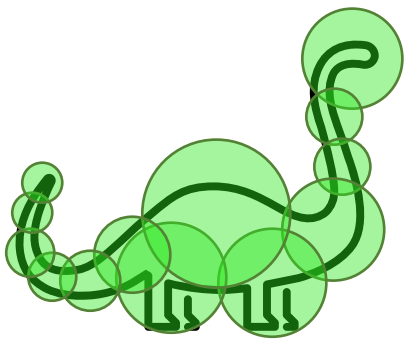
Συγκρούσεις. Αναπόφευκτα, ένα παιχνίδι στο οποίο πολλαπλές οντότητες κινούνται, πρέπει να ελέγχει για συγκρούσεις (collisions) μεταξύ ορισμένων από τα στοιχεία αυτά. Για παράδειγμα, όταν μια βολή φεύγει από τον παίχτη, πρέπει να ελέγχεται σε κάθε καρέ αν με βάση την τρέχουσα θέση της κίνησής της, έχει βρει το στόχο της. Αυτό γίνεται αρκετά απλά αν θεωρήσουμε ότι για κάθε οντότητα i που διαθέτει δυνατότητα ανίχνευσης σύγκρουσης (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.



Αν έχω κάποια αρκετά σύνθετα σχήματα, μπορώ να ορίσω πολλαπλούς κύκλους σύγκρουσης για αυτά, έτσι ώστε μικραίνοντας τις ακτίνες αυτών, να περιγράψω καλύτερα τα όρια του αντικειμένου:



[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.

Ένα αντικείμενο που περιλαμβάνει πολλαπλές περιοχές (κύκλους εδώ) σύγκρουσης, συγκρούεται με κάποιο άλλο αν έστω και μια περιοχή του βρει σύγκρουση με κάποια αντίστοιχη περιοχή σύγκρουσης του άλλου αντικειμένου.

Ομάδες

Οι εργασία παραδίδεται από ομάδες 1-2 ατόμων. Σε περίπτωση που μια ομάδα επιλέξει να υλοποιήσει ένα αρκετά πιο σύνθετο παιχνίδι (βλ. *hardcore mode* παρακάτω), τότε μπορεί να επεκταθεί μέχρι τα 3 μέλη, αλλά μπορεί να θέλει να αναλάβει ακόμα και ένα άτομο μια τέτοια εργασία.

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 από μόνα τους δεν έχουν ιδιαίτερες απαιτήσεις σε γραφικά ή ήχο, 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.

Hardcore Mode

Όσες ομάδες το επιθυμούν, μπορούν να επιλέξουν κάποιο παιχνίδι με πιο σύνθετη λειτουργία, κίνηση και προφανώς, υλοποίηση! Για παράδειγμα, παιχνίδια με μεγαλύτερο βαθμό δυσκολίας είναι scrolling shoot-em ups όπως το κλασικό 1942 [4] (και οι παραλλαγές του, π.χ. 1943, Xenon 2 [5] κλπ.) ή side scrolling shooters τύπου Galaxian (π.χ. R-Type [6]), όπου το υπόβαθρο κινείται μαζί με το αντικείμενο του παίχτη και οι «εχθροί» εμφανίζουν πιο πολύπλοκες «συμπεριφορές» και βαθμούς ελευθερίας.

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 17/1/2021, 23:00. Δε θα δοθεί καμία παράταση, καθώς θα ξεκινήσει την εβδομάδα που ακολουθεί η εξέταση της εργασίας.

Αναφορές

[1] Pong https://www.youtube.com/watch?v=fiShX2pTz9A, https://en.wikipedia.org/wiki/Pong

[2] Space Invaders https://en.wikipedia.org/wiki/Space_Invaders, https://www.youtube.com/watch?v=MU4psw3ccUI

[3] Asteroids https://en.wikipedia.org/wiki/Asteroids_(video_game), https://www.youtube.com/watch?v=WYSupJ5r2zo

[4] 1942 https://en.wikipedia.org/wiki/1942_(video_game), https://www.youtube.com/watch?v=kdICR49vbvg

[5] Xenon 2 – Megablast https://en.wikipedia.org/wiki/Xenon_2:_Megablast, https://www.youtube.com/watch?v=v9nD9DQwd80

[6] R-type https://en.wikipedia.org/wiki/R-Type, https://www.youtube.com/watch?v=1vFOfJGl1hQ