# Αλγοριθμική σκέψη

## Minions Blast

ΦΟΙΤΗΤΕΣ:

ΘΕΟΔΟΣΗΣ ΓΚΙΚΑΣ

ΔΗΜΗΤΡΑ-ΜΑΡΙΑ ΤΣΑΤΣΟΥΛΗ

ΚΑΘΗΓΗΤΕΣ: ΔΑΣΥΓΕΝΝΗΣ ΜΗΝΑΣ

ΠΛΟΣΚΑΣ ΝΙΚΟΣ

## Σκοπός του παιχνιδιού

Ο σκοπός του παιχνιδιού είναι ο παίκτης να μαζέψει μπανάνες για να ανεβάσει το σκορ αποφεύγοντας τις βόμβες όσο περισσότερο μπορεί με στόχο να ανέβει πρώτος στην λίστα με τα καλύτερα σκορ.

Ο παίκτης ξεκινάει το παιχνίδι παίρνοντας τον ρόλο ενός **μίνιον** και προσπαθεί να μαζέψει όσες περισσότερες **μπανάνες** μπορεί. Κάθε μονή μπανάνα πιάνει ένα πόντο ενώ κάθε τσαμπί από μπανάνες τρεις πόντοι. Ο κάθε παίκτης έχει πέντε ζωές πριν χάσει. Στόχος του παίκτη είναι να αποφεύγει τις βόμβες αλλιώς χάνει μια ζωή για κάθε βόμβα που τον χτυπάει. Οι μπανάνες και οι βόμβες πέφτουν από τον ουρανό προς το έδαφος και μετά εξαφανίζονται. Όταν ο παίκτης χάσει όλες τις ζωές του τότε τελειώνει και το παιχνίδι. Έχει τη δυνατότητα να γράψει το όνομα του για να αποθηκευτεί το σκορ του. Στην επιλογή για δύο παίκτες (multiplayer) πρέπει να χάσουν και οι δύο παίκτες για να τελειώσει το παιχνίδι.

Από αριστερά προς τα δεξιά:
Παίκτης 1, Παίκτης 2, μονή Μπανάνα, τσαμπί από Μπανάνες, βόμβα

## Λειτουργίες του παιχνιδιού

Όταν ξεκινάει το παιχνίδι εμφανίζεται το Κεντρικό Μενού.

Το Κεντρικό Μενού έχει 4 επιλογές.

Η πρώτη είναι “Start Singleplayer” η οποία είναι η λειτουργία παιχνιδιού για έναν παίκτη. Στην συγκεκριμένη λειτουργία ο παίκτης παίζει μόνος του για να πετύχει μεγαλύτερο σκορ.

Η δεύτερη επιλογή είναι “Start Multiplayer” η οποία είναι η λειτουργία παιχνιδιού για δύο παίκτες. Σε αυτή τη λειτουργία οι παίκτες ανταγωνίζονται μεταξύ τους για το μεγαλύτερο σκορ. Η διαφορά της λειτουργίας με έναν παίκτη είναι ότι και οι δύο παίκτες παίζουν ταυτόχρονα και τα αντικείμενα που πέφτουν είναι πληθύνονται σε σύγκριση με τη λειτουργία του ενός από παίκτη.

Η τρίτη επιλογή είναι “Show Highscores” κατά την οποία μας δίνεται η επιλογή να δούμε τα 5 καλύτερα σκορ ανά κατηγορία παιχνιδιού Singleplayer/Multiplayer.

Η τέταρτη επιλογή αποτελεί το “Exit Game” που είναι η έξοδος από το παιχνίδι.

Επιπροσθέτως, τα σκορ αποθηκεύονται σε μια sqlite database.

## Χειρισμός του παιχνιδιού

Singleplayer controls:

- Κίνηση: Left/Right arrows or A/D keys
- Άλμα: Space/W/Up arrow

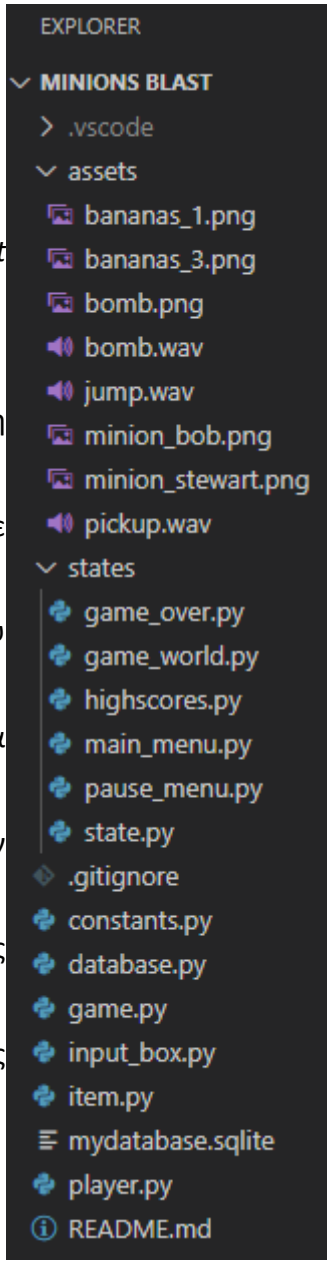
Multiplayer controls:

- Παίκτης 1:
  - Κίνηση: A/D
  - Άλμα: Space/W
- Παικτης 2:
  - Κίνηση: Left/Right arrows
  - Άλμα: Up arrow/Right Alt/Right Control

Common Controls

- Παύση: Esc

## Δομή αρχείων




Κεντρικός φάκελος *MinionsBlast*
Οι εικόνες και οι ήχοι βρίσκονται στον φάκελο *assets.*

Όλες οι διαφορετικες οθόνες (*states*) βρίσκονται στο φάκελο *states.*

Ως *State* ορίζουμε την κατάσταση που βρίσκεται το παιχνίδι τη συγκεκριμένη στιγμή. Στην ουσία αποτελούν τις διάφορες οθόνες του παιχνιδιού.

Στο state.py βρίσκεται η βασική κλάση με την δομή που θέλουμε μαζί με κάποιες έξτρα συναρτήσεις κοινές για όλες τις οθόνες.

Μέσα στο κεντρικό φάκελο έχουμε τα υπόλοιπα αρχεία κώδικα του παιχνιδιού.

Το game.py είναι το main αρχείο του παιχνιδιού, το οποίο εκτελείται για να ξεκινήσει το παιχνιδι.

Επίσης, το αρχείο .gitignore έχει μέσα τα ονόματα των αρχείων που δεν θέλουμε να ανεβαίνουν στο Αποθετήριο Git.

Επιπλέον, έχουμε το αρχείο που βρίσκεται η βάσης δεδομένων μας mydatabase.sqlite στο οποίο αποθηκεύονται τα σκορ μας.

Τέλος, υπάρχει και ένα αρχείο README.md το οποίο κατέχει λίγες πληροφορίες για το παιχνίδι.

## Επεξήγηση αρχείων

Το αρχείο **game.py** αποτελεί το αρχείο εκκίνησης του παιχνιδιού, στο οποίο αρχειοθετούνται οι παράμετροι και κατέχει πολλές βασικές συναρτήσεις που χρησιμοποιούνται σε όλες τις οθόνες. Στην περίπτωση που χρειαζόμαστε κάποια συνάρτηση, την καλούμε από διάφορες οθόνες και την βάζουμε μέσα στο αρχείο game.py ως συνάρτηση του αντικειμένου *Game,* το οποίο το δίνουμε ως όρισμα σε κάθε state. Επίσης, το αρχείο αυτό έχει την βασική λούπα του παιχνιδιού *game_loop* η οποία καλέιται με την σειρά *get_dt() get_events() update() render() clock.tick()***.**

- **get_dt:** μετράει το delta time δηλαδή τον χρόνο που πέρασε από το τελευταίο καρέ (frame).
- **get_events:** διαχειρίζεται όλα τα events του συστήματος και τα προωθεί στις οθόνες (state) όπου χρειάζονται.
- **update:** εδώ γίνονται όλοι οι υπολογισμοί του παιχνιδιού.
- **render:** εδώ δημιουργείται η σκηνή, όλα τα γραφικά.
- **clock.tick:** προχωράει το ρολόι όσα FPS του έχουμε πει.

Μία από τις βασικές δομές του παιχνιδιού είναι η χρήση μιας στοίβας στην οποία βάζουμε τις οθόνες που θέλουμε να δείξουμε κάθε φορά. Για παράδειγμα, με το που ξεκινάει το **game** βάζει στην λίστα το state του *MainMenu*. Αν πατήσουμε Start θα βάλει στην στοίβα το αντίστοιχο state. Με αυτόν τον τρόπο είναι εύκολο να πάμε στην προηγούμενη οθόνη γιατί θα βρίσκεται στην λίστα.

Το **game.py** δεν έχει game logic αλλά έχει μια βασική δομή που συναντάμε σε πολλά αρχεία. Η λογική αυτή έχει κάποιες κοινές συναρτήσεις που έχουν πολλά αρχεία: **handle_event, update**, **render.** Στο *game_loop* καλεί τις συναρτήσεις αυτές “πάνω” στην αντίστοιχη οθόνη που βρίσκεται πρώτη (πάνω πάνω) στην στοίβα.

Όλα τα αρχεία στο φάκελο *states* κληρονομούν την κλάση *State* και έχουν όλα τους την ίδια βασική δομή, δηλαδή **update(), render(), handle_event().** Την ίδια βασική δομή έχουν και οι κλάσεις **Item, Player, InputBox** απλώς αυτές δεν κληρονομούν κάποια άλλη κλάση.

Το αρχείο **input_box.py** έχει τον κώδικα για την δημιουργία κουτιού εισαγωγής κειμένου από τον χρήστη, πχ. για να βάλει το όνομα του στην αποθήκευση του σκορ. Έχει τις μεταβλητές και τις συναρτήσεις που χρειάζονται για να παίρνει την εστίαση (focus) με το ποντίκι και να μπορεί ο παίκτης να γράψει μέσα στο κουτί.

Το αρχείο **item.py** έχει τον κώδικα για τα αντικείμενα που πέφτουν από τον ουρανό, μπανάνες και βόμβες. Κρατάει τις μεταβλητές που χρειάζονται για την λειτουργία τους. Ενδεικτικά x,y τοποθεσία, το τετράγωνο (Rect) για της συγκρούσεις, την εικόνα του αντικειμένου, την ταχύτητα που πέφτει και το σκορ που δίνει, το αρνητικό σκορ αντιστοιχεί στη ζωή που χάνει ο παίκτης. Επίσης, παρέχει και τη λογική που έχουν τα αντικείμενα, τις κινήσεις τους, το πως θα ζωγραφιστούν στην σκηνή και από το σημείο που θα ξεκινήσουν να πέφτουν και με τι ταχύτητα.

Το αρχείο **player.py** έχει τον κώδικα του παίκτη, τις μεταβλητές του, όπως εικόνα, τοποθεσία, τετράγωνο (Rect) για τις συγκρούσεις, τις ζωές τους, το σκορ του, το πόσο ψηλά πηδάει και τους ήχους που ακούγονται όταν ο παίκτης πηδάει, μαζεύει μπανάνες ή

πέφτει πάνω σε βόμβα. Επιπλέον, διαθέτει και τη λογική που έχει ο παίκτης, τις κινήσεις που μπορεί να κάνει και το πως θα παρουσιαστεί στη σκηνή.

Το αρχείο **constants.py** περιέχει μερικές σταθερές (χρώματα, μεγέθη κ.α.) όπως και δύο *Enum* classes, οι οποίες αποτελούν βοηθητικές για τον κώδικα.

Το αρχείο **database.py** περιέχει οτιδήποτε έχει σχέση με database operation με sqlite3, όπως τη δημιουργία σύνδεσης, τη δημιουργία του πίνακα με τα highscores, τη προσθήκη του σκορ και το διάβασμα των σκορ.

## Οθόνες (States)

Έχουμε 5 οθόνες συνολικά. Το κεντρικό μενού *MainMenu*, τον κόσμο του παιχνιδιού *GameWorld*, το μενού παύσης *PauseMenu* που εμφανίζεται όταν ο παίκτης πατήσει το κουμπί της παύσης την ώρα που βρίσκεται στο GameWorld, την οθόνη της ήττας *GameOver* που εμφανίζεται όταν τελειώσει το παιχνίδι και την οθόνη που δείχνει τα καλύτερα σκορ *Highscores* .

Η οθόνη *PauseMenu* και *GameOver* εμφανίζονται πάνω από την εικόνα του κόσμου *GameWorld.* Αυτό επιτυγχάνεται με την χρήση της στοίβας καλώντας την συνάρτηση **render()** της προηγούμενης (δεύτερης) οθόνης στην στοίβα και μετά την **render()** της τωρινής οθόνης. Επίσης το παιχνίδι δείχνει σταματημένο επειδή δεν καλούμε την **update()** της προηγούμενης οθόνης που τρέχει όλο το logic του παιχνιδιού.

Σε όλα τα κουμπιά ζωγραφίζονται από πίσω τους ένα τετράγωνο σαν background, το οποίο αλλάζει χρώμα όταν το ποντίκι είναι από πάνω του για να δίνει την αίσθηση ότι πατιέται. Αυτό επιτυγχάνεται με τη βοήθεια μιας συνάρτησης στο αρχείο **state.py draw_button_background()** που κοιτάει της θέση του ποντικιού πάνω στην σκηνή.

Σε αυτό το σημείο, οφείλουμε να αναφέρουμε ότι για να πάρουμε την θέση του ποντικιού πρέπει να την μεταφράσουμε στο *game_canvas* μας. Στο **game.py** φτιάχνουμε δύο pygame.Surface, ένα *game_canvas* και ένα *screen, ο οποίο* είναι συνήθως μεγαλύτερο από το πρώτο. Η λογική του παιχνιδιού μας είναι στημένη πάνω στο *game_canvas* με ανάλυση 1280x720. Με αυτή τη λογική φτιάξαμε το παιχνίδι σε μία συγκεκριμένη ανάλυση και μετά το κάναμε scale στην οθόνη μας (*screen)*, για να φαίνονται όλα μεγαλύτερα.

Το αρχείο **main_menu.py** περιέχει την δημιουργία του κεντρικού μενού με τις επιλογές:

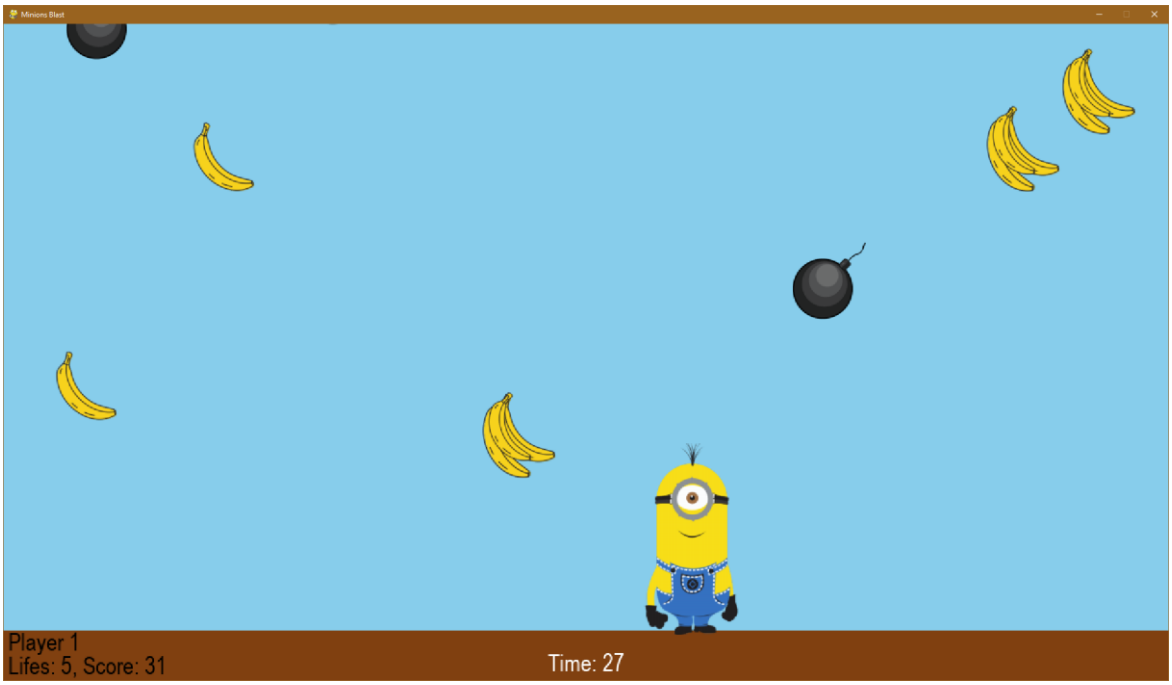
- Start Singleplayer εκκίνηση λειτουργίας παιχνιδιού με έναν παίκτη
- Start Multiplayer εκκίνηση λειτουργίας παιχνιδιού με δύο παίκτες
- Show Highscores εμφάνιση οθόνης με τα καλύτερα 5 σκορ απο την λειτουργία με έναν παίκτης και τα καλύτερα 5 σκορ απο την λειτουργία με δύο παίκτες.
- Exit Game έξοδος από το παιχνίδι

Η επιλογές σε όλα τα μενού γίνεται με την χρήση του ποντικιού. Επίσης, σε όλα τα μενού κεντράρονται οι επιλογές τους στο κέντρο της σκηνής.

Επιπροσθέτως, το κεντρικό μενού εμφανίζει το όνομα του παιχνιδιού στην κορυφή με μεγάλα γράμματα καθώς δείχνει δεξιά και αριστερά τα μίνιονς, τα οποία διαφοροποιούνται ανάλογα με ποιον τρόπο θα επιλέξει ο παίκτης να παίξει. Όλη η σκηνή έχει ένα πράσινο φόντο και το κάτω μέρος το έχουμε ζωγραφίσει με ένα καφέ χρώμα σαν έδαφος ώστε να μην φαίνονται τα μίνιονς ότι στέκονται στον αέρα.

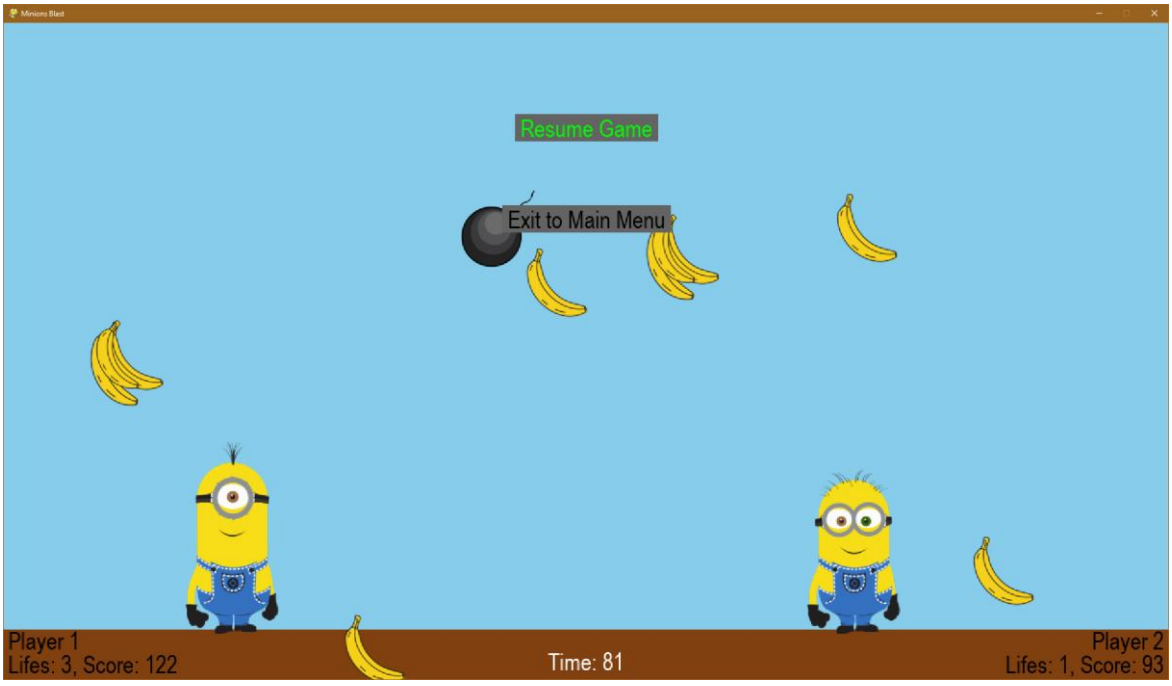
Κεντρικό Μενού

Το αρχείο **game_world.py** είναι ο κόσμος του παιχνιδιού. Δημιουργεί τους παίκτες τα αντικείμενα που πέφτουν και κρατάει και πόση ώρα παίζετε το παιχνίδι σε δευτερόλεπτα. Επίσης έχει την λογική για τις συγκρούσεις μεταξύ των παικτών και τον αντικειμένων. Οι παίκτες μεταξύ τους δεν συγκρούονται ούτε μπορούν να βγουν εκτός σκηνής. Ακόμα στην σκηνή ζωγραφίζεται και η διεπαφή το υ χρήστη. Την ζωή και το σκορ του κάθε παίκτη στις κάτω γωνίες και στο κέντρο κάτω πόσα δευτερόλεπτα έχουν περάσει από την εκκίνηση του κόσμου. Σε όλο το κάτω μέρος της σκηνής έχουμε ζωγραφίσει και εδώ με το ίδιο καφέ το έδαφος ώστε να μην “πετάνε” τα μίνιονς.



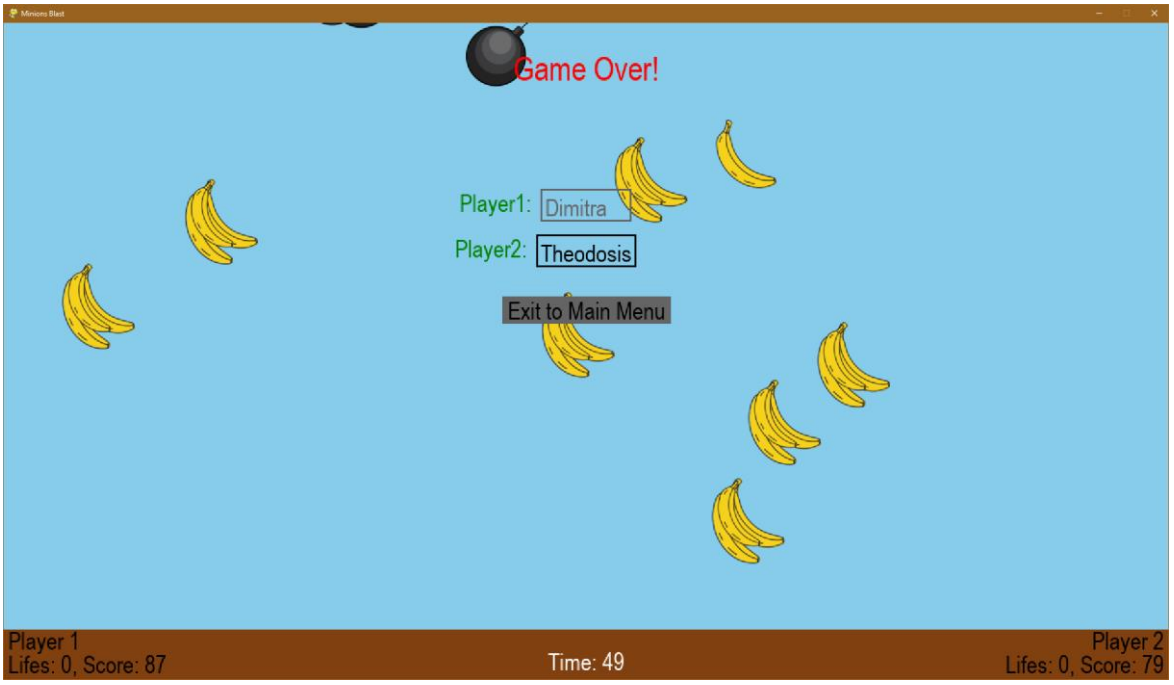
Ο κόσμος του παιχνιδιού σε λειτουργία για έναν παίκτη

Το αρχείο **pause_menu.py** περιέχει τον κώδικα για το μενού της παύσης που εμφανίζεται όταν ο παίκτης πατάει το κουμπί της παύσης κατά την διάρκεια του παιχνιδιού. Όπως είναι προφανές κάνει παύση του κόσμου του παιχνιδιού και εμφανίζει δύο επιλογές 'Resume Game' και 'Exit to Main Menu'. Είτε ο παίκτης ξαναπατήσει το κουμπί της παύσης είτε το κουμπί 'Resume Game' θα φύγει από το μενού παύσης και θα συνεχίσει το παιχνίδι από εκεί που έμεινε. Άμα επιλέξει το κουμπί 'Exit to Main Menu' θα επιστρέψει αμέσως στο κεντρικό μενού και θα χάσει την πρόοδο του από το παιχνίδι που έπαιζε.



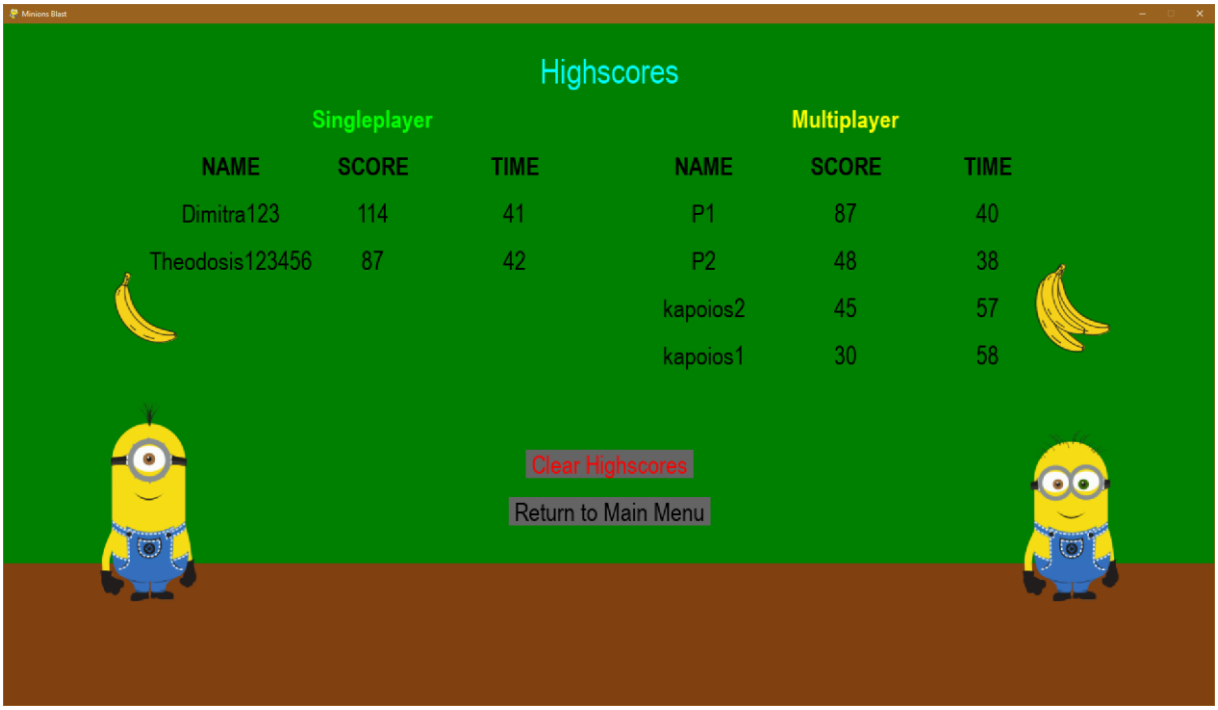
Το μενού παύσης παιχνιδιού κατά την λειτουργία για δύο παίκτες

Το αρχείο **game_over.py** περιέχει τον κώδικα για την οθόνη της ήττας (Game Over), η οποία μοιάζει με την οθόνη παύσης διότι παρατηρούμε ότι το παιχνίδι έχει σταματήσει αλλά με την διαφορά ότι λείπουν τα μίνιονς, αφού ο παίκτης έχει χάσει. Επιπλέον. βλέπουμε ότι εμφανίζεται στην κορυφή της οθόνης ένα μεγάλο κόκκινο 'Game Over'. Επίσης, σε αυτό το σημείο ο παίκτης έχει το δικαίωμα να γράψει το όνομα του στο αντίστοιχο κουτί έτσι ώστε να αποθηκευτούν τα σκορ στην βάση δεδομένων. Με το που θα πατήσει ο παίκτης την επιλογή 'Exit to Main Menu' θα πραγματοποιηθεί η αποθήκευση των σκορ στην βάση δεδομένων μαζί με το όνομα που εισήγαγαν, τον χρόνο που ήταν ζωντανός ο κάθε παίκτης και αν ήταν σε λειτουργία με έναν ή δύο παίκτες. Η αποθήκευση στην βάση δεδομένων γίνεται με την βοήθεια του αρχείου **database.py**.




Η οθόνη ήττας μετά το τέλος του παιχνιδιού στην λειτουργία για 2 παίκτες. Επίσης εδώ βλέπουμε και τα κουτάκια εισαγωγής ονομάτων των παικτών.

Το αρχείο **highscores.py** περιέχει τον κώδικα για την οθόνη των καλύτερων σκορ. Παρατηρούμε σε αυτή την οθόνη ομοιότητες με την οθόνη του κεντρικού μενού, το φόντο τα μίνιονς. Εδώ, ο παίκτης μπορεί να δει τα 5 καλύτερα σκορ ανά λειτουργία παιχνιδιού. Επίσης, εδώ ο παίκτης έχει δύο επιλογές τη 'Clear Highscores' και τη 'Return to Main Menu'. Με την πρώτη επιλογή διαγράφει όλα τα σκορ από την βάση δεδομένων. Με την δεύτερη επιλογή επιστρέφει στο κεντρικό μενού. Το διάβασμα και η διαγραφή των σκορ γίνεται από την βάση δεδομένων με την βοήθεια του αρχείου **database.py**.



Η οθόνη με τα 5 καλύτερα σκορ ανά λειτουργία παιχνιδιού

# Επεξήγηση κώδικα

**game.py**

Εδώ έχουμε την κλάση **Game**

Στον κατασκευαστή της κλάσης (__init__) αρχικοποιούμε το pygame το clock θέτουμε το όνομα του παραθύρου στο όνομα του παιχνιδιού μας “Minions Blast”. Μετά θέτουμε το πλάτος και μήκος του παιχνιδιού μας σε 1280x720, αυτό είναι και το game_canvas μας που ζωγραφίζουμε όλα τα γραφικά μας πρώτα πάνω εκεί και έπειτα το μεγαλώνουμε (scale) στην οθόνη μας screen, την οποία επιλέγουμε σύμφωνα με την οθόνη του εκάστοτε υπολογιστή αλλά με τις σωστές αναλογίες απο το game_canvas. Έπειτα αρχικοποιούμε μερικές μεταβλητές. Οι σημαντικότερες είναι:

- Η running μας δείχνει ότι το παιχνίδι μας τρέχει την βασική επανάληψη μας game_loop, αν γίνει False το παιχνίδι κλείνει.
- Η multiplayer δείχνει άμα παίζουμε την λειτουργία για δύο παίκτες.
- Η actions είναι μια λίστα με όλες τις ενέργειες που μπορεί να κάνει ο χρήστης. Αρχικοποιούμε όλα τα κουμπιά σε False τα οποία χρησιμοποιούνται για την κίνηση των παικτών ή την παύση.
- Η dt, prev_time είναι για τον υπολογισμό του delta time, δηλαδη του χρόνου που πέρασε από το τελευταίο καρέ.
- Η state_stack είναι η στοίβα που αποθηκεύουμε τις καταστάσεις που έχουμε περάσει και που βρισκόμαστε τώρα
- Η database είναι τύπου DatabaseSqlite που κρατάει την σύνδεση με την βάση δεδομένων και τις βοηθητικές συναρτήσεις που έχουμε φτιάξει για αυτήν.
- Φορτώνομαι τα διάφορα font μας
- Τα errors_update, errors_render κρατάμε τα συνεχόμενα σφάλματα που μπορεί να προκύψουν από τις αντίστοιχες μεθόδους **update()** και **render()** ώστε αν συμβούν παραπάνω από 2 συνεχόμενα σφάλματα να τον επιστρέψουμε στην αρχική οθόνη καλώντας της συνάρτηση του **restart().**

Τέλος, βάζουμε την οθόνη MainMenu στης στοίβα μας.

Η συνάρτηση **game_loop** καλείται 60 φορές το δευτερόλεπτο και κάνει με την σειρά..

Υπολογισμός **delta time**, χειρίζεται τα συμβάντα (**events**), καλεί την συνάρτηση **update()** που κάνει τους υπολογισμούς της κάθε οθόνης, καλεί την συνάρτηση **render()** η οποία εκτελεί τον κώδικα που ζωγραφίζει την σκηνή σε κάθε οθόνη. Τέλος, καλεί την clock.tick(FPS) ώστε να κρατάει σταθερά τα FPS στα 60 που του έχουμε πει.

Οι υπόλοιπες συναρτήσεις της κλάσης είναι:

- **get_events** χειρίζεται τα πατήματα του πληκτρολογίου και αλλάζει τα αντίστοιχα actions σε True αν πατιέται το κουμπί ή False αν όχι. Επίσης, προωθεί το κάθε event στην πρώτη οθόνη που είναι στην στοίβα σε περίπτωση που το χρειάζεται.

- **scaled_mouse** μεταφράζει την τοποθεσία του ποντικιού από την μεγαλύτερη οθόνη *screen* στο μικρότερο *game_canvas* μας ώστε να μπορούμε να υπολογίσουμε αν το ποντίκι πατάει κάποιο κουμπί ή όχι.
- **update** είναι υπεύθυνη για να καλεί την update της οθόνη που είναι φορτωμένη.
- **render** είναι υπεύθυνη για να καλεί την render της οθόνη που είναι φορτωμένη.
- **restart** καθαρίζει την στοίβα και βάζει νέα MainMenu οθόνη.
- **get_dt** υπολογίζει το delta time.
- **draw_text** χρησιμοποιείται για να γράψει κείμενο πάνω στην οθόνη. Παίρνει ορίσμα το surface που θα πάει να γράψει, το κείμενο που θα γράψει, το χρώμα των γραμμάτων, την τοποθεσία x,y. Αν θα χρησιμοποιήσει το μεγάλο font,το κανονικό ή το bold. Επίσης δέχεται το όρισμα centerX άμα δείχνει αμα θέλουμε να το κεντράρει στο κέντρο του surface. Επίδης, δεχεται το pivot που θέλουμε να χρησιμοποιήσει για το νοητό τετράγωνο του κειμένου που θα φτιάξει, με αυτό ορίζουμε ποια γωνία ή κέντρο του τετραγώνου εννοούμε με τις συντεταγμένες x,y (topLeft, topRight, bottomLeft, bottomRight,center).
- **CenterRect** με αυτή την συνάρτηση κεντράρουμε ένα οποιοδήποτε τετράγωνο **Rect** στο κέντρο του άξονα X του παιχνιδιού μας και προαιρετικά το βάζει και στο ύψος που θα του πούμε.
- **makeText** με αυτή την συνάρτηση φτιάχνεται ένα surface με το κείμενο που θα του πούμε και το τετράγωνο αυτού και τα επιστρέφει και τα δυο.
- **load_assets** εδώ ορίζουμε τον φάκελο των assets και τα διάφορα fonts (γραμματοσειρές) που χρησιμοποιούμε.
- **load_states** εδώ φτιάχνει την οθόνη MainMenu και την βάζει στην στοίβα
- **reset_keys** εδώ απλώς μηδενίζουμε όλα τα actions
- **LoadImage** μια βοηθητική συνάρτηση για να φορτώνουμε μια εικόνα στο pygame δέχεται το όνομα του αρχείου και προαιρετικα το μέγεθος που θέλουμε να το μετατρέψει.
- **PlayMusic** μια βοηθητική συνάρτηση για να φορτώσουμε μουσική στο pygame, δέχεται το όνομα του αρχείου και προαιρετικα την ένταση που το θέλουμε. Επίσης, παίζει συνεχόμενα σε επανάληψη την μουσική που θα επιλέξουμε.
- **PrepareSound** μια βοηθητική συνάρτηση για να φορτώνουμε έναν ήχο στο pygame δέχεται το όνομα του αρχείου και προαιρετικά την ένταση που το θέλουμε, επιστρέφει ένα αντικείμενο τύπου **Sound.**

**item.py**

Εδώ έχουμε την κλάση **Item**

Στον κατασκευαστή της κλάσης (__init__) ορίζουμε τις ιδιότητες που θέλουμε να έχει το αντικείμενο μας μέσα στο παιχνίδι. Η **__init__** ορίζει την εικόνα των αντικειμένων, το σκορ που προκύπτει από τη συλλογή των μπανανών, την ταχύτητα με την οποία κινούνται τα αντικείμενα στην οθόνη και το μέγεθος που έχουν τα αντικείμενα του παιχνιδιού. Επιπλέον, τοποθετεί τυχαία την αρχική θέση του αντικειμένου πάνω από την σκηνή.

Οι συναρτήσεις της κλάσης είναι:

- **render** ζωγραφίζει την εικόνα του αντικειμένου στην οθόνη στο σημείο που βρίσκεται το τετράγωνο πάνω στο offset
- **update** είναι υπεύθυνη για την ενημέρωση του αντικειμένου σύμφωνα με την ταχύτητα του
- **is_below_level** ανιχνεύει αν το αντικείμενο έχει φύγει κάτω από την πίστα
- **clamp** περιορίζει την τιμή και τα όρια που του δίνουμε σε ένα μέγιστο και ένα ελάχιστο όριο

**player.py**

Εδώ έχουμε την κλάση **Player**

Στον κατασκευαστή της κλάσης (__init__) ορίζουμε τις ιδιότητες που θα έχει το μίνιον, δηλαδή ο παίκτης, ως προς την κίνηση του, την εικόνα, τις διαστάσεις, τις ζωές που έχει, την ταχύτητα του, καθώς ορίζει ποιος είναι παίκτης που παίζει (1 ή 2) και φορτώνει τους ήχους που ακούγονται όταν ο παίκτης πηδάει  ή ακουμπάει τα υπόλοιπα αντικείμενα του χώρου.

Οι συναρτήσεις της κλάσης είναι:

- **render** ζωγραφίζει την εικόνα του αντικειμένου στην οθόνη στο σημείο που βρίσκεται το τετράγωνο πάνω στο offset.
- **update** είναι υπεύθυνη για την ενημέρωση του αντικειμένου σύμφωνα με τις κινήσεις του παίκτη και περιορίζει τον χαρακτήρα στα όρια της οθόνης. Παίζει τον ήχο του άλματος όταν χρειάζεται.
- **addScoreOrDamage** υπολογίζει το σκορ και τις ζωές που απομένουν στον παίκτη. Επίσης παίζει τους ήχους που θα ακούγονται στην περίπτωση που ο παίκτης συλλέξει μια μπανανα ή ακουμπήσει μια βόμβα.

**database.py**

Εδώ έχουμε την κλάση **DatabaseSqlite**

Στον κατασκευαστή της κλάσης (__init__) ορίζουμε μια μεταβλητή της σύνδεσης με την sqlite και εκτελεί ένα query για να δημιουργήσει το πίνακα highscore αν δεν υπάρχει.

Οι συναρτήσεις της κλάσης είναι:

- **create_connection** δημιουργεί τη σύνδεση και είναι υπεύθυνη για τις συνθήκες της συνδεσιμότητας
- **execute_query** εκτελεί το ερώτημα που του θέτουμε
- **execute_read_query** επιστρέφει τα δεδομένα στη βάση
- **read_all_highscores** διαβάζει όλα τα σκορ και τα επιστρέφει με την σωστή κατάταξη, τα μεγαλύτερα σκορ πρώτα.
- **read_all_highscoresByType** διαβάζει όλα τα σκορ ανά κατηγορία (multiplayer) και περιορίζει τον αριθμό αποτελεσμάτων με όσα του πούμε
- **create_highscore** δημιουργεί το όνομα, το σκορ και το χρόνο του παίκτη
- **clear highscore** διαγράφει το σκορ που δημιουργείται και επιστρέφει τα δεδομέν στη βάση

**input_box.py**

Εδώ έχουμε την κλάση **InputBox,** η οποία είναι υπεύθυνη για την διαχείριση ενός "κουτιού" εισαγωγής κειμένου από τον χρήστη. Στην __init__ ορίζουμε της ιδιότητες που θέλουμε να έχει το κουτί εισαγωγής, οι οποίες είναι: ο μέγιστος αριθμός χαρακτήρων που επιτρέπεται να γράψει ο χρήστης, την τοποθεσία που θέλουμε να βρίσκεται και σε σχέση με ποιά γωνία του τετραγώνου (PIVOT), το ύψος που θέλουμε να έχει το κουτί καθώς και το ελάχιστο πλάτος του. Προαιρετικά δέχεται και το αρχικό κείμενο που θα έχει γραμμένο μέσα στο κουτί. Επίσης, έχουμε και μια μεταβλητή για το αν είναι ενεργό το κουτί (αν εχει το focus δηλαδη) ή όχι ώστε όταν είναι ενεργό να δέχεται εισαγωγή από το πληκτρολόγιο. Οι συναρτήσεις που περιέχει είναι οι εξής:

- **handle_event** ελέγχει αν το ποντίκι κάνει κλικ πάνω στο κουτί μας ώστε να γίνει ενεργό ή ανενεργό και άμα είναι ενεργό ότι πληκτρολογεί ο χρήστης μπαίνει μέσα στο κουτί.
- **render** εδώ ζωγραφίζει το κείμενο που περιέχει το κουτί εισαγωγής μας και επίσης του ζωγραφίζει και ένα τετράγωνο πλαίσιο γύρω γύρω ώστε να φαίνεται ότι είναι κουτί εισαγωγής κειμένου και όχι ένα απλό κείμενο.
- **update** εδώ υπολογίζει το πλάτος του κουτιού σε σχέση με το ελάχιστο πλάτος που του έχουμε δηλώσει και το κείμενο που έχουμε εισάγει, με αυτόν τον τρόπο όταν μεγαλώνει το κείμενο μεγαλώνει και το πλαίσιο του και επίσης το κεντράρει στην αρχική του τοποθεσία με βάση το πλάτος του.

state.py

Εδώ έχουμε την κλάση **State,** δεν χρησιμοποιείτε αυτούσια αλλά είναι ο σκελετός για τις άλλες κλάσεις που την κληρονομούν. Αυτές είναι GameOver, GameWorld, Highscores, MainMenu, PauseMenu. Στην __init__ δέχεται ως όρισμα το Game που το αποθηκεύει σε δικιά του μεταβλητή για χρήση αργότερα, επίσης έχει μια μεταβλητή prev_state όπου εκεί αποθηκεύει το προηγούμενο State για γρήγορη πρόσβαση. Οι συναρτήσεις της κλάσης είναι:

Οι **render, update, handle_event** είναι ο κενές εδώ, υπάρχουν γιατί ορίζουν τον σκελετό για της άλλες κλάσεις που προαναφέραμε.

- **enter_state** βοηθητική συνάρτηση για την εισαγωγή State στην στοίβα.
- **exit_state** βοηθητική συνάρτηση για την αφαίρεση State από την στοίβα.
- **draw_button_background** είναι μια συνάρτηση που χρησιμοποιούν όλες οι οθόνες για να ζωγραφίσουν το τετράγωνο φόντο πίσω από τα κουμπιά. Δέχεται ως όρισμα το surface που θα ζωγραφίσει πάνω του, τα δύο χρώματα των κουμπιών, ένα για όταν το ποντίκι είναι από πάνω και ένα για όταν δεν είναι. Τέλος δέχεται το τετράγωνο κουτί του κουμπιού για να ανιχνεύσει αν υπάρχει “σύγκρουση” με το ποντίκι. Εσωτερικά καλει την **scaled_mouse** ώστε να πάρει την σωστή τοποθεσία του ποντικιού και έπειτα ελέγχει αν “συγκρούεται” με το τετράγωνο του κουμπιού και ανάλογα ζωγραφίζει το φόντο του τετραγώνου με το κατάλληλο χρώμα.
- Η **scaled_mouse** εδώ απλώς καλεί την **scaled_mouse** από το αντικείμενο game.

**game_world.py**

Εδώ έχουμε την κλάση **GameWorld**

Στον κατασκευαστή της κλάσης (__init__) ενημερώνει το αντικείμενο **game** για το αν το παιχνίδι είναι multiplayer και μετά αρχικοποιεί της μεταβλητές που χρειάζονται. Πιο σημαντικά, το ύψος του εδάφους, τα πόσα αντικείμενα θα υπάρχουν ταυτόχρονα στην πίστα, φορτώνει τις εικόνες του παιχνιδιού (χαρακτήρες και αντικείμενα) και δημιουργεί όσους παίκτες (**Player**) χρειάζεται.

Οι συναρτήσεις της κλάσης είναι:

- **add_item** είναι υπεύθυνη για την δημιουργία των αντικειμένων (**Item**) και το είδος τους, σύμφωνα με την δυσκολία του παιχνιδιού, η οποία αυξάνεται όσο περνάει ο χρόνος.
- **update** είναι υπεύθυνη για τους υπολογισμούς του παιχνιδιού, ελέγχει αν πατήθηκε η παύση, καλεί την **update** των παικτών (**Player**) και τον αντικειμένων (**Item**) με τα κατάλληλα ορίσματα. Επίσης, ελέγχει αν χρειάζεται να καλέσει την **add_item** για την δημιουργία νέων αντικειμένων, όπως επίσης αν τα αντικείμενα αυτά ακουμπήσουν κάποιο παίκτη να του δώσουν σκορ ή να χάσει ζωη ή αν αυτα πέσουν κάτω από την πίστα για να τα εξαφανίσει. Τέλος, ελέγχει αν έχουν

πεθάνει όλοι οι παίκτες για να εμφανίσει την οθόνη GameOver και προχωράει τον χρόνο.

- **render** είναι υπεύθυνη για το σωστό ζωγράφισμα της σκηνής. Παίζει ρόλο με ποιά σειρά τα ζωγραφίζει γιατί τα πιο κάτω καλύπτουν τα παραπάνω όταν βρίσκονται το ένα πάνω στο άλλο. Πρώτα ζωγραφίζει το φόντο, ένα γαλάζιο για τον ουρανό και ένα καφέ για το έδαφος. Μετά καλεί την **render** για κάθε αντικείμενο (**Item**) έπειτα για τους παίκτες (**Player**) και τέλος την διεπαφή του χρήστη με την βοήθεια της συνάρτησης **RenderPlayerUI** και τον χρόνο του παιχνιδιού**.**
- **RenderPlayerUI** είναι υπεύθυνη για την δημιουργία της διεπαφής του χρήστη (**UI**). Δέχεται ως όρισμα την οθόνη και τον παίκτη που έχει να ζωγραφίσει. Ζωγραφίζει στις κάτω γωνίες τα στοιχεία του παίκτη, πόσες ζωές του απομένουν και το σκορ του. Κάτω αριστερά τον παίκτη 1 και κάτω δεξιά τον παίκτη 2.

game_over.py
Εδώ έχουμε την κλάση **GameOver**

Στον κατασκευαστή της κλάσης (__init__) δέχεται τους παίκτες από το **GameWorld** για να ορίσει την κατάσταση του παίκτη, δηλαδή το σκορ του, ότι χρειάζεται για να αποθηκεύσει το σκορ του.

Έπειτα οριστικοποιούμε μερικές μεταβλητές και δημιουργεί όσα κουτιά εισαγωγής κειμένου χρειάζονται. Οι συναρτήσεις της κλάσης είναι:

- **update** εδώ απλός καλεί την **update** απο τα **InputBox** και καλεί τη **reset_keys** του **game**.
- **render** εδώ ζωγραφίζει το κείμενο που εμφανίζεται όταν χάνει ο παίκτης καθώς ορίζει τις διαστάσεις και τα χρώματα του μηνύματος επιστρέφοντας δεδομένα στο game world
- **handle_event** καλεί τη **handle_event** των **InputBox** και όταν επιλέξουμε την επιλογή ‘Exit to Main Menu’, αποθηκεύει τα σκορ στην βάση δεδομένων με την βοήθεια του αντικειμένου **DatabaseSqlite.**

highscores.py
Εδώ έχουμε την κλάση **Highscores**

Στον κατασκευαστή της κλάσης (__init__) ορίζει για κάθε αντικείμενο την εικόνα του καθώς και τις διαστάσεις τους, καθώς μας διαβάζει και τα σκορ.

Έπειτα οριστικοποιούμε μερικές μεταβλητές. Οι συναρτήσεις της κλάσης είναι:

- **update** εδώ καλεί τη **reset_keys** του **game**
- **render** εδώ ζωγραφίζει σε δύο στήλες τα 5 καλύτερα σκορ. Μια στήλη για το Singleplayer και μια για το Multiplayer.

- **handle_event** ελέγχει αν το ποντίκι κάνει κλικ πάνω στις επιλογές που έχουμε, διαγραφεί όλων των σκορ ή επιστροφή στο κεντρικό μενού.
- **display_highscores** είναι υπεύθυνη για την ζωγραφική των σκορ. Παίρνει ως όρισμα την οθόνη που θα ζωγραφίσει, τα ονόματα των στηλών και τα αποτελέσματα από τα σκορ. Έπειτα ζωγραφίζει ένα πίνακα με τα σκορ με την πρώτη γραμμή να είναι τα ονόματα των στηλών.

**main_menu.py**

Εδώ έχουμε την κλάση **MainMenu**

Στον κατασκευαστή της κλάσης (__init__) ορίζει τις μεταβλητές του παιχνιδιού και τις κεντράρει στην οθόνη.

- **update** εδώ καλεί τη **reset_keys** του **game**
- **render** εδώ ζωγραφίζει τις επιλογές του παίκτη στο βασικό μενού με διαφορετικά χρώματα
- **handle_event** ελέγχει ποιο κουμπί έχει πατήσει το ποντίκι και δημιουργεί την κατάλληλη οθόνη (**State**).

pause_menu.py
Εδώ έχουμε την κλάση **PauseMenu**

Στον κατασκευαστή της κλάσης (__init__) ορίζει τις μεταβλητές του ‘Resume Game’ και του ‘Exit to Main Menu’, δηλαδή

- **update** εδώ ορίζει τις επιλογή του exit_state και καλεί τη reset_keys του game
- **render** εδώ ζωγραφίζει το background ανάλογα με την επιλογή του ‘Resume Game’ και του ‘Exit to Main Menu’
- **handle_event** ελέγχει ποιο κουμπί έχει πατήσει το ποντίκι και δημιουργεί την κατάλληλη συνθήκη

Τα παρακάτω αφορούν το branch online στο github, το οποίο μπήκε σε ξεχωριστό, διότι εντάχθηκε τελευταίο και δεν ξέραμε αν θα το προλάβουμε με σκοπό να μην χαλάσουμε το main branch.

## Multiplayer μέσω δικτύου (IPv4)

Η διαφορά με το τοπικό Multiplayer είναι ότι το παιχνίδι μας τρέχει δύο φορές οπότε πρέπει με κάποιο τρόπο να συγχρονιστούν μεταξύ τους τα παιχνίδια. Οπότε μέσω του δικτύου ανταλλάσουν δεδομένα. Το πιο σύνηθες μοντέλο είναι το client-server, όπου το ένα παιχνίδι παίρνει τον ρόλο του server και το άλλο τον ρόλο του client. Ο server φτιάχνει, σερβίρει, το παιχνίδι στον client, πελάτη. Για να γίνει ο συγχρονισμός πρέπει να φτιαχτούν αντικείμενα τα οποία μπορούν τα γίνουν serialized και να πιάνουν λίγα bytes έτσι ώστε να είναι γρήγορη η μεταφορά των δεδομένων στο δίκτυο. Έτσι στέλνουμε μόνο τα απολύτως απαραίτητα δεδομένα τα οποία είναι αρκετά για να αναπαραστήσει την ίδια σκηνή. πχ ο server μπορεί να στείλει έναν αριθμό για τον τύπο του αντικειμένου και με αυτό ο client να ξέρει ποια εικόνα θα επιλέξει για να την δείξει από ότι ο server να στείλει ολόκληρη την εικόνα.

Για αυτό τον λόγο φτιάχτηκε το αρχείο **network_items.py** που περιέχει 3 κλάσεις.

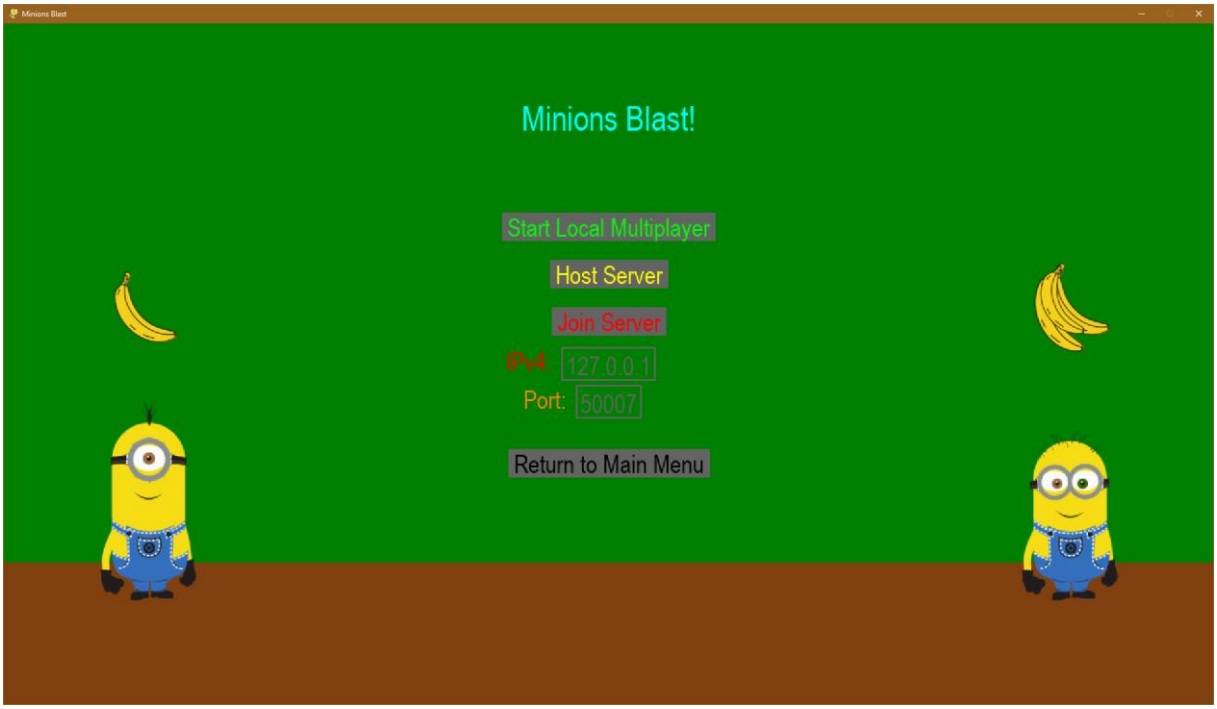
- **NetworkWorld** είναι το αντικείμενο που στέλνει ο server στον client για να φτιάξει τον κόσμο, επειδή εμάς ο κόσμος μας είναι πολύ απλός περιέχει μόνο τους παίκτες(**NetworkPlayer**), τα αντικείμενα (**NetworkItem**) μας (μπανάνες και βόμβες) και τον χρόνο του παιχνιδιού.
- **NetworkPlayer** περιέχει μόνο τις απαραίτητες μεταβλητές του **Player** που χρειάζονται για τον συγχρονισμό.
- **NetworkItem** περιέχει μόνο τις απαραίτητες μεταβλητές του **Item** που χρειάζονται για τον συγχρονισμό.

Επίσης, φτιάχτηκε ένα αρχείο για τον client και ένα για τον server, τα οποία είναι και αυτά τύπου State. Με αυτό τον τρόπο μπορούμε να φτιάξουμε το State **GameWorld** να το βάλουμε στην στοίβα και μετά να προσθέσουμε το αντίστοιχο **GameClient** ή **GameServer** ανάλογα με την επιλογή του χρήστη. Αυτά με την σειρά τους, πέρα από τις ενέργειές τους να καλούν και την **update** και την **render** του **GameWorld,** όπως και να επηρεάζουν τις μεταβλητές της. Έστι ο κόσμος του παιχνιδιού θα παίζει “πίσω” από το **GameClient** ή το **GameServer**.

## Επεξήγηση κώδικα

**multiplayer_menu.py**

Είναι η οθόνη μενού για το Multiplayer μέσω δικτύου (IPv4). Εδώ ο παίκτης έχει την επιλογή να παίξει τοπικό Multiplayer, να κάνει έναν server ή να συνδεθεί σε έναν. Επίσης έχει δύο κουτιά εισαγωγής κειμένου το πρώτο για την IPv4 του server που θέλει να συνδεθεί και το δεύτερο για την πόρτα του server. Τέλος έχει την επιλογή να επιστρέψει στο κεντρικό μενού.




Η οθόνη με το νέο (IPv4) multiplayer μενού

**game_client.py**

Εδώ έχουμε την κλάση **GameClient** το πρώτο που κάνει είναι να συνδεθεί με τον server που του έχουμε πει από το προηγούμενο μενού, αφού το κάνει με επιτυχία μετά μεταφράζει τον παίκτης του client (Player2) από τύπο **Player** σε **NetworkPlayer** και το στείλει στον Server σε κάθε επανάληψη. Όπως επίσης δέχεται σε κάθε επανάληψη ένα **GameWorld** από τον server. Από αυτό βρίσκει τον παίκτη του server (Player2) τον μεταφράζει σε **Player** από **NetworkPlayer** και ενημερώνει το τοπικό του αντίγραφο. Επίσης παίρνει όλα τα αντικείμενα τύπου **NetworkItem** τα μεταφράζει σε **Item** και μετά ενημερώνει το τοπικό πίνακα με τα **Item.** Άμα ο παίκτης πατήσει το κουμπί παύσης θα αποσυνδεθεί από τον server διότι δεν μπορείς να έχεις παύση σε έναν online παιχνίδι ή είναι πολύ δύσκολο.

**game_server.py**

Εδώ έχουμε την κλάση **GameServer** με το που ξεκινάει φτιάχνει έναν server που ακούει στην πόρτα που του έχουμε πει από το προηγούμενο μενού. Επίσης δείχνει στην οθόνη κάποιες χρήσιμες πληροφορίες όπως τις διευθύνσεις IPv4 του υπολογιστή που βρίσκεται και την πόρτα που είναι ανοιχτός ο server για ευκολία του χρήστη. Αφού συνδεθεί ο δεύτερος παίκτης ξεκινάει και του στέλνει τον κόσμο του παιχνιδιού σε κάθε επανάληψη. Φτιάχνει ένα αντικείμενο τύπου **NetworkWorld** και μεταφράζει τους **Player** σε **NetworkPlayer** και τα **Item** σε **NetworkItem** βάζει και το *current_time* και τα στέλνει στον client. Επίσης λαμβάνει από τον client το **NetworkPlayer** το μεταφράζει σε **Player** και ενημερώνει το τοπικό αντίγραφο του δεύτερου παίκτη. Άμα ο παίκτης πατήσει το κουμπί παύσης θα αποσυνδέσει τον client και θα κάνει παύση τον κόσμο του παιχνιδιού. Σε αυτή την περίπτωση ο παίκτης είτε θα περιμένει να ξανασυνδεθεί ένας client για τον δεύτερο παίκτη, ώστε να συνεχιστεί το παιχνίδι από εκεί που είχε μείνει είτε μπορεί να κάνει έξοδο στο κεντρικό μενού.

Στο τέλος του παιχνιδιού μέσω διαδικτύου και ο client και ο server θα δειχθεί η οθόνη GameOver για την αποθήκευση των σκορ.

Ακόμα για το multiplayer μέσω δικτύου έχει φτιαχτεί μια βοηθητική συνάρτηση στο **GameWorld,** την **extract_player** η οποία δέχεται τον αριθμό του παίκτη (1 ή 2) και ψάχνει να τον βρει στους ζωντανούς ή στους νεκρούς παίκτες του κόσμου και τον επιστρέφει.

# Βιβλιογραφία

Ήχοι που χρησιμοποιήθηκαν:

https://freesound.org/people/plasterbrain/sounds/399095/

https://freesound.org/people/BananaMilkshake/sounds/632701/

https://freesound.org/people/eardeer/sounds/402005/

Οι εικόνες είναι μια δημιουργία της Δήμητρας-Μαρίας Τσατσούλη εκτός από την εικόνα της βόμβας https://www.deviantart.com/raenko87/art/FREE-Simple-Bomb-Sprite-746596954