Μάθημα

Αλγοριθμική Σκέψη

Εργασία:

Tanks: Ένα Python Multiplayer Game,

με χρήση της Server – Client Αρχιτεκτονικής

Υπεύθυνοι Καθηγητές:

Μηνάς Δασυγένης

Πλόσκας Νικόλαος

Φοιτήτρια: Κωνσταντινίδου Αθηνά

Συνεργάτης: *Κοσμάς Σουρλής*

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΝΟΤΗΤΑ 1η

ΕΝΟΤΗΤΑ 2η

ΕΝΟΤΗΤΑ 3η

.

ΕΝΟΤΗΤΑ 1η

1 ΕΙΣΑΓΩΓΗ

Με αφορμή, την εκπόνηση της εργασίας στο μάθημα Αλγοριθμική Σκέψη του ΠΜΣ Ανάπτυξη Ψηφιακών Παιχνιδιών και Πολυμεσικών Εφαρμογών του Πανεπιστημίου Δυτικής Μακεδονίας, υλοποιήθηκε ένα multiplayer network παιχνίδι, στη γλώσσα Python. Το παιχνίδι ονομάζεται ''Tanks!''. Επιλέχθηκε το multiplayer χαρακτηριστικό, καθώς είναι πιο δημοφιλές και στα σύγχρονα παιχνίδια. Επιπλέον, δόθηκε ιδιαίτερη έμφαση στην κατασκευή βάσεων δεδομένων και ανταλλαγή (σε πραγματικό χρόνο) δεδομένων μεταξύ ενός Server και ενός Client, καθώς αποτελεί επίσης πολύ δημοφιλές χαρακτηριστικό στα σημερινά παιχνίδια. Σε αυτήν εδώ την αναφορά, εμβαθύνουμε περισσότερο στα χαρακτηριστικά του gameplay του παιχνιδιού, στις τεχνικές πληροφορίες, στη διαδικασία υλοποίησης του και γενικά σε πληροφορίες σχετικά με την ανάπτυξη του.

2 ΓΕΝΙΚΗ ΑΝΑΣΚΟΠΗΣΗ

Βρισκόμαστε στη μέση ενός νησιού (όπως διακρίνεται και από τη οθόνη έναρξης), προστατεύοντας ένα κάστρο, το οποίο είναι το τελευταίο καταφύγιο της χώρας. Όλα τα υπόλοιπα καταφύγια έχουν πέσει στα χέρια των εισβολέων. Προσπαθούμε να πετύχουμε τα εχθρικά τανκς ενώ κινούμαστε γύρω από το κάστρο με το δικό μας τανκ. Σκοτώνοντας τους εχθρούς, προστατεύουμε το κάστρο και αυξάνουμε το σκορ μας.

3 ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΠΑΙΧΝΙΔΙΟΥ

Πρόκειται για ένα Multiplayer παιχνίδι, όπου δύο διαφορετικοί παίκτες εφόσον βρίσκονται στο ίδιο δίκτυο Wifi, συνδέονται από δύο διαφορετικούς υπολογιστές. Στο παράθυρο του lobby, επιλέγει ο καθένας ξεχωριστά το βαθμό δυσκολίας (Easy, Medium, Hard), το tank που θα έχει και συνδέεται στο παιχνίδι.

Ένα ενδιαφέρον χαρακτηριστικό που υλοποιείται στην εφαρμογή, είναι η αυτόματη επιλογή των επιπέδων και του εναπομείναντος tank, για τον παίκτη που θα επιλέξει δεύτερος, καθώς ο παίκτης που επέλεξε πρώτος χρονικά έχει ήδη αποκλείσει τον δεύτερο από κάποιες επιλογές. Για παράδειγμα, αν ο ένας επιλέξει το μπλε τανκ, τότε αναγκαστικά, η μόνη επιλογή που δίνεται στον άλλο παίκτη, είναι το κίτρινο.

Η αρχική ιδέα του παιχνιδιού ήταν η υποστήριξη πολλών modes. Θα υπήρχε, δηλαδή η δυνατότητα τόσο τα tanks να πολεμούν μεταξύ τους, όσο και ύπαρξη ενός δεύτερου mode. Εκεί, τα δύο tanks συνεργάζονται στην ίδια ομάδα και αντιμετωπίζουν εχθρούς που κάνουν spawn, με σκοπό να επιτεθούν στο κάστρο. Η υλοποίηση του δεύτερου mode με Client-Server αρχιτεκτονική, ήταν αρκετά πολύπλοκη, οπότε η ιδέα αυτή αφέθηκε. Παρόλα αυτά, οι κώδικες για το spawn τέτοιων sprites υφίστανται στον κώδικα, χωρίς όμως να δημιουργούν κάτι.

4 ΧΡΗΣΗ & MECHANICS ΤΟΥ ΠΑΙΧΝΙΔΙΟΥ

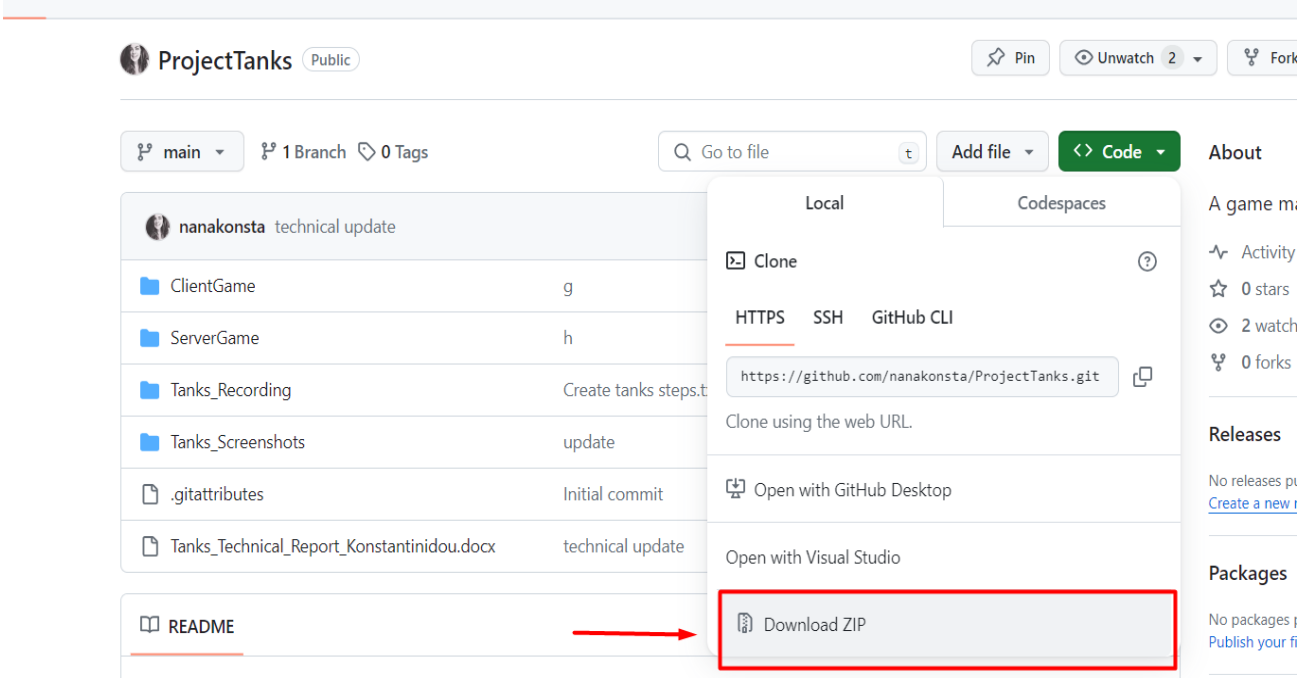
Το πρώτο που αντικρύζει ο παίκτης με το που ανοίγει το παράθυρο του game set είναι ένα κάστρο στη μέση του ταμπλό και το αντίπαλο τανκ.

Παίκτης: είναι ένα τανκ που αποτελείται από δύο μέρη. Το κάτω μέρος (base) κινείται μπρος πίσω με τα W και S αντίστοιχα. Ενώ το πάνω μέρος (turret), μπορεί να περιστρέφεται προς τα αριστερά με το A και προς τα δεξιά με το D. Με το δεξί κλικ του ποντικιού ο παίκτης βαράει (στην ουσία το turret) με σκοπό να σκοτώσει τον αντίπαλο. Το αρχικό πλάνο ήταν ένα mode, όπου τα δύο τανκς του ταμπλό είναι συμπαίκτες και προστατεύουν το κάστρο από αντιπάλους που κάνουν spawn.

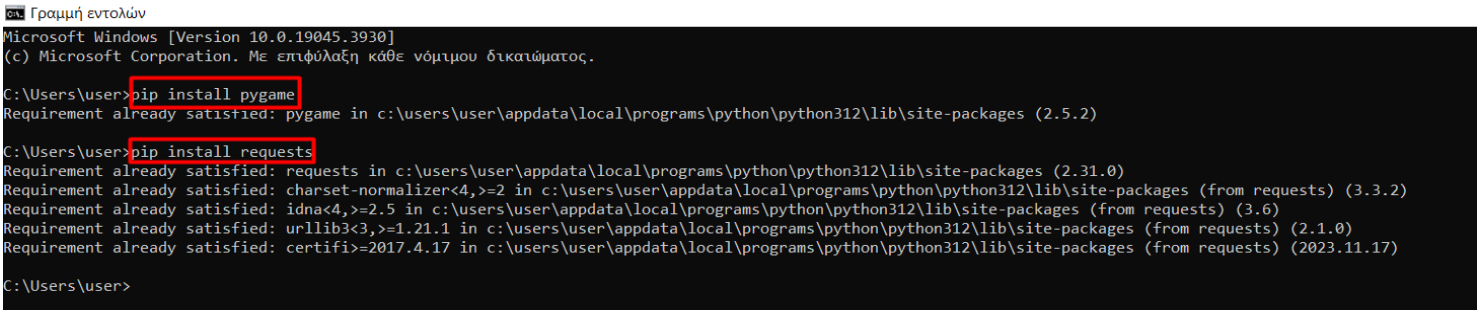
5 ΟΔΗΓΟΣ

Απαραίτητες προϋποθέσεις για την εγκατάσταση και την εύρυθμη λειτουργία της εφαρμογής:

1) Αρχικά, γίνεται η λήψη και κατόπιν αποσυμπίεση του συμπιεσμένου αρχείου zip από το ProjectTanks στο github, όπως στην εικόνα αλλά και του video capture από το One Drive (λόγω περιορισμού στο ανέβασμα στα 25 MB στο github)



2) Έπειτα, αφού εγκατασταθούν όλα τα αναγκαία προγράμματα και επεκτάσεις (Microsoft Visual Studio Code, Python), μέσω του command window εκτελούμε τις παρακάτω εντολές, όπως φαίνεται στην εικόνα. Είναι απαραίτητες για την εύρυθμη λειτουργία του κώδικα. Επίσης, αν χρειαστεί η ipv4 του localhost, μπορούμε να τη βρούμε από το command window, με αντίστοιχη εντολή.



3) Εφόσον έχουν εγκατασταθεί όλα τα αναγκαία softwares, είμαστε έτοιμοι να ανοίξουμε κανονικά το Visual Studio Code, ανοίγοντας New Window και επιλέγοντας Open Folder. Πρώτα ανοίγουμε το

φάκελο του Server και μετά του Client. Περισσότερες λεπτομέρειες αναφέρονται και στο video capture του παιχνιδιού

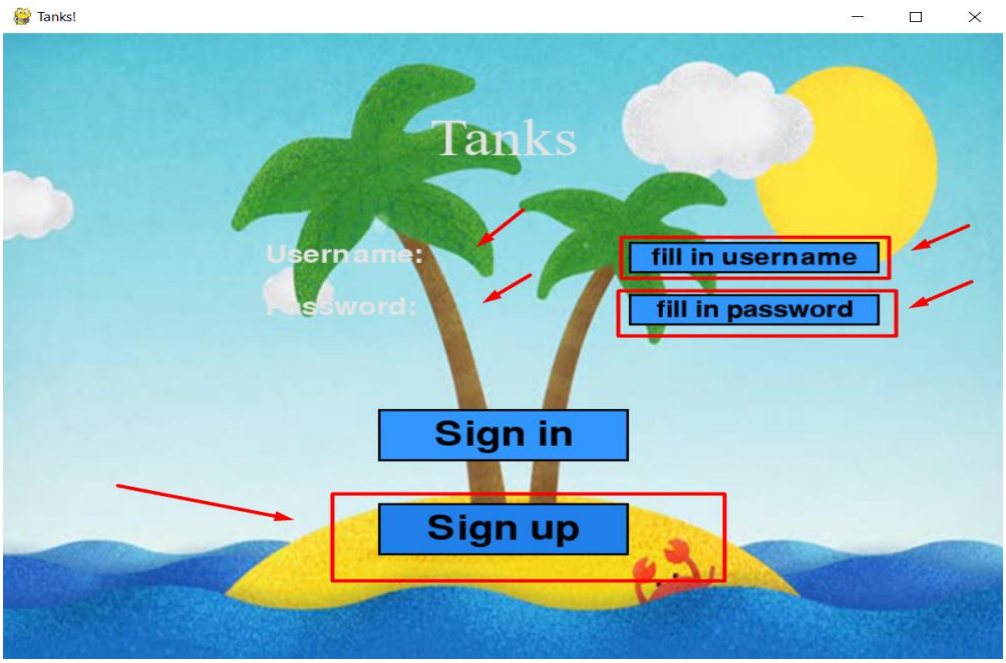
6 ΣΚΟΡ, ΝΙΚΗ & ΗΤΤΑ

Στην αρχή, οι παίκτες ξεκινάνε με ένα συγκεκριμένο νούμερο στο health bar τους, το οποίο μπορεί να κατέβει αν τους πετύχει ο αντίπαλος με κάποια σφαίρα. Όταν το health μηδενίσει, οι παίκτες χάνουν. Μάλιστα υπάρχει διαθέσιμο half implemented feature, ώστε να «πεταχτούν» αυτόματα από το παιχνίδι στο ενδεχόμενο ήττας. Ενώ αν μηδενίσει το health του αντιπάλου (κάτι που επιτυγχάνεται πυροβολώντας τον), τότε νικάνε. Παράλληλα ενσωματώσαμε και καταμέτρηση του σκορ και των πόντων που πετυχαίνουν, παρότι δεν έχει ρυθμιστεί το να εκφράζεται ακόμη. Υπάρχει δυνατότητα επέκτασης του κώδικα με το health του κάστρου, για μετέπειτα εξέλιξη του παιχνιδιού στο mode, όπου τα δύο τανκς είναι συμπαίκτες και προστατεύουν το κάστρο από npc αντιπάλους (sprites) που κάνουν spawn σε διάφορα σημεία του ταμπλό. Σε κάθε περίπτωση υπάρχουν σαφείς συνθήκες νίκης και ήττας και καταμέτρηση health και score.

7 ΙΔΙΑΙΤΕΡΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

Κάτι πολύ ιδιαίτερο που αναπτύχθηκε σε αυτό το παιχνίδι είναι η οργανωμένη αρχιτεκτονική του. Πρώτον, όσον αφορά το διαχωρισμό και την κατηγοριοποίηση των κλάσεων που περιγράφουν συγκεκριμένες λειτουργίες ή αντικείμενα (για παράδειγμα η κλάση tank). Δεύτερον, όσον αφορά το network connection, που αποτέλεσε και βασικό στόχο αυτού εδώ του project. Το μόνο που χρειάζεται είναι δύο διαφορετικές συσκευές υπολογιστών, συνδεδεμένες στο ίδιο δίκτυο Wifi. Τρίτον, όσον αφορά τη δυνατότητα περαιτέρω ανάπτυξης του κώδικα και επέκτασης των χαρακτηριστικών του παιχνιδιού.

Επιπρόσθετα, όπως απεικονίζεται και παρακάτω, αξιοσημείωτο feature αποτελεί και το γεγονός ότι οι παίκτες μπορούν να κάνουν τόσο register, όσο και sign in, κάτι που υλοποιείται με τις κλάσεις που περιέχονται στο signInLevelSignUp.py του Client.



ΕΝΟΤΗΤΑ 2η

8 ΤΕΧΝΙΚΕΣ ΛΕΠΤΟΜΕΡΕΙΕΣ

Σε πρώτο στάδιο, υλοποιήθηκε η κατασκευή απλών γραφικών και χρωμάτων, βασικών για τη λειτουργία της εφαρμογής. Αυτά τα γραφικά ήταν: το φόντο, τα μέρη των τανκς, το background image για το αρχικό παράθυρο, το κάστρο και οι σφαίρες. Η επεξεργασία των εικόνων έγινε μέσω του λογισμικού Adobe Photoshop.

Για την υλοποίηση του κώδικα χρησιμοποιήθηκαν εξ ολοκλήρου: η Pygame και το πρόγραμμα Microsoft Visual Studio Code, όπου εγκαταστάθηκαν οι αντίστοιχες επεκτάσεις, όπως το Python Extension.

Για την καταγραφή των screenshots χρησιμοποιήθηκε το lightshot, ενώ για το video capture χρησιμοποιήθηκε το software free cam.

Οι ήχοι που χρησιμοποιήθηκαν, ήταν διαθέσιμοι ως free audio effects στο internet, ωστόσο τους έχουμε κάνει μία επεξεργασία μέσω Audacity (pitch, αυξομείωση db, cut κλπ).

9 ΕΠΕΞΗΓΗΣΗ ΤΟΥ ΚΩΔΙΚΑ & ΛΕΙΤΟΥΡΓΙΚΟΤΗΤΑ

Όπως φαίνεται και από το διαχωρισμό των αρχείων .py σε δύο φακέλους (Client Game και Server Game), στην ουσία η εφαρμογή αποτελείται από δύο projects, τον client και τον server. Αν και αρκετές κλάσεις είναι κοινές και στα δύο, υπάρχουν κάποιες σημαντικές διαφορές που καλό είναι να τονιστούν.

- Η υλοποίηση βασικών αντικειμένων που βλέπει ο παίκτης στον υπολογιστή του και η γενικότερη δημιουργία του παραθύρου, γίνεται από τον Client. Δηλαδή, τα αντικείμενα δημιουργούνται τοπικά.
  Φυσικά, έχουμε προβλέψει την αποστολή των δεδομένων από τον Client στον Server, αλλά εν γένει, τα στοιχεία δημιουργούνται στον Client και αποστέλλονται στον Server, με χρήση αρχείων json και βάσεων δεδομένων.
- Όλα τα δεδομένα, όπως όνομα username, password, ταυτόχρονη αντίληψη και εμφάνιση του αντιπάλου στέλνονται και αποθηκεύονται στον server.
- Για τη λειτουργία του παιχνιδιού, πρώτα ξεκινά να ''τρέχει'' ο server και μετά ο client.
- Αν ο κάθε παίκτης συνδεθεί από διαφορετικό υπολογιστή, απαιτείται μόνο ένα run του project client game και εκχώρηση της IPv4 του localhost (δηλαδή του δικτύου WIfi). Για να εμφανιστούν και τα δύο παράθυρα παικτών στον ίδιο υπολογιστή, όμως, απαιτείται ένα run και μετά ένα debug python file.
- Πολλά στοιχεία είναι κοινά και στον server και στον client με διαφορετική όμως ενσωμάτωση.

Ακολουθεί συνοπτική περιγραφή της λειτουργικότητας των διαφορετικών αρχείων, κλάσεων, μεθόδων και συναρτήσεων που περιέχονται στο server και στο client project.

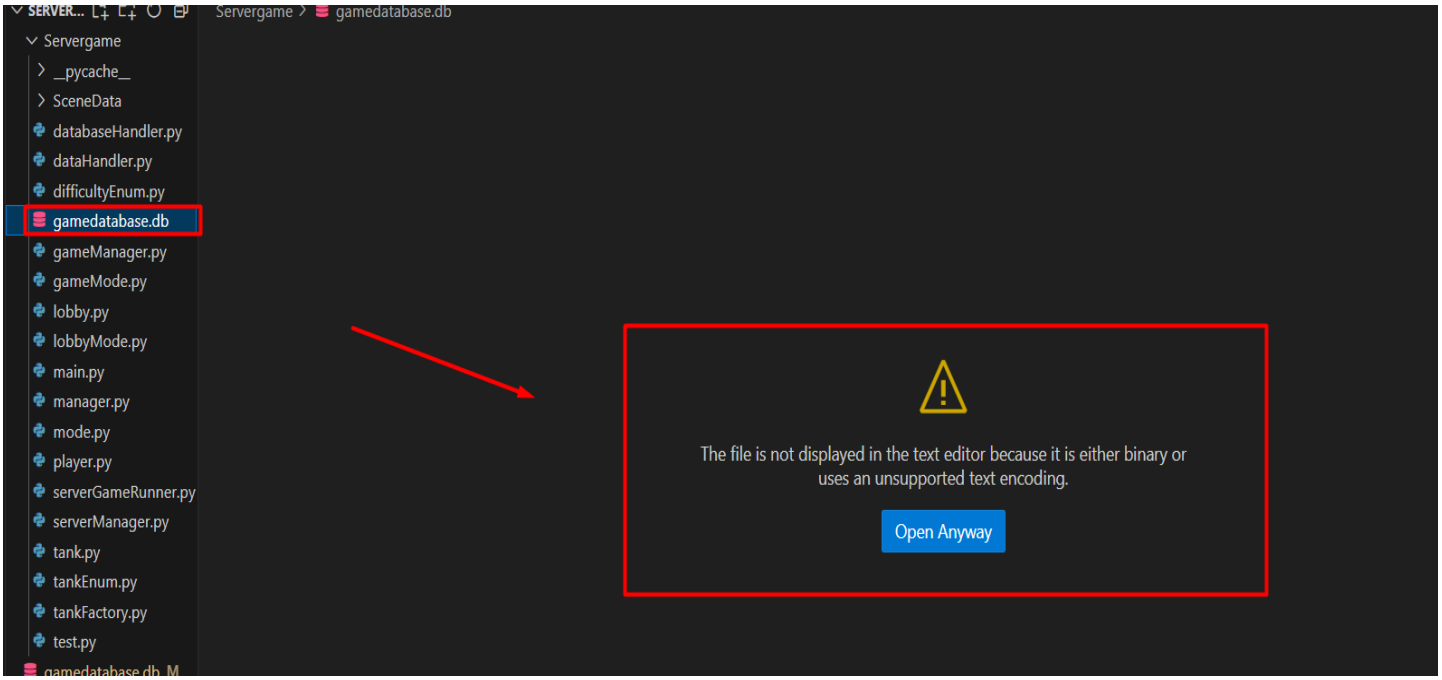
A) SERVER

Η γενική λειτουργία του server είναι να λαμβάνει δεδομένα από τους χρήστες και να τα στέλνει πίσω. Τέτοια δεδομένα μπορεί να είναι, ποιο χρώμα tank διάλεξε ο παίκτης, που βρίσκεται ο αντίπαλος, η κατάσταση του health bar κλπ. Ο τρόπος με τον οποίο διατηρεί τα δεδομένα είναι με αρχεία json, φυλάσσοντας τα σε μία database.

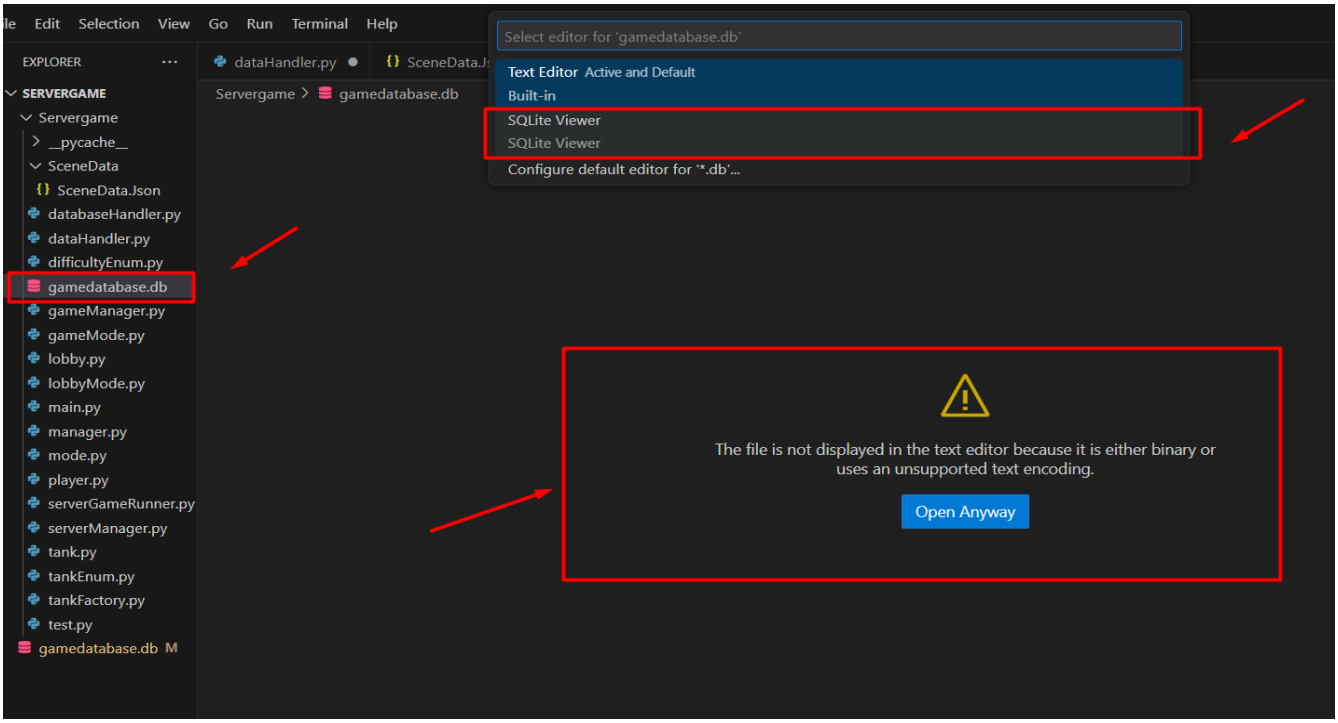
Ας εμβαθύνουμε στη λειτουργικότητα όλων των αρχείων (κυρίως κλάσεων) της μορφής .py που βρίσκονται στον Server.

Στην ουσία, όπως περιγράφεται και στο συνοδευτικό βίντεο, πρώτα ξεκινάει ο server και ανοίγει ένα παράθυρο που τρέχει το κεντρικό νήμα (thread) και δύο νήματα, ένα για κάθε παίκτη (νήμα container επεξεργασίας στον υπολογιστή). Αυτά τα δύο νήματα αναλαμβάνουν τη λήψη και τη μετάδοση δεδομένων σε σχέση με τον παίκτη. Πιο αναλυτικά, ακολουθεί περιγραφή όλων των λειτουργικών του αρχείων (κυρίως κλάσεων).

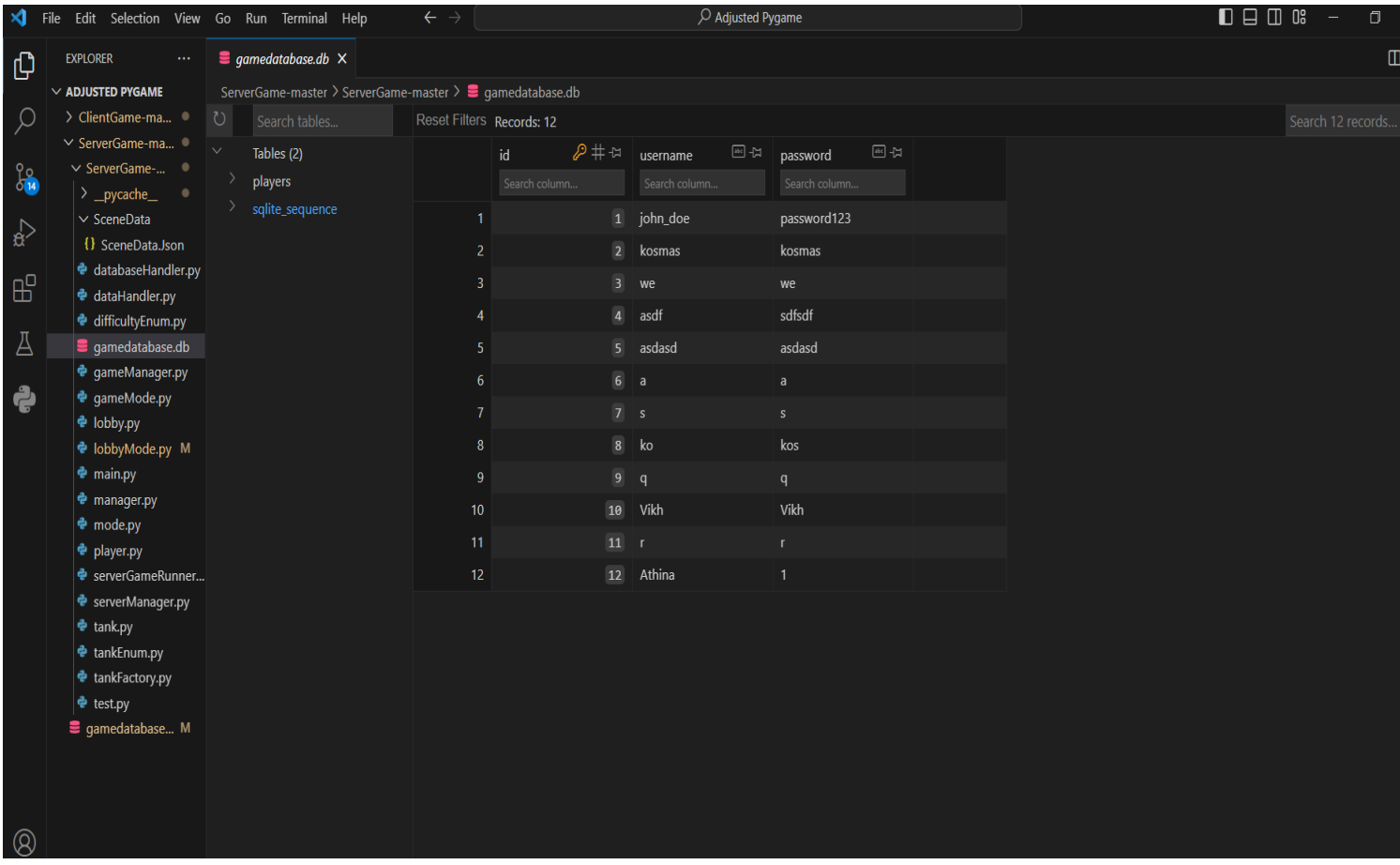
1. databaseHandler.py : εκτελεί δοσοληψίες ερωτημάτων (requests) στη βάση, στέλνοντας δεδομένα. Για παράδειγμα create, delet, κλπ
2. difficultyEnum.py : είναι στην ουσία μια κλάση αρίθμησης των επιπέδων του παιχνιδιού. Δηλαδή τα επίπεδα Easy, Medium και Hard, αντιστοιχίζονται στα value 1, 2, 3, αντίστοιχα. Ανάλογα με τη δυσκολία του παιχνιδιού, ο παίκτης έχει μικρότερο ή μεγαλύτερο fire rate και ζωή.
3. gamedatabase.db: είναι η βάση όπου φυλάσσονται όλα τα δεδομένα των παικτών. Διατηρεί όλα τα στοιχεία των παικτών, τα οποία αποστέλλονται (παράδειγμα για επαλήθευση) μέσω του databaseHandler. Αρχικά εμφανίζεται κρυμμένη αλλά μπορεί να φανερωθεί, αν ακολουθήσουμε τα παρακάτω βήματα, όπως περιγράφεται και στις εικόνες που ακολουθούν.



Αρχική εμφάνιση του database



Άνοιγμα του database με SQLite Viewer



Εμφάνιση στοιχείων των παικτών

4. gameManager.py: αποτελεί μία από τις πιο σημαντικές κλάσεις του παιχνιδιού, καθώς κάνει διαχείριση (management) διάφορων aspects του παιχνιδιού. Πρόκειται για μία βοηθητική κλάση η οποία, κατά τη διάρκεια του παιχνιδιού, συγκρατεί δεδομένα, τα οποία χρησιμοποιούνται σε διάφορες άλλες κλάσεις. Παραδείγματος χάριν, ποιος παίκτης αντιστοιχεί σε ποιο τανκ και ποιο είναι το username του παίκτη, πόσο fire rate και πόσο health έχει ο κάθε παίκτης. Επίσης, περιέχει κάποια instances της κλάσης player (player1, player2).

   Εδώ επίσης, αναφέρονται κάποια attributes , των οποίων ο τίτλος είναι αυτοπεριγραφικός, όπως: difficulty (επίπεδο δυσκολίες), castleHealth, enemiesHealth, enemiesFireRate, players και enemies (λίστες όπου αποθηκεύονται αντίστοιχα instances) και castle_rect (ένα rectangle που αντιπροσωπεύει το κάστρο). Αξίζει να σημειωθεί ότι το property difficulty λαμβάνει το επίπεδο δυσκολίας σε μορφή string, ενώ η μέθοδος difficulty setter, ορίζει το επίπεδο δυσκολίας και κάνει update διάφορες παραμέτρους, ανάλογα με το επίπεδο δυσκολίας (player 1 & 2 health, enemies health, player 1 & 2 fire rate, enemies fire rate). Για παράδειγμα, όπως φαίνεται και στη διπλανή εικόνα, το castle health μικραίνει όσο ανεβαίνουμε επίπεδο δυσκολίας.

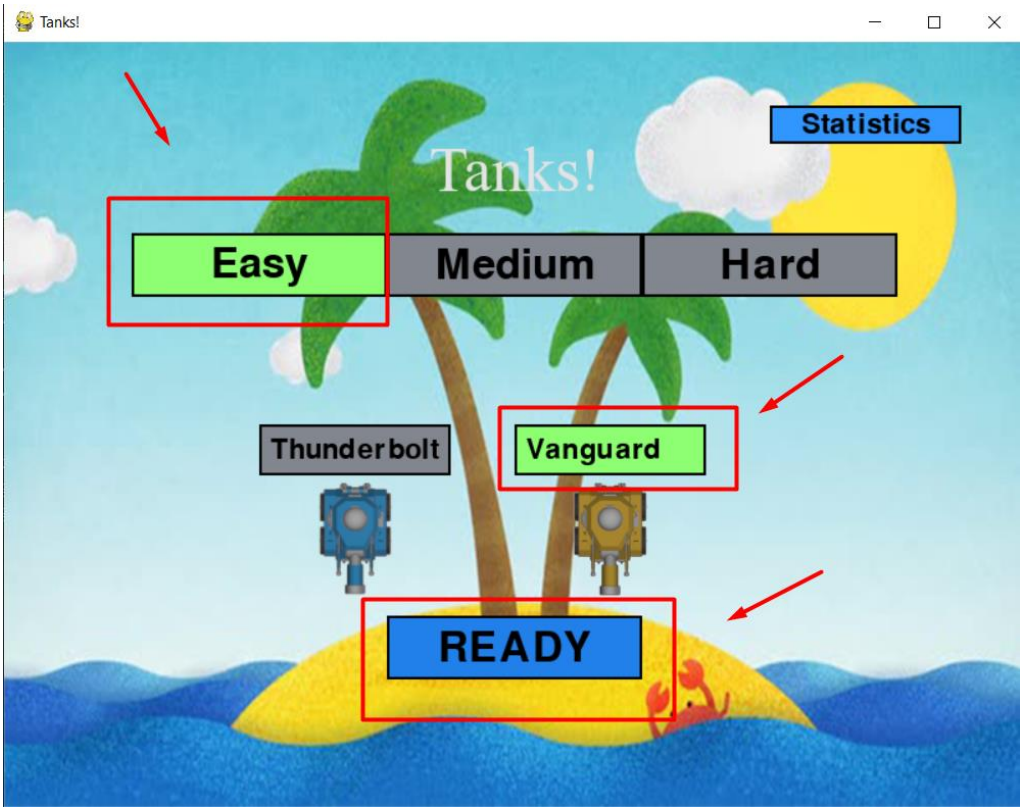
```python
@difficulty.setter
def difficulty(self, value):
    self._difficulty = value
    match self._difficulty:
        case Difficulty.EASY:
            self.castleHealth = 1500
            self.player1.health = 300
            self.player2.health = 300
            self.enemiesHealth = 100
            self.player1.fireRate = 200
            self.player2.fireRate = 200
            self.enemiesFireRate = 1500
        case Difficulty.MEDIUM:
            self.castleHealth = 1000
            self.player1.health = 200
            self.player2.health = 200
            self.enemiesHealth = 150
            self.player1.fireRate = 300
            self.player2.fireRate = 300
            self.enemiesFireRate = 1000
        case Difficulty.HARD:
            self.castleHealth = 500
            self.player1.health = 100
            self.player2.health = 100
            self.enemiesHealth = 200
            self.player1.fireRate = 400
            self.player2.fireRate = 400
            self.enemiesFireRate = 1000
```

5. gameMode.py : μία επίσης πολύ σημαντική κλάση, καθώς χάρη σε αυτή ''παίζει'' το παιχνίδι. Θα ήταν ορθότερο αν τη χαρακτηρίζαμε «υποκλάση» της κλάσης mode, από την οποία κληρονομεί. Σκοπός της gameMode είναι να κάνει διαχείριση του game state (θέσεις παικτών ή εχθρών, health, αν κάποιος έχει βαρέσει με το κανόνι, αν έχει χαθεί ζωή κλπ). Διατηρεί πληροφορίες για τους παίκτες και τα δεδομένα τους και διευκολύνει updates και drawings στο game set. Περιλαμβάνει: τη μέθοδο init (αρχικοποιεί το game mode instance, οργανώνοντας κάποια χαρακτηριστικά σε σχέση με τα player data, game manage και game runner), τη μέθοδο update (κάνει ανανέωση του game state, με βάση δεδομένα που λαμβάνει για τους παίκτες, εξάγει πληροφορίες όπως η κατάσταση spawning παίκτη, θέσεις turret (από εκεί που βαράει) και base, γωνίες, health, fire rates), τη μέθοδο draw (καλεί τη μέθοδο draw της κλάσης γονέα Mode και απεικονίζει το current game state στην οθόνη) και τη μέθοδο getJSONgameModeData (μετατρέπει τα ενημερωμένα game data σε string τύπου json).

6. lobby.py: λειτουργεί ως βοηθητική κλάση στο lobbyMode, η οποία διατηρεί τα δεδομένα αρχείων json, ώστε το lobbyMode να μπορεί να τα αποστείλει στους clients. Στην ουσία διαχειρίζεται το συγχρονισμό των επιλογών που γίνονται στο παράθυρο του lobby, μεταξύ server και client. Η κλάση lobby είναι αναγκαία για το συντονισμό επιλογών επιπέδου, vehicle και readiness των παικτών.

   Αξίζει να σημειωθούν οι μέθοδοι:

   - idDifficultyLocked: κάνει έλεγχο αν έχει επιλεγεί το difficulty level στο lobby, επιστρέφοντας True αν έχει επιλεγεί τουλάχιστον ένα
   - lockVehicleA και lockVehicleB: κλειδώνουν την επιλογή οχήματος (Thunderbolt ή Vanguard), επιστρέφοντας False αν έχουν ήδη επιλεγεί τα οχήματα.
   - lockEasy, lockMedium, lockHard: κλειδώνουν το αντίστοιχο level που έχει επιλεγεί.
   - isVehicleLocked: ελέγχει αν έχει επιλεγεί ένα συγκεκριμένο vehicle, επιστρέφοντας true, αλλιώς False.
   - lockPlayer: κλειδώνει την κατάσταση ready του παίκτη. Για παράδειγμα, αν ο παίκτης p2 πατήσει ready, κλειδώνει ready_Player2Locked.
   - isPlayerReadyLocked: ελέγχει αν το readiness ενός συγκεκριμένου παίκτη είναι κλειδωμένο, επιστρέφοντας True, αν πάτησε Ready.
   - bothPlayersLocked: ελέγχει αν και οι δύο παίκτες έχουν πατήσει το Ready, επιστρέφοντας True σε αυτή την περίπτωση.
   - getJSONLobbyData: μετατρέπει τα δεδομένα του lobby σε string τύπου JSON.



Level, Vehicle, Readiness locked

7. lobbyMode.py: συγχρονίζει τα δεδομένα στο lobby των παικτών. Δηλαδή, την επιλεγμένη δυσκολία, το επιλεγμένο τανκ και το ready. Πιο αναλυτικά, διαχειρίζεται τα δεδομένα του lobby, επεκτείνει την κλάση Mode και περιλαμβάνει μεθόδους για update και drawing του lobby. Αλληλοεπιδρά με την κλάση lobby με σκοπό τη διαχείριση των επιλογών των παικτών, των vehicles και των levels. Ιδιαίτερα, αναφορικά με τη μέθοδο lockVehicle, αυτή ορίζει το επιλεγμένο όχημα για ένα συγκεκριμένο παίκτη στο game manager. Τονίζουμε ότι εφαρμόζεται η match statement για τη διαχείριση διαφορετικών οδηγιών από τους παίκτες. Γενικά, η LobbyMode παίζει κρίσιμο ρόλο στο συγχρονισμό και τη διαχείριση δεδομένων των επιλογών των παικτών και δεδομένων του lobby, προετοιμάζοντας το έδαφος για το gameplay.

8. Main.py : η κύρια κλάση του κώδικα, μέσω της οποίας τρέχει ο server. Στην ουσία από εδώ ξεκινά να τρέχει το πρόγραμμα του server. Αφού γίνονται τα απαραίτητα imports (pygame, socket, threading…), υλοποιούνται τα παρακάτω:

    - δημιουργεί instances για το ServerGameRunner και το ServerManager και εκτυπώνει τη δημόσια IP του server
    - ορίζει ένα exit_event για να σηματοδοτήσει τη λήξει του νήματος του server.
    - Ορίζει τη συνάρτηση handle_connection, η οποία αναλαμβάνει τη διαχείριση μιας σύνδεσης client, χρησιμοποιώντας τη μέθοδο threaded_client του ServerManager
    - Ορίζει τη συνάρτηση start_server, που τρέχει σε loop, συνεχώς, δεχόμενη νέους clients.
    - Δημιουργεί ένα νέο thread (server_thread_obj, που τρέχει τη συνάρτηση start_server, περνώντας το ServerManager instance και το exit event.
    - Αρχικοποιεί την pygame και έπειτα καλεί τη μέθοδο preRunServerWindow από το ServerGameRunner, ώστε να σεταριστεί το server window.

9. Manager.py: η κλάση αυτή είναι half implemented feature. Ο αρχικός σκοπός της ήταν να αφαιρεί και να προσθέτει στο παιχνίδι παίκτες. Για παράδειγμα, αν κάποιος παίκτης πέθαινε, θα έβγαινε εντελώς εκτός παιχνιδιού.
10. Mode.py: περιλαμβάνει μεθόδους ώστε να συγκρατεί δεδομένα όπως: μέγεθος παραθύρου, χρώμα, δεδομένα παίκτη, διαχείριση Json δεδομένων για τον παίκτη. Μάλιστα το πρώτο update που γίνεται, γεμίζει την οθόνη με κόκκινο (είναι το παράθυρο που εμφανίζεται μετά το run python file της main στον server) και η μέθοδος draw γεμίζει το παράθυρο. Η ανάθεση τιμής γίνεται στο runtime (δηλαδή όταν ξεκινήσουμε να παίζουμε το παιχνίδι.

11. Player.py: περιέχει τη username και τη vehicle. Εδώ ελέγχεται αν είναι ο πρώτος ή ο δεύτερος παίκτης που συνδέεται και αρχικοποιείται πόση ζωή έχει και το fire rate.

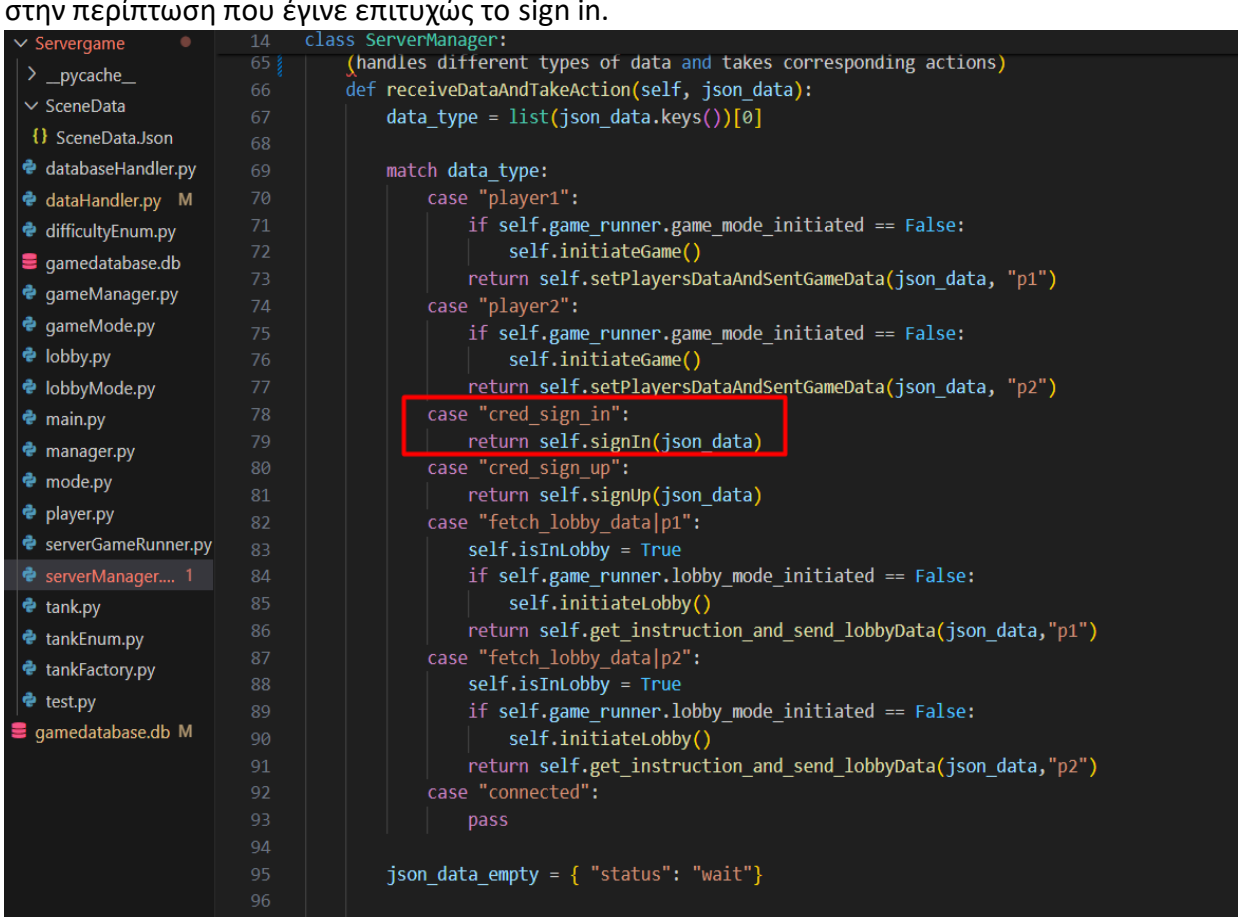
12. serverGameRunner.py: τρέχει το βασικό παράθυρο του παιχνιδιού και είναι υπεύθυνη για πάρε δώσε δεδομένων, όπως: τα events (στην περίπτωση του server δεν χρησιμοποιούνται κάπου διότι δεν υπάρχει input από τον παίκτη.

13. serverManager: αποτελεί ιδιαίτερα σημαντική μέθοδο για το παιχνίδι. Είναι επί της ουσίας, η κλάση που αρχικά παραλαμβάνει τα δεδομένα των παικτών από τους εκάστοτε clients. Τα requests (πχ sign in, username, password) που γίνονται από τους Clients, στέλνονται εδώ, όπου γίνεται ένας έλεγχος επαλήθευσης του παίκτη και των δεδομένων του. Επίσης από εδώ εγκρίνεται η είσοδος του.

    Αξιοσημείωτες αναφορές εντός του serverManager:

    - Η init: παίρνει και αποθηκεύει την IP του Server, αρχικοποιεί τα flags των παικτών ('self.p1_in_game' και 'self.p2_in_game') και αρχικοποιεί ένα socket για server communication
    - Η get_public_ip: χρησιμοποιεί ένα API (application programming interface) για να πάρει την public διεύθυνση IPv4.
    - Η μέθοδος receiveDataAndTakeAction: αναζητά δεδομένα του τύπου player1, cred_sign_in, cred_sign_up, fetch_lobby_datalp1, connected, διχειρίζεται διαφορετικού τύπου data που στέλνονται από τους clients και, ανάλογα, εκτελεί τις αντίστοιχες εντολές.

    Στην ουσία, παίρνει το πρώτο node του json αρχείου και ανάλογα με την αντίστοιχη οδηγία, εκτελεί ένα action. Για παράδειγμα, ένα action pc sign in παίρνει το username και το password, δημιουργεί ένα query-request στην databaseHandler και επιστρέφει σε μορφή json, στην περίπτωση που έγινε επιτυχώς το sign in.



14. Tank.py: εδώ καθορίζεται η κλάση tank. Χαρακτηριστικά που αντιπροσωπεύονται εδώ είναι: η αρχική θέση και διαστάσεις του tank, το draw του τανκ, διαχωρισμός των ιδιοτήτων του πάνω μέρους (με την TankTurret) του τανκ από το κάτω (με την TankBase), update του τανκ, ορισμός της κλάσης ''παιδί'' bullet και της fireCannon(υπεύθυνη για τη βολή σφαιρών) και collision detection. Επίσης από τις παραπάνω κλάσεις, αυτές που δημιουργούν αντικείμενα, περιέχουν και δυνατότητα συγχρονισμού με το δίκτυο, δηλαδή κατασκευάζεται αντικείμενο που στέλνεται στο δίκτυο.

15. tankEnum.py : αποτελεί enumeration , το οποίο χρησιμοποιεί την κλάση Enum από το module enum της Python. Γενικά τα enumerations χρησιμεύουν στη δημιουργία named constant τιμών. Στη συγκεκριμένη κλάση, τα P1, P2, E1, E2 είναι τα 4 μέλη enumerators, ενώ οι τιμές 1, 2, 3,4 είναι οι αντίστοιχες τιμές που ανατέθηκαν σε αυτά (P1 = player1 = 1 , P2 = player1 = 2, E1 = enemy1 =3, E2 = enemy2 = 4). Στην ουσία εδώ γίνεται μια αντιπροσώπευση των τανκς, ώστε να έχουν νόημα σαν οντότητες εντός του κώδικα.

16. tankFactory.py: αποτελεί μία από τις πιο σημαντικές κλάσεις, η οποία περιέχει επίσης δύο κλάσεις, την init και την constructTank. Η κλάση tankFactory, ενσωματώνοντας το Factory Pattern, είναι υπεύθυνη για την κατασκευή αντικειμένων τύπου τανκς. Η χρήση της init εδώ είναι αυτή του constructor, που κάνει αρχικοποίηση των image paths, ώστε να δημιουργηθούν τα τανκς. Αξίζει να αναφερθεί ότι τα τανκς αποτελούνται από δύο μέρη, τη βάση, η οποία μετακινείται μπρος πίσω και το πάνω μέρος, το οποίο περιστρέφεται ώστε να βαράει προς κάθε κατεύθυνση.

17. test.py: η κλάση αυτή ορίζει τη συνάρτηση recieveData, η οποία δέχεται ως παράμετρο μια δομή δεδομένων τύπου json_data. Η συνάρτηση εξάγει τον τύπο δεδομένων από τα keys του dictionary στο json_data Έπειτα, χρησιμοποιεί μια δήλωση match για να καθορίσει τον τύπο δεδομένων και να εκτελέσει συγκεκριμένες λειτουργίες, βασισμένες σε αυτόν τον τύπο. Ειδικότερα, η εντολή data_type = list(json_data.keys ())[0] , είναι αυτή που εξάγει το πρώτο key από το dictionary, θεωρώντας ότι αντιπροσωπεύει τον τύπο δεδομένων, ενώ με την παρακάτω δομή, ελέγχεται αν ο τύπος δεδομένων είναι cred_sign_in, cred_sign_up ή game_data και ανάλογα εκτυπώνει αντίστοιχο μήνυμα ή όχι.

```
match data_type:
    case "cred_sign_in":
        print(1)
    case "cred_sign_up":
        print(2)
    case "game_data":
        pass
```

Τέλος με την εντολή receiveData(data_to_server), καλείται η συνάρτηση, δημιουργώντας μια προσομοίωση sign-in request και αντίστοιχα τυπώνοντας 1 για το cred_sign_in.

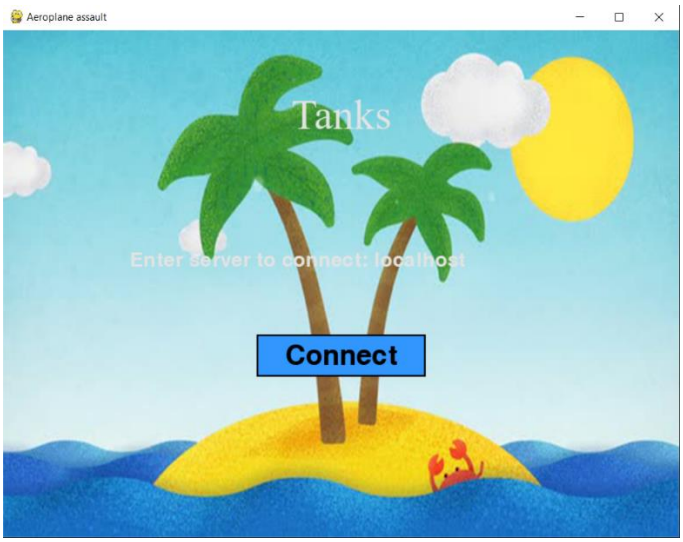
Β) CLIENT

Η γενική λειτουργία του client είναι να δημιουργεί δεδομένα, όπως είναι ο σχηματισμός του παραθύρου., η ανανέωση των frames κλπ. Επίσης τέτοια δεδομένα μπορεί να είναι ποιο χρώμα tank διάλεξε ο παίκτης, που βρίσκεται ο αντίπαλος, η κατάσταση του health bar κλπ. Ας εμβαθύνουμε στη λειτουργικότητα όλων των αρχείων της μορφής .py που βρίσκονται στον Client.

- Αν ο κάθε παίκτης συνδεθεί από διαφορετικό υπολογιστή, απαιτείται μόνο ένα run του project client game. Για να εμφανιστούν και τα δύο παράθυρα παικτών στον ίδιο υπολογιστή, όμως, απαιτείται ένα run και μετά ένα debug python file.
- Τα δεδομένα τυπικά καταχωρούνται μόνο στο Server, παρόλα αυτά κάποια προσωρινά δεδομένα που αφορούν την οπτική υλοποίηση, ενδέχεται να καταχωρούνται και εδώ.

Ακολουθεί περιγραφή όλων των λειτουργικών αρχείων (κυρίως κλάσεων) του client.

1. clientGameManager.py: είναι μία κλάση που διαχειρίζεται το game flow και την επικοινωνία με το δίκτυο. Με την έναρξη, δημιουργεί ένα ClientNetworkManager instance, οριοθετεί το τρέχων level και αρχικοποιεί έναν παίκτη.
   - Η μέθοδος switchLevel διευκολύνει τη μετάβαση σε ένα νέο level, κάνοντας update το τρέχων level και οργανώνοντας το client game manager.
   - Η μέθοδος runGame τρέχει το παιχνίδι σε loop, κάνει update και draw το τρέχων level, ενώ κάνει event handling.
2. clientNetworkManager.py: βασική κλάση, η οποία δίνει και παίρνει πληροφορίες του παίκτη και τα στέλνει στο δίκτυο. Στην ουσία αυτή η κλάση, κάνει εφικτή τη σύνδεση με το server. Ειδικότερα, αρχικοποιεί ένα socket και λεπτομέρειες σύνδεσης, συμπεριλαμβανομένων των host και port.
   - Με τη μέθοδο connect, εκτελείται απόπειρα σύνδεσης με τον server.
   - Οι μέθοδοι setHost και getHost, χρησιμεύουν στον καθορισμό και στην ανάκτηση της host address.
   - Η μέθοδος send, στέλνει δεδομένα στον server, μέσω JSON και η μέθοδος receive, ανακτά δεδομένα από τον server.
3. entryLevel.py: αντιπροσωπεύει το πρώτο παράθυρο που εμφανίζεται όπου ο παίκτης συνδέεται στο localhost και πατά connect.



4. gameLevel.py: εξίσου σημαντική κλάση, καθώς αντιπροσωπεύει την κεντρική πίστα του παιχνιδιού. Διαχειρίζεται το gameplay,εντός ενός level, περιλαμβάνει αρχικοποίηση των στοιχείων του παιχνιδιού, handling του player input, updating του game state, επικοινωνία με τον server μέσω JSON κλπ. Στην ουσία εδώ γίνεται draw του παραθύρου, των παικτών, του κάστρου, των σφαιρών και οτιδήποτε άλλο εμφανίζεται στο παράθυρο που παίζουμε. Με τα updates που εκτελούνται, επιτυγχάνεται event handling, ανανέωση του game state , πάρε δώσε δεδομένων με τον server (ανάλογα και με τη θέση του παίκτη) και δημιουργία των UI στοιχείων. Επίσης, εδώ περιέχονται μέθοδοι spawning παικτών και αντικειμένων. Συνδέεται και με την κλάση level.py που ακολουθεί παρακάτω.
5. level.py: λειτουργεί ως βάση για όλα τα levels στο παιχνίδι, παρέχοντας μία δομή για updating του game logic και draw των στοιχείων του επιπέδου στην οθόνη. Η μέθοδος setClientManager , καθορίζει στην ουσία, το game manager του client για ένα συγκεκριμένο level. Γενικά η κλάση αυτή λειτουργεί ως κάτι που κληρονομείται από συγκεκριμένες level κλάσεις, επιτρέποντας έτσι την προσαρμοστικότητα.
6. lobbyLevel.py: είναι μία κλάση που έχει άμεση σχέση με τη level.py. Η κλάση αυτή αντιπροσωπεύει το lobby, όπου εισέρχονται οι παίκτες για να επιλέξουν επίπεδα, vehicles κλπ. Αποτελεί ένα δομημένο περιβάλλον αναμονής των παικτών πριν την έναρξη του παιχνιδιού.
   Τα πιο σημαντικά στοιχεία σε αυτή την κλάση είναι:
   - τα UI στοιχεία, όπως τα κουμπιά, η ενσωμάτωση γραφικών (tank sprites, background image) και η εισαγωγή ηχητικών εφέ.
   - Το Input handling που εκτελείται καθώς και το player tracking. Δηλαδή οι διαχειρίσεις και έλεγχοι που γίνονται για το input επιλογών του παίκτη και η διατήρηση αυτών των επιλογών.

7. main.py: η κύρια κλάση που τρέχει όλο το client. Επίσης συνδέεται με το server. Αρχικά, η main διαβάζει τα δεδομένα από ένα JSON file (EntryScene.json), αρχικοποιεί την pygame, δημιουργεί αντικείμενα (screen: αντιπροσωπεύει το παράθυρο, currentLevel: instance του entryLevel.py που αντιπροσωπεύει την αρχική σκηνή). Επιπλέον, χρησιμοποιεί ένα instance του ClientGameManager και με παράμετρο το currentLevel, γίνεται διαχείριση πτυχών του παιχνιδιού όπως η αλληλοεπίδραση μεταξύ client και server. Τέλος, με τη μέθοδο runGame του ClientGameManager, ξεκινά το παιχνίδι να τρέχει σε loop, επιτρέποντας τη σύνδεση του παίκτη στον server. Η λειτουργία της γενικά παρομοιάζει αρκετά αυτή του server.

8. player.py: η κλάση του παίκτη, που περιλαμβάνει:
   - attributes όπως: username, vehicle, _is_P1, _is_P2 (Boolean flags που δείχνουν αν ο παίκτης είναι ο P1 ή ο P2),
   - properties και συναρτήσεις setters: is_P1, is_p2 (έτσι παίρνουμε και θέτουμε τα Player 1 και Player 2 flags), username, vehicle (properties για πρόσβαση και τροποποίηση στα username και vehicle attributes
   - τη μέθοδο getPlayerStr(): επιστρέφει ένα string (συμβολοσειρά) identifier (p1 ή p2), βασισμένη στο ποιος είναι ο παίκτης

9. signInLevelSignUp.py: ιδιαίτερα σημαντική κλάση, όπου ορίζονται δύο άλλες κλάσεις, η SignInLevel και η SignUpLevel, οι οποίες προέρχονται από την κλάση Level.

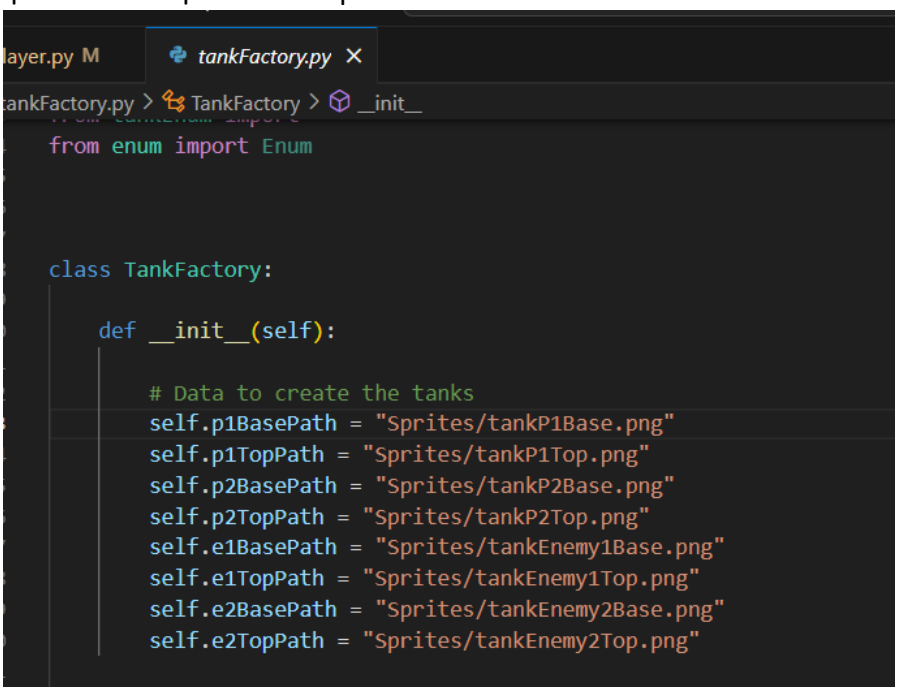
Η κλάση SignInLevel, αντιπροσωπεύει την εμφάνιση, δημιουργία και τα γενικότερα GUI στοιχεία της οθόνης όπου γίνεται sign in, όπως κουμπιά, γραμματοσειρά, ήχοι, πεδία εκχώρησης, ενώ η SignUpLevel τα αντίστοιχα όπου όμως γίνεται sign up. Κάτι ενδιαφέρον εδώ είναι ότι, πχ το πάτημα του κουμπιού sign in, εκκινεί ένα request στο server για επαλήθευση στοιχείων (line 96-122) και μόνο αν είναι πετυχημένο θα μεταφερθεί στο lobby. Αλλιώς εμφανίζεται το μήνυμα Authentication Failed. Η μέθοδος update διαχειρίζεται events που αφορούν τα user input, όπως πάτημα πλήκτρων, κλικ ποντικιού, ενώ η μέθοδος draw κάνει επίστρωση πάνω στη σκηνή.

10. tank.py: αρκετά παρόμοια λειτουργικότητα με την αντίστοιχη tank.py του server. Δημιουργεί ένα tank με base και turret, το οποίο μπορεί να μετακινείται, να περιστρέφεται και να πυροβολεί. Η ''συμπεριφορά'' του αλλάζει ανάλογα με το ποιος παίκτης το οδηγεί. Περιλαμβάνονται διάφορες κλάσεις: TankTurret, Bullet (χρησιμοποιείται για τοπικό gameplay, ενδέχεται να θέλει διόρθωση), NetworkBullet (πιο νοητή, αντιπροσωπευτική κλάση, χρησιμοποιείται για συγχρονισμό με το δίκτυο, ενδέχεται να θέλει διόρθωση), updateNetworkData (για το συγχρονισμό του τανκ).

11. tankEnum.py: όμοια δράση με την αντίστοιχη tankEnum.py του server.

12. tankFactory.py: δημιουργεί διαφορετικούς τύπους tanks. Πιο αναλυτικά:

- περιέχει attributes που αποθηκεύουν file paths για τα διάφορα μέρη του tank, όπως φαίνονται στην εικόνα παρακάτω



- Η μέθοδος constructTank, δημιουργεί tank instances, βασισμένη στο tankType και παραμέτρους όπως: tank_pos_x, tank_pos_y, distanceTostop, fireRate, health, gameManager κ.ά. Ανάλογα με το tankType που δίνεται, χρησιμοποιεί μια match statement για να επιλέξει το κατάλληλο σχηματισμό τανκ. Για κάθε ένα tankType, δημιουργεί ένα TankBase instance και ένα TankTurret instance (με παραμέτρους όπως position, image path, speed κλπ και στα δύο)

Επιπλεόν δημιουργεί ένα TankTurret instance, που συνδέεται με τη βάση. Το tank turret θεωρείται ''παιδί'' της tank base, κάτι που υποδηλώνεται και από τη μέθοδο add_child, σε όλη την έκταση του κώδικα του αρχείου αυτού.

- Η χρήση του factory pattern, μας επιτρέπει να δημιουργήσουμε εύκολα διαφορετικούς τύπους tanks με συγκεκριμένα χαρακτηριστικά, καθώς και σχέσεις μεταξύ των μερών του tank (base και turret).

10 ΠΡΟΚΛΗΣΕΙΣ & BUGS

Κατά τη διάρκεια υλοποίησης του application, όπως και σε κάθε δημιουργική διαδικασία, φυσικά και εμφανίστηκαν κάποιες δυσκολίες, κυρίως όσον αφορά το συντονισμό Server και Client. Ειδικότερα, λόγω της αρχιτεκτονικής του παιχνιδιού, οποιαδήποτε μικρή αλλαγή στον κώδικα του ενός, σήμαινε αλλαγή και στον κώδικα του άλλου, με αποτέλεσμα να γίνει αρκετά πιο πολύπλοκη οποιαδήποτε απόπειρα τροποποίησης του κώδικα, παρότι είναι καλά δομημένος.

Μεγάλη πρόκληση αποτέλεσε και ο συντονισμός και καταμοιρασμός των tasks των μελών ασύγχρονα, η οργάνωση των ομαδικών meetings και γενικότερα, η χρονική δέσμευση που απαιτεί ένα τέτοιο μεγαλόπνοο πλάνο από άτομα που παράλληλη εργάζονται full time. Σε αυτό το σημείο, ας αναφέρουμε ότι, τελικά, ένα project που απαιτούσε την απόλυτη δέσμευση οπωσδήποτε τριών ατόμων, κατέληξε να υποστηρίζεται μόνο από ένα άτομο. Κάτι που, σε συνδυασμό και με το χρονικό περιορισμό, έπαιξε καθοριστικό ρόλο στη διαδικασία υλοποίησης του. Παρόλα αυτά, μέσα από την ομαδική συνεργασία, μάθαμε χρήσιμες πληροφορίες και τακτικές για τις δυνατότητες της Pygame και εξελιχθήκαμε όσον αφορά τις προγραμματιστικές μας γνώσεις. Θα μπορούσε κανείς να πει ότι «γνωρίσαμε νέους κόσμους».

Ένα bug που παρατηρήθηκε κάποιες φορές είναι η αποτυχία μείωσης των health points, αλλά και adding του score. Ενδεχομένως, αν αφιερώναμε ακόμη περισσότερο χρόνο, να λυνόταν, με τη διαμόρφωση επικοινωνίας μεταξύ Client και Server, με σκοπό να γίνεται update του health state, χρησιμοποιώντας κάποιο socket.

Επειδή το αρχικό πλάνο περιλάμβανε και ένα mode όπου τα τανκς είναι συνεργάτες, μένει να υλοποιηθεί το collision της σφαίρας με το, πλέον αντίπαλο μόνο, τανκ, ώστε να εμφανίζεται και στον αντίπαλο ότι τον πυροβολούμε, αλλά και να συγχρονιστεί με το health, που βλέπει ο server, ώστε να το στέλνει πίσω στα clients σαν πληροφορία και να φαίνεται η μείωση. Διότι η μείωση health και η αύξηση score, προβλέπονται εντός του κώδικα, απλά μένει ο συγχρονισμός τους με τον server. Ενδέχεται να χρειαστεί τροποποίηση όσον αφορά το bullet.

ΕΝΟΤΗΤΑ 3η

11 ΙΣΤΟΡΙΚΟ ΥΛΟΠΟΙΗΣΗΣ

Από την αρχική ημερομηνία ανάθεσης, το πρώτο βήμα ήταν η απόφαση του concept και των mechanics του παιχνιδιού. Ομολογουμένως, λόγω και μιας άγνοιας κινδύνων (υπό την έννοια του τι είναι εφικτό, υλοποιήσιμο, χρονοβόρο, σύνθετο ή όχι τόσο όσον αφορά την ίδια την Python και κατ' επέκταση την Pygame), η έννοια του concept μας απασχόλησε ιδιαίτερα. Και αυτό επειδή, δεχτήκαμε την πρόκληση και των δώδεκα ζητούμενων της εργασίας. Των 7 ελάχιστων απαιτήσεων και των υπόλοιπων 5 επιπλέον επιθυμητών.

Παράλληλα με τον περιορισμένο χρόνο που είχαμε, ξεκινήσαμε από στοιχειώδη μελέτη της Python, του περιβάλλοντος εργασίας μας (Visual Studio Code), φτάνοντας μέχρι και πολύ προχωρημένη μελέτη, όσον αφορά τις βάσεις δεδομένων και την ικανότητα συνδεσιμότητας σε διαδίκτυο.

Κάθε κομμάτι του κώδικα, προέκυψε από trial and error, αμέτρητες αναζητήσεις στο google και το stackOverflow, αλλά και από επίσης άπειρους διαλόγους με το https://chat.openai.com/ . Ιδιαίτερη χρησιμότητα είχε και στη διαδικασία του debugging. Μεγάλη βοήθεια στάθηκαν και κάποια video tutotrials, στην πλατφόρμα του youtube.

Βέβαια, η διαδικασία φιλτραρίσματος από τη γιγάντια υπερπληθώρα πληροφοριών και απόσπασης κάποιας χρηστικής οδηγίας, που πράγματι να λειτουργούσε στο περιβάλλον και τη δομή του παιχνιδιού, αποτέλεσε άθλο. Δηλαδή μεγάλο κομμάτι του project αυτού αποτέλεσε και η εκμάθηση της Python, ειδικότερα της Pygame και οι πειραματισμοί με το τι δυνατότητες υπάρχουν.

Κάθε φορά , όμως, που κάποιος συνδυασμός πληροφοριών με κάτι που είχαμε ήδη γράψει, λειτουργούσε, η χαρά μας ήταν απερίγραπτη. Αυτό ακριβώς μας ώθησε να συνεχίσουμε φτάνοντας το application σε ένα σχεδόν ολοκληρωμένο σημείο, παρέχοντας του χαρακτηριστικά και δυνατότητες επέκτασης και για περαιτέρω εξέλιξη.

12 ΠΕΡΑΙΤΕΡΩ ΕΞΕΛΙΞΗ

Σχετικά με τις δυνατότητες βελτίωσης του project, μπορεί εύκολα να παρατηρήσει κάποιος, ότι καθ' όλη την έκταση του κώδικα τόσο στο server όσο και στον client, υπάρχουν κάποια half implemented features (όπως η κλάση manager στο Server), τα οποία ενδεχομένως θα μπορούσαν να εμπλουτιστούν μελλοντικά για την αναβάθμιση της εφαρμογής.

Πιθανές βελτιώσεις που θα μπορούσαν να γίνουν αφορούν: την ενσωμάτωση μουσικής υπόκρουσης, βελτιστοποίηση των γραφικών, επιδιορθώσεις σε δύο bugs του κώδικα (μείωση health και καταμέτρηση score όχι μόνο τοπικά αλλά και στον server) και τη διερεύνηση του mode όπου τα 2 τανκς είναι συμπαίκτες αντί για αντίπαλοι και αντιμετωπίζουν sprites εχθρών που κάνουν spawn οπουδήποτε μέσα στο ταμπλό.

Τέλος, χάρη στην οργανωμένη δομή του κώδικα, τα σχόλια που εμπεριέχονται σε κάθε αρχείο και την αρχιτεκτονική του, μπορεί εύκολα να υποστεί επεξεργασία από τρίτους. Γενικότερα, σε ένα επίπεδο πιο προχωρημένο και ίσως και επαγγελματικό, υπάρχουν οι βάσεις για εξέλιξη του έργου. Άλλωστε, σε τέτοιου τύπου projects, συνήθως υπάρχει βελτίωση, όσο περισσότερο διάστημα ασχοληθούν οι δημιουργοί. Το έναυσμα δόθηκε. Από εκεί και πέρα, οι δυνατότητες είναι απεριόριστες, αν κάποιος απλά αφοσιωθεί αποκλειστικά σε αυτή την υλοποίηση.

13 ΣΥΜΠΕΡΑΣΜΑΤΑ

Ολοκληρώνοντας την παρουσίαση και περιγραφή του παιχνιδιού "Tanks!" που υλοποιήσαμε, προκύπτουν κάποιες συμπερασματικές σκέψεις.

Η Pygame, αλλά και η ίδια η Python, αποτελεί ένα πολύτιμο εργαλείο για τον εν δυνάμει game developer, λόγω της ευκολίας που παρέχεται στην εκμάθηση και κατανόηση του κώδικα, των βιβλιοθηκών και των εντολών. Παρόλα αυτά, ενδείκνυται περισσότερο για 2D παιχνίδια. Παραμένει ωστόσο κατάλληλη για την ομαλή ένταξη αρχάριων στα βαθιά concepts του προγραμματισμού και συγκεκριμένα του game developing.

Κρίνοντας επίσης από την πολυπλοκότητα και το χρόνο που απαιτήθηκε για την αρχιτεκτονική Client Server, είναι ένα project που απαιτεί πλήρη ενασχόληση μαζί του, αποκλείοντας ενδεχομένως άλλες υποχρεώσεις.

Παρόλα αυτά, η ενασχόληση με το project αυτό, αποτέλεσε ένα καλό εναρκτήριο λάκτισμα, για τη βαθύτερη ενασχόληση με τον προγραμματισμό και κατανόηση βασικών εννοιών, ειδικότερα τον κλάδο κατασκευής παιχνιδιών. Σίγουρα έχουμε πολλά ακόμη να μάθουμε και, όπως αναμένεται με κάθε δημιουργική διαδικασία, τα λάθη, τα bugs, οι δυσκολίες, είναι κομμάτι της διαδρομής αυτής και όχι ανασταλτικός παράγοντας.

Μέσα στις δυνατότητες που προκύπτουν από τη δημιουργία του εν λόγω application, προκύπτει και η παιδαγωγική σημασία του εγχειρήματος. Με τη βοήθεια μιας αρκετά απλής γλώσσας προγραμματισμού, την οποία μπορούμε να δείξουμε και σε μαθητές (άλλωστε η Python ήδη διδάσκεται στο μάθημα Προγραμματισμός Η/Υ του Γ ΕΠΑΛ), έχουμε τη δυνατότητα να δημιουργήσουμε πολλών ειδών παιχνίδια με επιμορφωτικό χαρακτήρα. Επομένως, υπάρχει διπλή παιδαγωγική σημασία. Πρώτον ως κάτι που μπορεί να δείξει ένας εκπαιδευτικός στο μαθητή πως να το κάνει και δεύτερον, ως έτοιμη

Εν κατακλείδι, ακόμη και αν το παιχνίδι έμεινε εν μέρει ελλιπές και κάποιοι αρχικοί στόχοι εν τέλει δεν έγιναν ακόμη εφικτοί (με τους κώδικες που αναφέρθηκαν πιο πάνω), τα ακόλουθα απαιτούμενα έχουν υλοποιηθεί ως εξής.

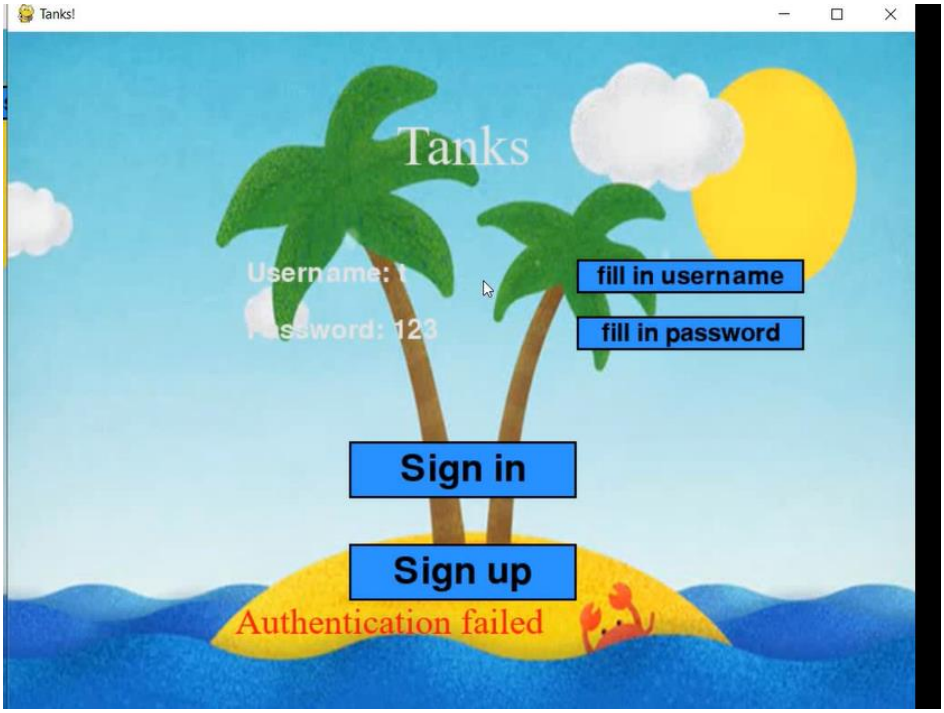
Ελάχιστες απαιτήσεις:

1. Ύπαρξη τουλάχιστον ενός παίκτη, για την ακρίβεια δύο παικτών με διαφορετικά χαρακτηριστικά
2. Ύπαρξη ταμπλό – πίστας, όπου βρίσκονται ένα κάστρο και 2 τανκς.
3. Ύπαρξη τριών επιπέδων δυσκολίας (Easy, Medium, Hard)
4. Αποθήκευση σκορ για τον παίκτη και «ανέβασμα» στον server

5. Αποθήκευση του password και του username του παίκτη σε database.db
6. Ύπαρξη γραφικών, όπως είναι τα τανκς που δημιουργήθηκαν συνεργατικά και από τα τρία μέλη της ομάδας με χρήση του προγράμματος Adobe Photoshop, αλλά και τα background images στο lobby και στο ταμπλό.
7. Ύπαρξη δύο ήχων (mouse click και type letter)

Επιπλέον επιθυμητές λειτουργίες που έχουν υλοποιηθεί στον κώδικα της εφαρμογής:

1. Ύπαρξη multiplayer mode
2. Ύπαρξη βάσης δεδομένων
3. Πραγματικού χρόνου application
4. Ύπαρξη IPv4 για το multiplayer mode: πράγματι, η διεύθυνση του localhost είναι στην ουσία η IPv4 του κοινού δικτύου Wifi στο οποίο συνδέονται οι δύο παίκτες
5. Υπάρχουν κατάλληλοι έλεγχοι για τις εξαιρέσεις και προβληματικές καταστάσεις, όπως αναλύθηκε και στην παράγραφο ''Επεξήγηση του κώδικα & Λειτουργικότητα'', της παρούσας τεχνικής αναφοράς. Ένα τέτοιο παράδειγμα αποτελεί και ο έλεγχος των στοιχείων του client στο παράθυρο όπου γίνεται sign in και sign up. Όπως επιδεικνύεται και στο video capture του παιχνιδιού, αν προσπαθήσει να συνδεθεί παίκτης με λανθασμένα στοιχεία (που στην ουσία δεν είναι καταχωρημένα στο database των username και passwords των παικτών), τότε η εφαρμογή θα εμφανίσει το μήνυμα ''Authentication Failed'', μην επιτρέποντας στον παίκτη να συνδεθεί, μέχρι να καταχωρήσει τα σωστά στοιχεία ή μέχρι να κάνει register με νέα.



Συνεπώς, παρά τις λίγες ελλείψεις και bugs, το project αυτό αποτέλεσε ένα μεγαλόπνοο πλάνο για κάτι που μπορεί να εξελιχθεί με πολλούς διαφορετικούς τρόπους, ενώ παράλληλα στοχεύσαμε να υλοποιήσουμε όσο περισσότερα από τα ζητούμενα της ανάθεσης.

14 ΕΥΡΕΤΗΡΙΟ ΟΡΩΝ

Σε αυτό εδώ το παράρτημα, αναφέρονται εξειδικευμένες ορολογίες και έννοιες.

- Αρχείο JSON (JavaScript Object Notation): είναι μία δημοφιλής μορφή αρχείου, που χρησιμοποιείται για την αποθήκευση προσωρινών δεδομένων, όπως δεδομένα ιστοσελίδων ή διαδικτυακών εφαρμογών. Αποτελεί συνήθης τακτική, η χρήση αρχείων json για τη μετάδοση και παραλαβή δεδομένων (σε μορφή κειμένου) μεταξύ ενός server και μιας δικτυακής εφαρμογής. Τα αντικείμενα σε ένα αρχείο json, αντιπροσωπεύονται ως ζευγάρια name/value. Με την εντολή *import json* στην Python, μπορούμε να αρχίσουμε να χρησιμοποιούμε αρχεία json.
- Factory Pattern/Method: είναι ένα σχεδιαστικό πρότυπο, που χρησιμοποιείται κυρίως για τη δημιουργία αντικειμένων (objects), χωρίς να καθορίζουμε ακριβώς την κλάση του αντικειμένου που θα δημιουργηθεί. Συνήθως, δημιουργούμε μία γενική αφαιρετική κλάση, η οποία περιέχει τις μεθόδους που δημιουργούν αντικείμενα, συγκεκριμένες κλάσεις που την επεκτείνουν και αντιπροσωπεύουν τους τύπους των αντικειμένων, μια διεπαφή ή κλάση τύπου Factory που δηλώνει τη μέθοδο για τη δημιουργία αντικειμένων της διεπαφής και είναι υπεύθυνη για δημιουργία περιπτώσεων, συγκεκριμένες κλάσεις τύπου Factory που είναι υπεύθυνες για τη δημιουργία συγκεκριμένου τύπου αντικειμένου και ένα αρχείο τύπου client code, που καλεί τη μέθοδο factory για να δημιουργήσει objects instances, χωρίς να χρειάζεται να γνωρίζει μια συγκεκριμένη κλάση ή συγκεκριμένο αντικείμενο.
- Gameplay: η γενικότερη εμπειρία που βιώνει ένας παίκτης παίζοντας ένα παιχνίδι. Συνδέεται με στοιχεία όπως: τα mechanics του ίδιου του παιχνιδιού, τους κανόνες, τις προκλήσεις, τους στόχους, το flow και τη γενικότερη διαδικασία απορρόφησης και διάδρασης του παίκτη στον κόσμο του παιχνιδιού.
- Sprite: 2D εικόνα ή animation που ενσωματώνεται σε μια σκηνή. Συχνά τα sprites χρησιμοποιούνται για να απεικονίσουν χαρακτήρες, παίκτες, npc, αντικείμενα και γενικά οποιοδήποτε είδους γραφικό.
- Pygame: είναι ένα γκρουπ από modules της Python, το οποίο περιέχει ειδικές συναρτήσεις, μεταβλητές και κλάσεις, σχετικές με την ανάπτυξη παιχνιδιών χρησιμοποιώντας την προγραμματιστική γλώσσα Python
- Socket: αποτελεί δομικό στοιχείο για δικτυακές εφαρμογές, λειτουργώντας σαν κόμβος, στη διαδικασία αποστολής και παραλαβής δεδομένων. Δημιουργούνται σαν δίαυλοι επικοινωνίας, από τον client για να συνδεθεί στον server και από τον server σε αναμονή σύνδεσης του client. Από τη στιγμή που οριστικοποιηθεί μια επικοινωνία μεταξύ client και server, ανταλλάσσονται ελεύθερα δεδομένα μεταξύ των αντίστοιχων sockets τους.
- Property: μία μέθοδος που αποτελεί ενσωματωμένη συνάρτηση της Python, η οποία μας επιτρέπει να ορίσουμε getter, setter και deleter μεθόδους για γνωρίσματα κλάσεων. Με την property(), μπορούμε να χρησιμοποιήσουμε χαρακτηριστικά, τα λεγόμενα properties, για να τροποποιήσουμε την εσωτερική υλοποίηση, χωρίς να αλλάξουμε τη δημόσια API της κλάσης.
- API (Application Programming Interface): είναι servers που χρησιμοποιούνται για την ανάκτηση και αποστολή δεδομένων, με χρήση κώδικα. Συνηθέστερη χρήση τους είναι η ανάκτηση δεδομένων, μέσω ενός request. Για να μπορέσουμε να το χρησιμοποιήσουμε πρέπει στο command window να γράψουμε την εντολή pip install requests.

- UI (User Interface): συμβολίζει τη διεπαφή μεταξύ του παίκτη και της συσκευής από την οποία παίζει ένα παιχνίδι. Είναι ένα σύστημα με οπτικά ενδεχομένως και ηχητικά μέρη, που διευκολύνουν τον παίκτη να αλληλοεπιδράσει με το περιβάλλον του παιχνιδιού, κατά προτίμηση με τρόπο εύκολο και απλό. Όσο πιο πετυχημένη είναι η αλληλοεπίδραση αυτή τόσο μεγαλύτερο είναι το flow. Κάτι τέτοιο επιτυγχάνεται με χαρακτηριστικά όπως: καλά και ευχάριστα γραφικά, ανταποκρισιμότητα, φιλικό προς το χρήστη μενού (New Game, Resume, Pause), βύθιση στο narrative του παιχνιδιού κ.ά.
- Instance: στον αντικειμενοστραφή προγραμματισμό, το instance(περίπτωση ή στιγμιότυπο) είναι μια συγκεκριμένη διατύπωση ενός αντικειμένου. Αποτελεί abstract έννοια. Πιο συγκεκριμένα, αν έχουμε ένα γενικό πρότυπο για το τι χαρακτηριστικά πρέπει να αποτελούν ένα αντικείμενο, δηλαδή έναν τρόπο να το αντιπροσωπεύσουμε, όταν παίρνουμε το instance δίνουμε σε κάθε ένα από αυτά τα χαρακτηριστικά του αντικειμένου συγκεκριμένη τιμή και έτσι έχουμε ένα ''στιγμιότυπο''. Για παράδειγμα, αν η γενική ιδέα του αντικειμένου είναι ένα τραπέζι, τότε για να μιλήσουμε για ένα συγκεκριμένο τραπέζι πρέπει να προσδιορίσουμε τις διαστάσεις του, τη θέση του, το χρώμα του, το υλικό του κλπ. Αυτή η συγκεκριμενοποίηση είναι το instance.

Σας ευχαριστώ.