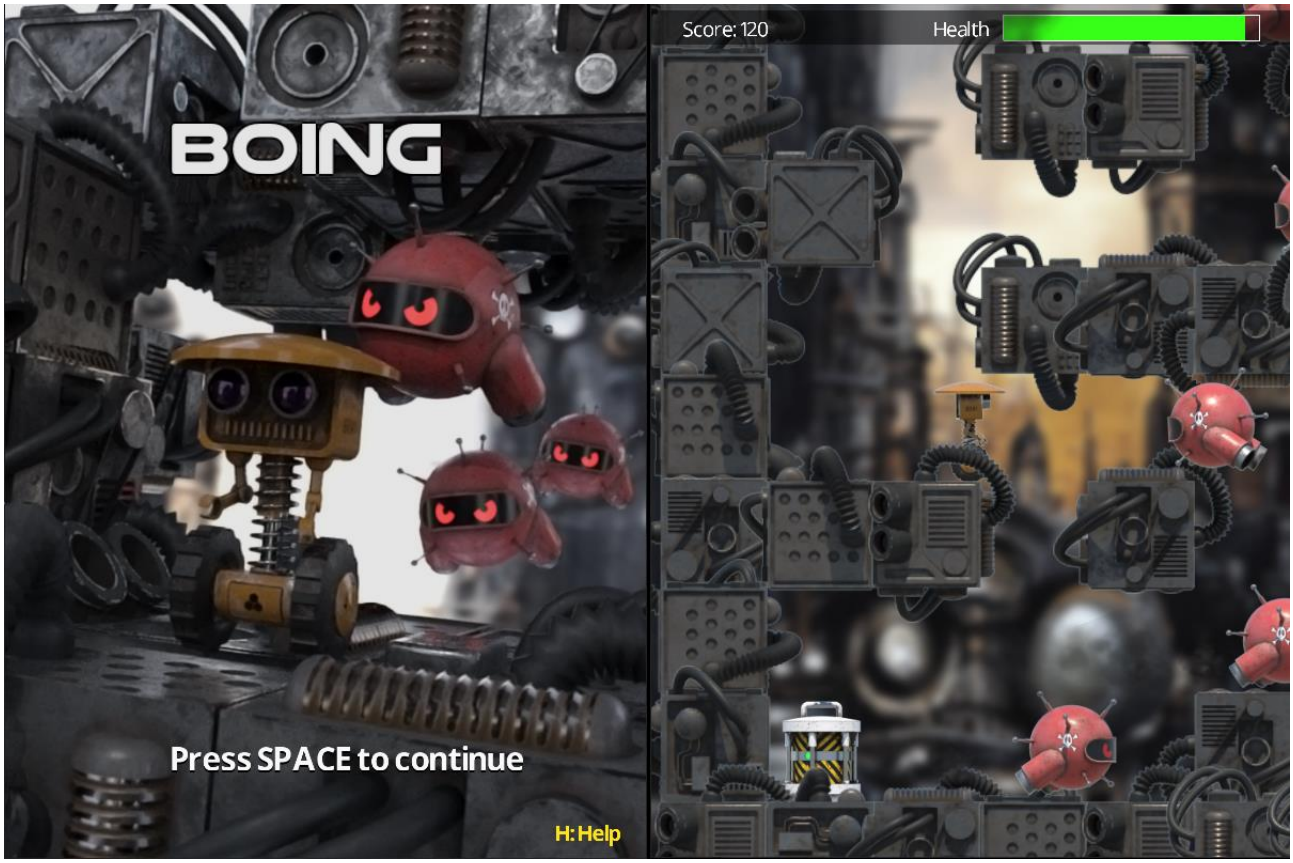
Εργασία στον Προγραμματισμό Υπολογιστών με C++

Ακαδ. Έτος 2023-24

Παιχνίδι πλατφόρμας



1. Περίληψη

Ο στόχος της εργασίας είναι να δημιουργήσετε ένα δικό σας παιχνίδι πλατφόρμας (platform game), βασιζόμενοι στη βιβλιοθήκη Simple Graphics Library (SGG) που έχει φτιαχτεί για το μάθημα.

Σε ένα παιχνίδι πλατφόρμας, τυπικά ελέγχετε έναν «χαρακτήρα» τον οποίο βοηθάτε να πλοηγηθεί μέσα σε μια «πίστα» με σκοπό να φτάσει σε ένα τερματικό σημείο, αποφεύγοντας κινούμενους και στατικούς εχθρούς και μαζεύοντας στη διαδρομή βοηθητικά στοιχεία και πόντους με διάφορους τρόπους. Μπορείτε να επιλέξετε να προσεγγίσετε τον τρόπο λειτουργίας κάποιου από τα πολύ κλασικά παιχνίδια του είδους, όπως το Super Mario, να τα επεκτείνετε με περισσότερες δυνατότητες, να βάλετε τη δυνατότητα να παίξουν 2 παίκτες στον ίδιο υπολογιστή, ή να δημιουργήσετε κάτι εντελώς καινούριο και πρωτότυπο. Θυμηθείτε, η δημιουργικότητα ανταμείβεται. Αν έχετε αμφιβολία για το αν κάποια ιδέα που σκεφτήκατε εντάσσεται στην κατηγορία παιχνιδιού της εργασίας, ρωτήστε.

2. 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.

2.1. Ζητούμενα

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 εχθροί, power-ups, στιγμιαία «εφέ» (λάμψεις, καπνός, διάφορες ενδείξεις, κλπ.) ή άλλα δυναμικά στοιχεία του παιχνιδιού. Τέτοια στοιχεία πρέπει να δημιουργούνται δυναμικά στη μνήμη (με new) και να καταστρέφονται όταν δε χρειάζονται.

- **Κληρονομικότητα και πολυμορφισμός**. Τα διάφορα στοιχεία του παιχνιδιού αυθόρμητα έχουν μια εσωτερική οντολογική ιεραρχική δομή. Σε λειτουργικό επίπεδο, διαθέτουμε οντότητες τύπου “GameObject” που κρατάνε και ενημερώνουν τη δική τους κατάσταση, και εξειδικευόμενες, παριστάνουν ένα είδος συγκεκριμένης λειτουργικής μονάδας του παιχνιδιού. 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 «παίχτης», «εχθρός», «εφέ», «πίστα» (level), «στατικό αντικείμενο» (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.

- Η χρήση μιας βασικής κλάσης (base class) με όνομα **GameObject** για όλων των ειδών τα εξειδικευμένα αντικείμενα είναι υποχρεωτική. Αυτή θα πρέπει να έχει τουλάχιστον τα παρακάτω δεδομένα και μεθόδους:

```
class GameObject
{
      static int m_next_id;

protected:
      class GameState* m_state;
      std::string m_name;
      int m_id = 0;
      bool m_active = true;

public:
      GameObject(GameState * gs, const std::string& name = "");
      virtual void update(float dt) {}
      virtual void init() {}
      virtual void draw() {}
      virtual ~GameObject() {}
      bool isActive() { return m_active; }
      void setActive(bool a) { m_active = a; }

};
```

Η μέθοδος update() θα πρέπει να καλείται σε κάθε κύκλο ενημέρωσης της κατάστασης της εφαρμογής (βλ. οδηγίες της βιβλιοθήκης SGG). Αντίστοιχα, η μέθοδος draw() καλείται για να σχεδιαστεί το στοιχείο. Η μέθοδος init() είναι υπεύθυνη να αρχικοποιήσει ένα στιγμιότυπο, μετά από την κατασκευή του, αν

χρειάζεται κάτι τέτοιο. Επίσης, μπορεί να κληθεί για να «ξανα-αρχικοποιήσει» ένα αντικείμενο του παιχνιδιού άμα κάνουμε κάποιο reset, επαναφέροντας την αρχική κατάστασή του, χωρίς να απαιτείται να ξαναδημιουργήσουμε το αντικείμενο.

Η κλάση **GameState** από την οποία θα πρέπει να έχετε ένα και μοναδικό στιγμιότυπο στην εφαρμογή σας:

α) φυλάει δεδομένα για τη ροή και την κατάσταση του παιχνιδιού (π.χ. σε πιο level είμαστε, τα διαθέσιμα levels, στατιστικά, σκορ, κλπ.)

β) διαθέτει τις βασικές μεθόδους draw, init και update που καλούν τις αντίστοιχες έμμεσα ή άμεσα για οποιοδήποτε αντικείμενο λειτουργεί μέσα στο παιχνίδι σας.

γ) είναι υπεύθυνη για το ξεκίνημα, την εναλλαγή επιπέδων (πιστών) και τον τερματισμό του παιχνιδιού.

δ) παρέχει ένα κεντρικό σημείο πρόσβασης για να εντοπιστούν στοιχεία του παιχνιδιού (π.χ. GameState::getPlayer())

2.2. 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.

2.3. 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]. Μπορείτε να φτιάξετε δυναμικό φόρτωμα πολλαπλών επιπέδων του παιχνιδιού, καθώς αυτό εξελίσσεται, να φορτώνετε οποιονδήποτε αριθμό από επίπεδα αποθηκευμένα σε αρχεία στο δίσκο (όπως το demo που σας δίνεται). Μπορείτε να υλοποιήσετε σύστημα δημιουργίας και διαχείρισης συμβάντων που να ελέγχονται από το GameState ή κάποιον κατανεμημένο μηχανισμό ανταλλαγής μηνυμάτων μεταξύ των οντοτήτων του παιχνιδιού. Για παράδειγμα, στο demo παιχνίδι που σας έχει δοθεί, παράγονται συνεχώς σύντομα εξειδικευμένα αντικείμενα ως events με περιορισμένο χρόνο ζωής τα οποία διαχειρίζεται και ελέγχει (πότε ξεκινούν, πόσο διαρκούν, πώς καταστρέφονται) το κεντρικό αντικείμενο τύπου GameState (π.χ. εφέ μετάβασης επιπέδων, λεζάντες συγκομιδής πόντων, σύννεφα καπνού).

2.4. Συγκρούσεις

Αναπόφευκτα, ένα παιχνίδι στο οποίο πολλαπλές οντότητες επικαλύπτονται, κινούνται και πρέπει να ενεργοποιηθούν λειτουργίες όταν ένα στοιχείο του παιχνιδιού επικαλυφθεί με κάποιο άλλο, πρέπει να ελέγχει για συγκρούσεις (collisions) μεταξύ ορισμένων από τα στοιχεία αυτά. Τέτοια παραδείγματα είναι ο περιορισμός της κίνησης πάνω στις πλατφόρμες και το σταμάτημα σε τοίχους, η σύγκρουση με εχθρούς και

[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. Αν έχετε πάρα πολλούς ελέγχους συγκρούσεων ή άλλους υπολογισμούς (π.χ. AI εχθρών), μπορείτε να τους εκτελέσετε παράλληλα με άλλες διαδικασίες.

η συγκομιδή powerups. Η διαδικασία ελέγχου γίνεται αρκετά απλά με μια συνάρτηση επικάλυψης μεταξύ παραλληλογράμμων (σας δίνεται).

3. Ομάδες

Οι εργασία παραδίδεται από ομάδες 1-2 ατόμων. Ο σχηματισμός ομάδων 3 ατόμων δεν ενθαρρύνεται. 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. Η πολυπλοκότητα του παιχνιδιού, θα πρέπει να συνεπάγεται και αντίστοιχη πολυπλοκότητα στο σχεδιασμό του κώδικα και την υλοποίηση. Ενδεικτικά, μια τέτοια υλοποίηση θα πρέπει να περιλαμβάνει αρκετά από τα προαιρετικά στοιχεία που αναφέρονται στην ενότητα 2.3. 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.

4. 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.

5. 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!

- 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. Προσοχή: αν χρησιμοποιείτε πολλά ή/και μεγάλα αρχεία εικόνας ή ήχου και αυξάνει πολύ το μέγεθος του zip σας (>4MB), προτιμήστε να ανεβάσετε χωριστά το φάκελο των asserts σε κάποιον εξωτερικό σύνδεσμο (που να μη λήγει – όχι π.χ. WeTransfer) και στο zip σας βάλτε ένα txt αρχείο που να περιγράφει α) τη διεύθυνση του εξωτερικού συνδέσμου, β) τον σχετικό κατάλογο ως προς το εκτελέσιμο που περιμένει η εφαρμογή να βρει τα δεδομένα αυτά. Π.χ. αν το εκτελέσιμο βρίσκεται στο φάκελο D:\game\bin\ και ψάχνει τα αρχεία δεδομένων στον κατάλογο D:\game\bin\assets, βάλτε στο txt αρχείο τον σχετικό κατάλογο .\assets.
- 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.

Μην ανεβάσετε:

- Τα αρχεία των βιβλιοθηκών δυναμικής σύνδεσης (DLLs. SOs).
- Τα .lib αρχεία της SGG. Τα έχουμε
- 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. Προσοχή: κάποια από αυτά είναι κρυφά, όπως ο κατάλογος .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 11/2/2024, 23:00. Δε θα δοθεί καμία παράταση. Η εξέταση της εργασίας θα ξεκινήσει αμέσως μετά.

6. Εξέταση και Βαθμολόγηση Εργασιών

Η εξέταση των εργασιών θα γίνει στα μέλη των ομάδων, μέσω MS Teams. Θα καταρτιστεί κατάλογος με τις υποβληθείσες εργασίες και την εβδομάδα που θα ακολουθήσει την καταληκτική ημερομηνία παράδοσης της εργασίας, θα σας καλέσουν οι βοηθοί του μαθήματος ανά ομάδα να παρουσιάσετε τη δουλειά σας. Κατά την εξέταση των εργασιών, τα μέλη των ομάδων θα εξεταστούν χωριστά πάνω στην εργασία και θα βαθμολογηθούν επίσης χωριστά.

Η βαθμολόγηση των εργασιών, με άριστα το 4, θα γίνει σύμφωνα με τα ακόλουθα κριτήρια:

Κριτήριο	**Επίπτωση**	**Σχόλιο**
Μη χρήση της βιβλιοθήκης SGG	Απόρριψη εργασίας	Η αποκλειστική χρήση της βιβλιοθήκης SGG είναι υποχρεωτική
Παράδοση εργασίας που δεν εμπίπτει θεματολογικά στην εκφώνηση	Απόρριψη εργασίας	Αν δεν είστε σίγουροι για το αν το θέμα που επιλέξατε είναι «Platform game» ή συναφές παιχνίδι, ρωτήστε.
Μη χρήση κληρονομικότητας	Έως -1,0	Βλ. ενότητα «Υλοποίηση»

Μη πολυμορφική κλήση κοινών μεθόδων	Έως -1,0	Βλ. ενότητα «Υλοποίηση»
Μη δήλωση και υλοποίηση κλάσης βάσης GameObject και κλάσης κατάστασης παιχνιδιού GameState	Έως -0,5	Βλ. ενότητα «Υλοποίηση»
Μη ικανοποιητικός σχεδιασμός εφαρμογής	Έως -0,5	Βλ. ενότητα «Υλοποίηση»
Μη λειτουργική διεπαφή με το χρήστη / έλεγχος ροής παιχνιδιού	Έως -0,5	-
Υλοποίηση επιθυμητών (μη υποχρεωτικών) χαρακτηριστικών	Έως +0,5	Βλ. ενότητα «Υλοποίηση»

Σημειώνεται ότι, με βάση τα παραπάνω, μια άρτια εργασία 2 ατόμων που έχει υλοποιημένα επιπρόσθετα χαρακτηριστικά από τα υποχρεωτικά, δύναται να υπερβεί σε βαθμό το 4.