21. ΚΑΛΟΚΑΙΡΙΝΑ PROJECTS IV – ASTEROIDS GAME (Α'ΜΕΡΟΣ)

21.1.0 Εισαγωγή

Συνεχίζουμε τις καλοκαιρινές μας περιπέτειες, επεκτείνοντας όλο και περισσότερο τις γνώσεις μας στον προγραμματισμό, χωρίς ταυτόχρονα να παραμελούμε τη δημιουργικότητα και τη διασκέδαση.

Σήμερα, θα αναπτύξουμε το κλασικό παιχνίδι arcade, "Asteroids". Είναι πολύ πιθανό, αρκετοί από σας να το έχουμε παίξει, πολύ παλιότερα.

Μεταξύ άλλων, θα μάθουμε να φορτώνουμε εικόνες στην οθόνη μας, να χειριζόμαστε την είσοδο του χρήστη από το πληκτρολόγιο, να μετακινούμε αντικείμενα στην οθόνη, σύμφωνα με τη λογική του παιχνιδιού, να ανιχνεύουμε τη σύγκρουση των αντικειμένων, να δείχνουμε κείμενο στην οθόνη και να παίζουμε ήχους.

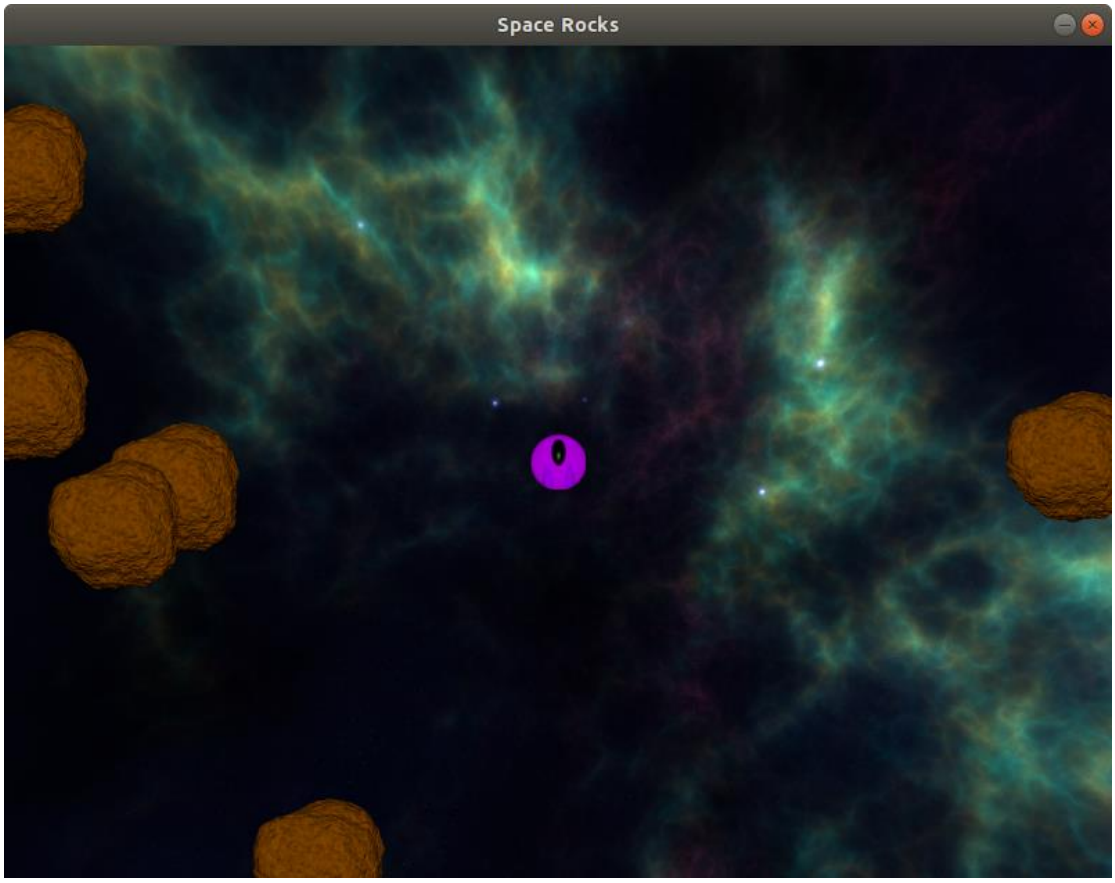
Ήδη έχουμε ασχοληθεί με όλα τα παραπάνω, όμως τώρα, θα τα φτιάξουμε όλα με ένα τρόπο πιο «επαγγελματικό» και ακόμα πιο εξειδικευμένο στη δημιουργία παιχνιδιών, χρησιμοποιώντας τη βιβλιοθήκη ***pygame***.

Ας την κατεβάσουμε λοιπόν. Ανοίξτε τη γραμμή εντολών των Windows, ή του λειτουργικού συστήματος που χρησιμοποιείτε και πληκτρολογήστε: ***pip install pygame***

Επίσης, κατά τη διάρκεια της δημιουργίας του παιχνιδιού, θα θυμηθούμε τη χρήση των κλάσεων, καθώς και άλλων σημαντικών πρακτικών προγραμματισμού, όπως τη δημιουργία ξεχωριστών αρχείων και την κλήση μεθόδων μέσα απ' αυτά, στη ροή του παιχνιδιού.

21.1.1 Περιγραφή του παιχνιδιού

Θα φτιάξουμε ένα κλώνο του παιχνιδιού Asteroids. Στο παιχνίδι, έχουμε ένα σκάφος το οποίο έχει σαν σκοπό να διαλύει αστεροειδείς που έρχονται προς το μέρος του. Αν ένας αστεροειδής αγγίξει το σκάφος, τότε χάνουμε. Το παιχνίδι μας, θα είναι όπως παρακάτω:



21.1.2 Επισκόπηση του παιχνιδιού

Το παιχνίδι θα διαθέτει ένα μόνο διαστημόπλοιο. Το διαστημόπλοιο μπορεί να περιστρέφεται, καθώς και να κινείται, αλλά και να ρίχνει σφαίρες. Οι σφαίρες, σπάνε κάθε αστεροειδή σε μικρότερα κομμάτια.

Το παιχνίδι θα παίζεται με τα παρακάτω πλήκτρα:

Πλήκτρο	**Ενέργεια**
Δεξί βέλος →	Περιστροφή του σκάφους δεξιά
Αριστερό βέλος ←	Περιστροφή του σκάφους αριστερά
Επάνω βέλος ↑	Επιτάχυνση του σκάφους προς τα εμπρός
Space	Πυροβολώ
Escape	Διακοπή του παιχνιδιού

Λεπτομερέστερα θα έχουμε:

- Θα υπάρχουν έξι μεγάλοι αστεροειδείς στο παιχνίδι. Όταν μια σφαίρα χτυπήσει έναν μεγάλο αστεροειδή, θα χωριστεί σε δύο μεσαίους.
- Όταν μια σφαίρα χτυπήσει έναν μεσαίο αστεροειδή, θα χωριστεί σε δύο μικρούς.
- Ένας μικρός αστεροειδής δεν θα χωριστεί αλλά θα καταστραφεί από μια σφαίρα.
- Όταν ένας αστεροειδής συγκρουστεί με το διαστημόπλοιο, το διαστημόπλοιο θα καταστραφεί και το παιχνίδι θα τελειώσει με ήττα.
- Όταν χτυπηθούν όλοι οι αστεροειδείς, το παιχνίδι θα τελειώσει με νίκη!

21.1.3 Σχεδιασμός του παιχνιδιού

Το έργο θα χωριστεί σε δέκα βήματα:

1. Ρύθμιση του Pygame για το έργο μας
2. Χειρισμός εισόδου του παίκτη στο παιχνίδι
3. Φόρτωση εικόνων και εμφάνισή τους στην οθόνη
4. Δημιουργία αντικειμένων παιχνιδιού με εικόνα, θέση και κάποια λογική
5. Μετακίνηση του διαστημοπλοίου
6. Μετακίνηση των αστεροειδών και ανίχνευση συγκρούσεων με το διαστημόπλοιο
7. Ρίξιμο σφαιρών και καταστροφή αστεροειδών
8. Διάσπαση των αστεροειδών σε μικρότερους
9. Αναπαραγωγή ήχων

10.Χειρισμός του τέλους του παιχνιδιού

21.1.4 Αρχή του παιχνιδιού

Ας σχολιάσουμε τον παρακάτω κώδικα:

1.initialize_pygame()

2.

3.while True:

4. handle_input()

5. process_game_logic()

6. draw_game_elements()

Η γραμμή 3 ξεκινά έναν βρόχο, που ονομάζεται **βρόχος του παιχνιδιού**. Κάθε επανάληψη αυτού του βρόχου δημιουργεί ένα μόνο πλαίσιο του παιχνιδιού και συνήθως εκτελεί τις ακόλουθες λειτουργίες:

1. **Χειρισμός εισόδων:** Συλλέγονται οι είσοδοι από το πληκτρολόγιο. όπως πλήκτρα που πατιούνται, η κίνηση του ποντικιού και στη συνέχεια γίνεται ο χειρισμός. Ανάλογα με το παιχνίδι, μπορεί να τα αντικείμενα να αλλάξουν τη θέση τους, να δημιουργήσουν νέα αντικείμενα, να ζητήσουν το τέλος του παιχνιδιού και ούτω καθεξής.
2. **Λογική του παιχνιδιού:** Εδώ υλοποιούνται οι περισσότεροι μηχανισμοί του παιχνιδιού. Εφαρμόζονται οι κανόνες της φυσικής, οι συγκρούσεις εντοπίζονται και αντιμετωπίζονται κ.ο.κ. Αυτό το μέρος είναι επίσης υπεύθυνο για τον έλεγχο εάν ο παίκτης έχει κερδίσει ή χάσει το παιχνίδι.
3. **Σχεδιασμός στοιχείων:** Εάν το παιχνίδι δεν έχει τελειώσει ακόμα, τότε εδώ θα σχεδιαστεί το πλαίσιο στην οθόνη. Θα περιλαμβάνει όλα τα αντικείμενα που βρίσκονται αυτήν τη στιγμή στο παιχνίδι και είναι ορατά στον παίκτη.

Η γενική δομή ενός παιχνιδιού Pygame δεν είναι περίπλοκη και πιθανότατα θα μπορούσαμε να ενεργήσουμε βάζοντας τα πάντα στον

παραπάνω βασικό βρόχο. Ωστόσο, λαμβάνοντας υπόψη ότι ενδέχεται να επεκτείνουμε το παιχνίδι στο μέλλον, αλλά και για την καλύτερη αναγνωσιμότητα του κώδικα και τη χρήση καλύτερων προγραμματιστικών πρακτικών, είναι καλή ιδέα να ενσωματώσουμε όλες αυτές τις λειτουργίες σε μια κλάση.

Η δημιουργία μιας κλάσης σημαίνει ότι πρέπει να επιλέξουμε ένα όνομα για το παιχνίδι. Έχουμε επιλέξει ήδη το "Asteroids" αλλά θα μπορούσαμε αν βάλουμε να το ονομάσουμε και “Space Rocks”.

Δημιουργούμε ένα αρχείο και το ονομάζουμε game.py. Εδώ θα βάλουμε την κύρια λειτουργία του παιχνιδιού μας.

Ο κώδικάς μας θα είναι:

```
1  import pygame
2
3  class SpaceRocks:
4      def __init__(self):
5          self._init_pygame()
6          self.screen = pygame.display.set_mode((800, 600))
7
8      def main_loop(self):
9          while True:
10             self._handle_input()
11             self._process_game_logic()
12             self._draw()
13
14     def _init_pygame(self):
15         pygame.init()
16         pygame.display.set_caption("Asteroids")
17
18     def _handle_input(self):
19         pass
20
21     def _process_game_logic(self):
22         pass
```

```
23
24      def _draw(self):
25          self.screen.fill((0, 0, 255))
26          pygame.display.flip()
```

Πάμε να σχολιάσουμε τον κώδικα:

- **Η γραμμή 1** εισάγει τη λειτουργική μονάδα Pygame για να αποκτήσουμε πρόσβαση σε όλες τις λειτουργίες της.
- **Η γραμμή 3** δημιουργεί την κλάση SpaceRocks.
- **Η γραμμή 4** είναι ο κατασκευαστής της κλάσης SpaceRocks και είναι το τέλειο μέρος για να τοποθετήσουμε όλες τις μεθόδους που απαιτούνται για την προετοιμασία του Pygame. Η πραγματική προετοιμασία του Pygame γίνεται στο _init_pygame(). Θα μάθουμε περισσότερα για αυτήν τη μέθοδο σε λίγο.
- **Η γραμμή 6** δημιουργεί μια επιφάνεια (screen) προβολής. Οι εικόνες στο Pygame αντιπροσωπεύονται από **επιφάνειες** . Εδώ είναι μερικά πράγματα που πρέπει να γνωρίζουμε γι' αυτές:
    - Οι επιφάνειες μπορούν να σχεδιαστούν η μία πάνω στην άλλη, επιτρέποντάς μας να δημιουργήσουμε σύνθουμες σκηνές από απλές εικόνες.
    - Υπάρχει μία ειδική επιφάνεια σε κάθε έργο Pygame. Αυτή η επιφάνεια αντιπροσωπεύει την οθόνη και είναι αυτή που τελικά θα εμφανίζεται στους παίκτες. Όλες οι άλλες επιφάνειες πρέπει να σχεδιαστούν σε αυτήν κάποια στιγμή. Διαφορετικά, δεν θα εμφανίζονται.
    - Για να δημιουργήσουμε την επιφάνεια της οθόνης, το πρόγραμμά μας χρησιμοποιεί τη γραμμή pygame.display.set_mode(). Το μόνο όρισμα που μεταβιβάζουμε σε αυτήν τη μέθοδο είναι το μέγεθος της οθόνης, που αντιπροσωπεύεται από μια πλειάδα δύο τιμών: πλάτος και ύψος. Σε αυτήν την περίπτωση, το Pygame θα δημιουργήσει μια οθόνη με πλάτος 800pixel και ύψος 600pixel.
- **Η γραμμή 8** είναι ο βρόχος παιχνιδιού που συζητήθηκε παραπάνω. Περιλαμβάνει τα ίδια τρία βήματα για κάθε πλαίσιο:
    - **Η γραμμή 10** περιέχει χειρισμό εισόδου.
    - **Η γραμμή 11** περιέχει τη λογική του παιχνιδιού.
    - **Η γραμμή 12** περιέχει τον σχεδιασμό.
- **Η γραμμή 14** ορίζει μια μέθοδο που ονομάζεται _init_pygame(). Εδώ συμβαίνει μια εφάπαξ προετοιμασία του Pygame. Η μέθοδος κάνει δύο πράγματα:

- Κλήση **στη γραμμή 15** pygame.init() . Αυτή η μοναδική γραμμή κώδικα είναι υπεύθυνη για τη ρύθμιση των χαρακτηριστικών του Pygame. Κάθε φορά που εργαζόμαστε με το Pygame, θα πρέπει να καλούμε την pygame.init() στην αρχή του προγράμματος για να βεβαιωνόμαστε ότι το πλαίσιο θα λειτουργεί σωστά.
- **Η γραμμή 16** ορίζει τη λεζάντα του προγράμματος Pygame χρησιμοποιώντας το pygame.display.set_caption(). Σε αυτήν την περίπτωση, η λεζάντα θα είναι το όνομα του παιχνιδιού μας: Asteroids.

- **Οι γραμμές 18 και 21** ορίζουν τις _handle_input() και _process_game_logic(). Είναι δύο συναρτήσεις τις οποίες και αφήνουμε άδειες προς το παρόν, αλλά στις επόμενες ενότητες θα προσθέσουμε κώδικα για να κάνουμε το παιχνίδι μας πιο ενδιαφέρον.
- **Η γραμμή 24** ορίζει την _draw(). Δεν θα είχε πολύ νόημα να δημιουργήσουμε ένα πρότυπο για το παιχνίδι μας χωρίς να εμφανίζεται τίποτα στην οθόνη, επομένως αυτή η μέθοδος έχει ήδη κάποιο κώδικα. Ονομάζεται κάθε καρέ για να σχεδιάσει το περιεχόμενο της οθόνης και αυτό το κάνει σε δύο βήματα:
  - **Η γραμμή 25** γεμίζει την οθόνη με ένα χρώμα χρησιμοποιώντας τη screen.fill(). Η μέθοδος παίρνει μια πλειάδα με τρεις τιμές, που αντιπροσωπεύουν τρία βασικά χρώματα: κόκκινο, πράσινο και μπλε. Κάθε τιμή χρώματος κυμαίνεται μεταξύ 0 και 255, αντιπροσωπεύοντας την έντασή του. Σε αυτό το παράδειγμα, μια πλειάδα των (0, 0, 255) σημαίνει ότι το χρώμα θα αποτελείται μόνο από μπλε, χωρίς ίχνη κόκκινου ή πράσινου.
  - **Η γραμμή 26** ενημερώνει το περιεχόμενο της οθόνης χρησιμοποιώντας το pygame.display.flip(). Επειδή το παιχνίδι θα εμφανίζει κινούμενα αντικείμενα, θα καλούμε αυτήν τη μέθοδο σε κάθε καρέ για να ενημερώνουμε την οθόνη. Εξαιτίας αυτού, πρέπει να γεμίζουμε την οθόνη μας με χρώμα σε κάθε καρέ, καθώς η μέθοδος θα καθαρίζει τα περιεχόμενα που δημιουργήθηκαν στο προηγούμενο καρέ.

Μέχρι στιγμής, ο κώδικάς μας φαίνεται να έχει πολλά επιπλέον βήματα, αλλά τώρα ο κώδικάς είναι όμορφα δομημένος και έχει μεθόδους με περιγραφικά ονόματα. Την επόμενη φορά που θα χρειαστεί να αλλάξουμε κάτι που σχετίζεται με το σχεδιασμό, θα ξέρουμε ότι θα πρέπει να παραμετροποιήσουμε την _draw(). Για να προσθέσουμε

χειρισμό εισόδου, θα τροποποιήσουμε την _handle_input() και ούτω καθεξής.

Σχόλιο: Κανονικά, θα θέταμε μεταβλητές όπως το μέγεθος και το χρώμα της οθόνης σε μια σταθερά στην αρχή της κλάσης. Ωστόσο, σε λίγα βήματα, θα αντικαταστήσουμε το χρώμα με μια εικόνα και δεν θα χρησιμοποιήσουμε το μέγεθος της οθόνης εκτός απ' αυτή τη μέθοδο. Έτσι αφήνουμε τις τιμές ως έχουν.

Στη συνέχεια, δημιουργούμε ένα __main__.py αρχείο. Αυτό το αρχείο θα φροντίσει να δημιουργήσει ένα νέο στιγμιότυπο του παιχνιδιού και να το ξεκινήσει εκτελώντας το main_loop().

Ο κώδικας είναι:

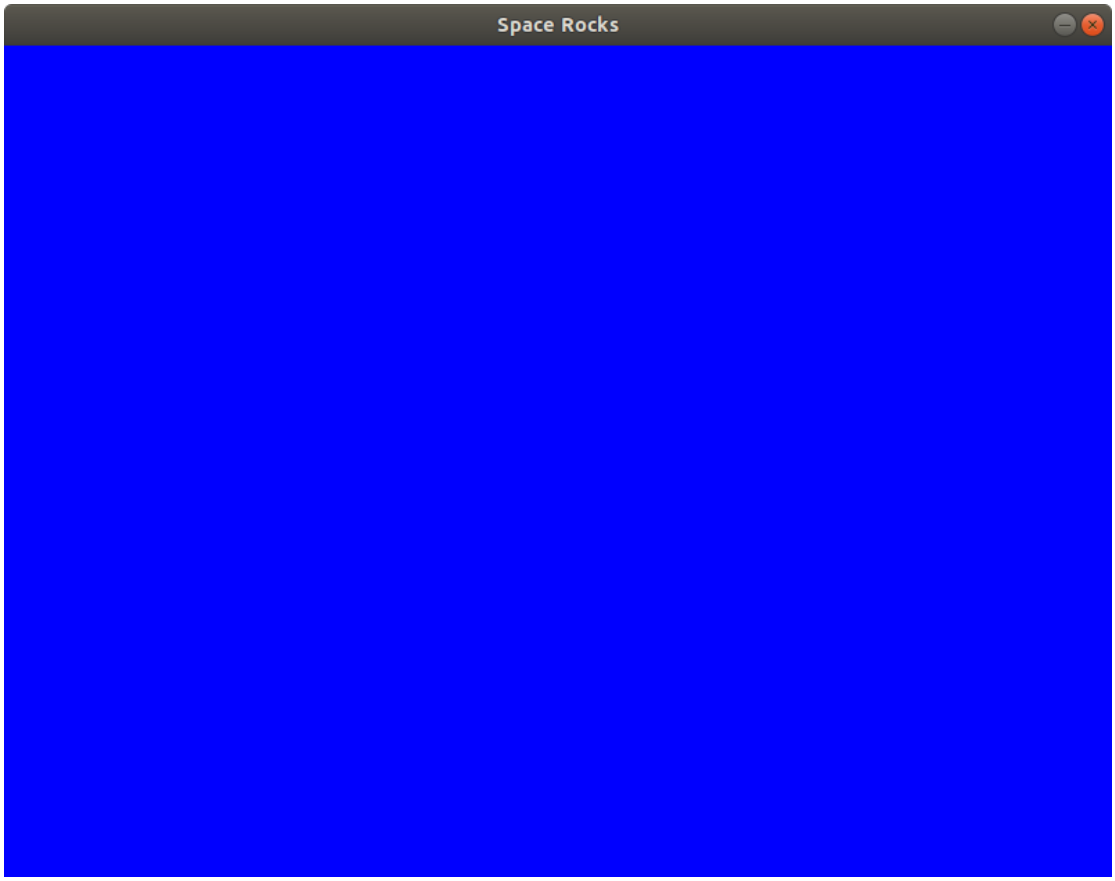
```
from game import SpaceRocks


if __name__ == "__main__":
    space_rocks = SpaceRocks()
    space_rocks.main_loop()
```

Έχουμε τώρα 2 αρχεία, τα game.py και __main__.py τα οποία μπορούμε να βάλουμε στο φάκελο desktop ή σε κάποιον άλλο της προτίμησής μας, αφού τα βάλουμε και σε ένα φάκελο που θα δημιουργήσουμε με το όνομα scode1.

Κατόπιν, μπορούμε να τρέξουμε τον κώδικά μας από το command line των Windows (αφού μεταβούμε στο παραπάνω directory), δίνοντας την εντολή:

python space_rocks

κι έτσι θα πάρουμε το παράθυρο:



Φυσικά, μέχρι στιγμής, έχουμε μόνο ένα κενό παράθυρο, όμως αυτό έχουμε γράψει στον κώδικά μας. Επίσης, δεν έχουμε τρόπο να εξέλθουμε τερματίζοντας το πρόγραμμα, οπότε θα πρέπει να χρησιμοποιήσουμε το Alt+C στο command line μας.

21.1.5 Χειρισμός εισόδου

Σε αυτό το σημείο, έχουμε τον κύριο βρόχο του παιχνιδιού, έτοιμο να τρέξει τη λογική του παιχνιδιού.

Στο τέλος αυτού του βήματος, θα έχουμε επίσης μια θέση για να αρχίσουμε να συνδέουμε τα στοιχεία ελέγχου του χρήστη.

Η μεγαλύτερη επεξεργασία εισόδου στο Pygame γίνεται μέσα σε έναν βρόχο συμβάντων . Σε κάθε καρέ, το πρόγραμμά μας μπορεί να λάβει μια συλλογή γεγονότων που συνέβησαν από το προηγούμενο καρέ.

Αυτό περιλαμβάνει την κίνηση του ποντικιού, τα πιθανά πατήματα πλήκτρων και ούτω καθεξής.

Στη συνέχεια, ένα προς ένα, αυτά τα γεγονότα μπορούν να αντιμετωπιστούν. Στο Pygame, η μέθοδος απόκτησης αυτής της συλλογής είναι η pygame.event.get().

Η δήλωση που χρειαζόμαστε αυτή τη στιγμή είναι pygame.QUIT. Έτσι μπορούμε να σταματήσουμε το παιχνίδι, είτε κάνοντας κλικ στο Κλείσιμο είτε πατώντας Alt+F4 σε Windows και Linux ή Cmd+W στο macOS. Ας τροποποιήσουμε τώρα το αρχείο game.py ξαναγράφοντας την SpaceRocks._handle_input() ως εξής:

```
def _handle_input(self):
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            quit()
```

Μπορούμε ήδη να το δοκιμάσουμε κι εδώ είναι ευκαιρία να πούμε πως μπορούμε να συσχετίζουμε πλήκτρα του πληκτρολογίου με λειτουργίες του παιχνιδιού, διαβάζοντας την τεκμηρίωση του pygame.

Στο παιχνίδι μας τώρα, για να κλείσουμε το παιχνίδι πατώντας Esc, θα χρησιμοποιήσουμε το pygame.K_ESCAPE.

Ας τροποποιήσουμε ξανά τη μέθοδο _handle_input():

```
def _handle_input(self):
    for event in pygame.event.get():
        if event.type == pygame.QUIT or (
            event.type == pygame.KEYDOWN and event.key ==
pygame.K_ESCAPE
        ):
            quit()
```

Τώρα το παιχνίδι μας κλείνει επίσης όταν πατάμε Esc.

Καταφέραμε να εμφανίσουμε ένα παράθυρο και να το κλείσουμε σωστά. Αλλά το παράθυρο είναι ακόμα γεμάτο με ένα μόνο χρώμα. Στη συνέχεια, θα μάθουμε πώς να φορτώνουμε μια εικόνα και να την εμφανίζουμε στην οθόνη.

21.1.6 Εισαγωγή Εικόνων

Μέχρι εδώ, έχουμε ένα παράθυρο του παιχνιδιού που μπορούμε να το κλείσουμε πατώντας ένα πλήκτρο. Στο τέλος αυτού του βήματος, θα εμφανίσουμε μια εικόνα σε αυτό το παράθυρο.

Αν και θα μπορούσαμε να φτιάξουμε ένα παιχνίδι στον υπολογιστή μόνο με χρωματιστά ορθογώνια και άλλα απλά σχήματα, η χρήση εικόνων θα το κάνει πολύ πιο ελκυστικό. Στην ανάπτυξη παιχνιδιών υπολογιστή, οι εικόνες ονομάζονται συνήθως sprites. Φυσικά, τα παιχνίδια χρησιμοποιούν πολύ περισσότερους τύπους πόρων, όπως ήχους, γραμματοσειρές, κινούμενα σχέδια και ούτω καθεξής. Οι πόροι αυτοί, συνήθως ονομάζονται assets.

Καθώς το παιχνίδι μας μεγαλώνει, είναι σημαντικό να διατηρεί τη σωστή δομή. Δημιουργούμε λοιπόν έναν φάκελο με το όνομα assets και, μέσα σε αυτόν, έναν άλλο με το όνομα sprites. Εκεί θα βάλουμε όλα τα sprites που θα χρησιμοποιεί το παιχνίδι μας.

Στη συνέχεια, ας κατεβάσουμε την εικόνα του φόντου του διαστήματος και ας την τοποθετήσουμε στο φάκελο assets/sprites.

Επίσης, επειδή οι εικόνες θα φορτωθούν πολλές φορές στο πρόγραμμά μας, είναι καλή ιδέα να εξαγάγουμε αυτήν τη λειτουργία σε ξεχωριστή μέθοδο σε ξεχωριστό αρχείο. Ας δημιουργήσουμε ένα αρχείο με το όνομα space_rocks/utils.py που θα διατηρεί όλες τις

επαναχρησιμοποιήσιμες μεθόδους. Στη συνέχεια, εφαρμόζουμε τη φόρτωση εικόνας:

```
from pygame.image import load


def load_sprite(name, with_alpha=True):
    path = f"assets/sprites/{name}.png"
    loaded_sprite = load(path)

    if with_alpha:
        return loaded_sprite.convert_alpha()
    else:
        return loaded_sprite.convert()
```

Ας σχολιάσουμε τον παραπάνω κώδικα:

Η γραμμή 1 εισάγει μια μέθοδο που ονομάζεται load()η οποία θα είναι απαραίτητη για την ανάγνωση εικόνων αργότερα.

Η γραμμή 4 δημιουργεί μια διαδρομή προς μια εικόνα, υποθέτοντας ότι είναι αποθηκευμένη στο φάκελο assets/sprites και ότι είναι αρχείο PNG. Με αυτόν τον τρόπο, θα χρειαστεί να δώσουμε μόνο το όνομα του sprite αργότερα.

Η γραμμή 5 φορτώνει την εικόνα χρησιμοποιώντας τη load(). Αυτή η μέθοδος επιστρέφει μια επιφάνεια, η οποία είναι ένα αντικείμενο που χρησιμοποιείται από την Pygame για την αναπαράσταση εικόνων. Μπορούμε αργότερα να την εισάγουμε στην οθόνη (ή σε άλλη επιφάνεια, αν θέλουμε).

Οι γραμμές 8 και 10 μετατρέπουν την εικόνα σε μορφή που ταιριάζει καλύτερα στην οθόνη για να επιταχύνει τη διαδικασία σχεδίασης. Αυτό γίνεται με τις μεθόδους convert_alpha() ή convert(), ανάλογα με το αν θέλουμε να χρησιμοποιήσουμε τη διαφάνεια.

Σημείωση: Γενικά, θα μπορούσαμε απλώς να χρησιμοποιήσουμε την convert_alpha() για όλους τους τύπους εικόνων, καθώς αυτή η μέθοδος, μπορεί επίσης να χειριστεί μια εικόνα χωρίς διαφανή pixel. Ωστόσο, η σχεδίαση διαφανών εικόνων είναι λίγο πιο αργή από τη σχεδίαση αδιαφανών εικόνων.

Δεδομένου ότι τα παιχνίδια στον υπολογιστή έχουν να κάνουν με την απόδοση (ταχύτητα), θα εξασκηθούμε στη βελτιστοποίηση των παιχνιδιών μας επιλέγοντας τον σωστό τύπο εικόνας και αυξάνοντας την ταχύτητα του παιχνιδιού, έστω και λίγο.

Τώρα που το πρόγραμμά μας μπορεί να φορτώσει εικόνες, ήρθε η ώρα να αλλάξουμε το μπλε φόντο σε κάτι πιο ενδιαφέρον.

Επεξεργαζόμαστε το αρχείο game.py κι έχουμε:

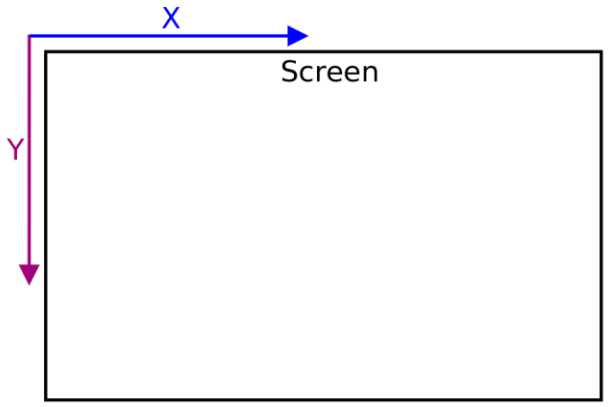
```
class SpaceRocks:
    def __init__(self):
        self._init_pygame()
        self.screen = pygame.display.set_mode((800, 600))
        self.background = load_sprite("space", False)

    def main_loop(self):
        while True:
            self._handle_input()
            self._process_game_logic()
            self._draw()
```

```
    def _init_pygame(self):
        pygame.init()
        pygame.display.set_caption("Space Rocks")

    def _handle_input(self):
        for event in pygame.event.get():
            if event.type == pygame.QUIT or (
                event.type == pygame.KEYDOWN and event.key ==
pygame.K_ESCAPE
            ):
                quit()

    def _process_game_logic(self):
        pass

    def _draw(self):
        self.screen.blit(self.background, (0, 0))
        pygame.display.flip()
```

Για να εμφανίσουμε μια επιφάνεια σε μια άλλη στο Pygame, πρέπει να καλέσουμε την επιφάνεια που θέλουμε να σχεδιάσουμε με τη μέθοδο blit(). Αυτή η μέθοδος παίρνει δύο ορίσματα:

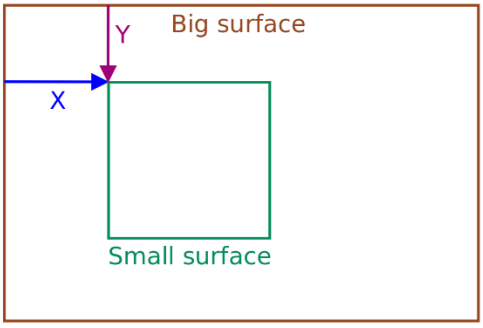
- Την επιφάνεια που θέλουμε να σχεδιάσουμε
- και το σημείο που θέλουμε να τη σχεδιάσουμε

Είναι καλό να έχουμε κατά νου ότι στο Pygame, το σύστημα συντεταγμένων ξεκινά από την επάνω αριστερή γωνία. Ο άξονας x πηγαίνει από αριστερά προς τα δεξιά και ο άξονας y πηγαίνει από πάνω προς τα κάτω:



Όπως μπορούμε να δούμε, το UP διάνυσμα, που δείχνει προς τα πάνω, θα έχει αρνητική συντεταγμένη y.

Οι συντεταγμένες που δίνονται με την blit() αντιπροσωπεύουν το σημείο όπου θα βρίσκεται η επάνω αριστερή γωνία της επιφάνειας μετά τη δήλωση:

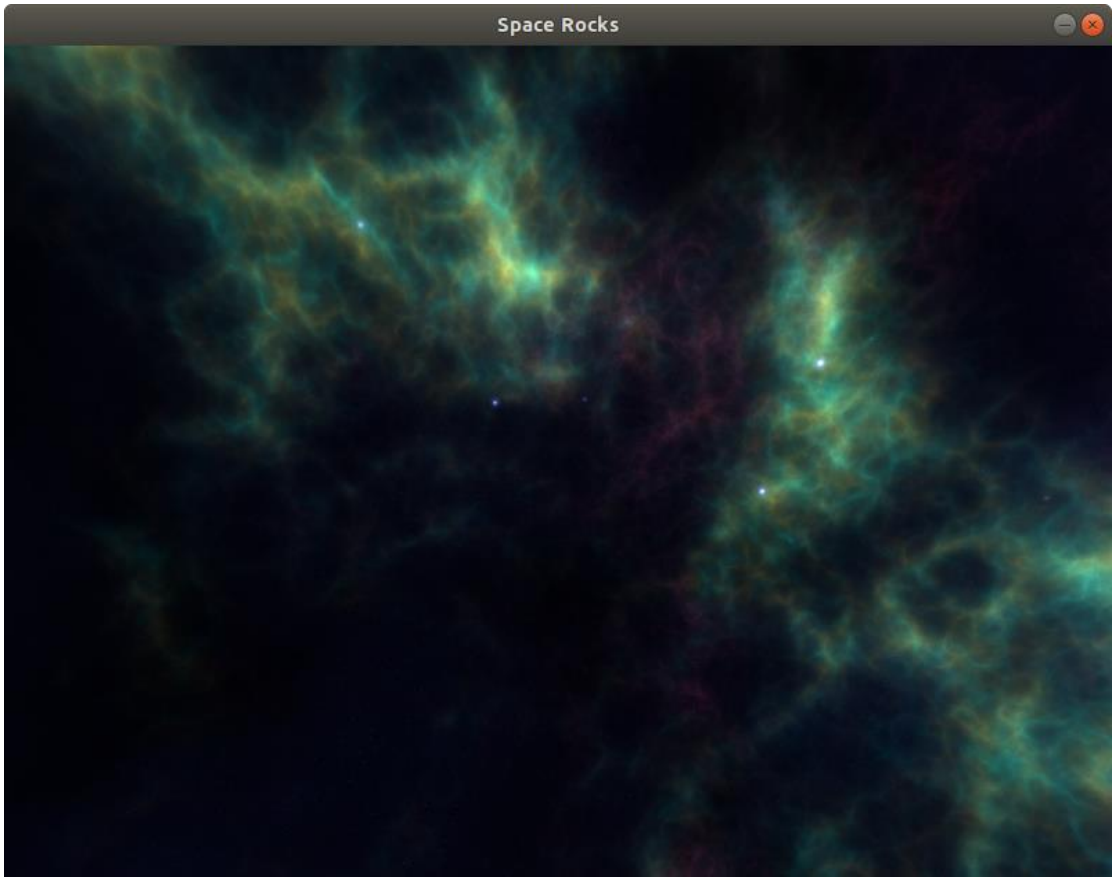


Εξήγηση του blitting

Όπως μπορούμε να δούμε, η επάνω αριστερή γωνία μετακινείται από τις συντεταγμένες του blit για να υπολογιστεί η σωστή θέση.

Στην περίπτωσή μας, το νέο φόντο έχει το ίδιο μέγεθος με την οθόνη (800× 600pixel), επομένως οι συντεταγμένες θα είναι (0, 0), που αντιπροσωπεύουν την επάνω αριστερή γωνία της οθόνης. Με αυτόν τον τρόπο, η εικόνα φόντου θα καλύπτει ολόκληρη την οθόνη.

Τώρα, εκτελώντας το πρόγραμμα, βλέπουμε την νέα οθόνη του φόντου:



Το παιχνίδι μας έχει τώρα μια πολύ ωραία εικόνα φόντου, αλλά τίποτα δεν συμβαίνει ακόμα εκεί.

Ας το αλλάξουμε αυτό προσθέτοντας κάποια αντικείμενα.

21.1.7 Έλεγχος των αντικειμένων του παιχνιδιού

Σε αυτό το σημείο, το πρόγραμμά μας εμφανίζει μια εικόνα φόντου ενός μικρού κομματιού του σύμπαντος όπου θα λάβει χώρα το παιχνίδι μας με τους Αστεροειδείς.

Είναι λίγο άδειο αυτήν τη στιγμή, οπότε σε αυτήν την ενότητα θα το γεμίσουμε.

Θα δημιουργήσουμε μια κλάση που αντιπροσωπεύει άλλα αντικείμενα του παιχνιδιού με δυνατότητα σχεδίασης και θα τη χρησιμοποιήσουμε για να φτιάξουμε και να χρησμιμοποιήσυμε ένα διαστημόπλοιο και έναν αστεροειδή.

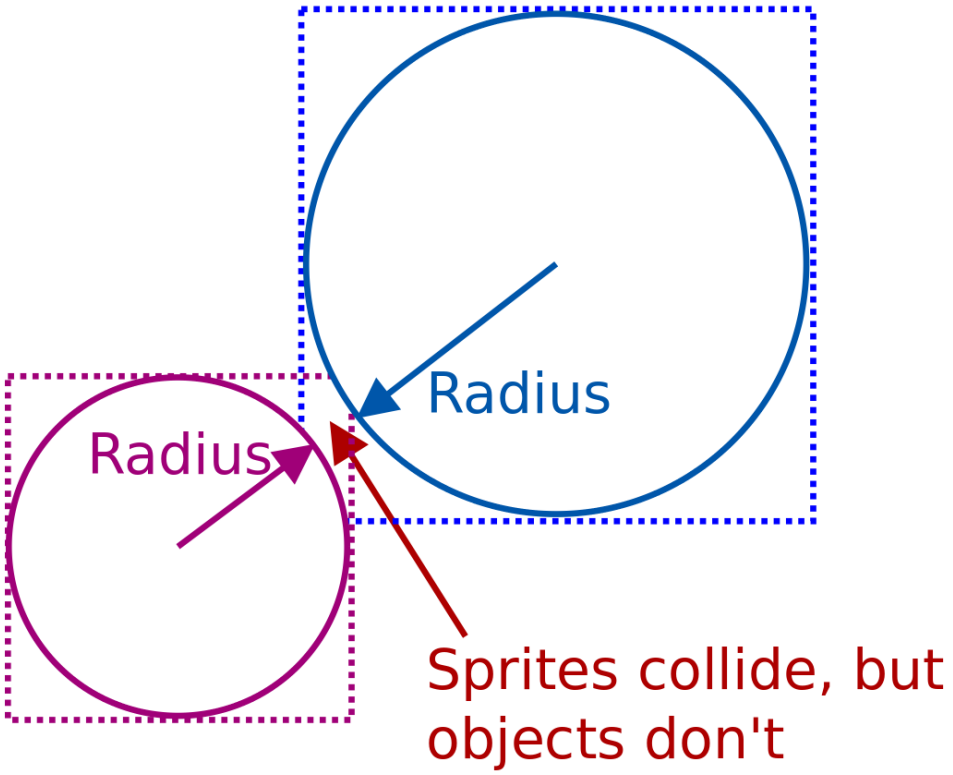
Προηγμένη Συμπεριφορά

Έχουμε ήδη χρησιμοποιήσει επιφάνειες, αλλά το Pygame προσφέρει επίσης μια άλλη κατηγορία Sprite, η οποία προορίζεται ως βασική κατηγορία για ορατά αντικείμενα. Περιέχει μερικές χρήσιμες μεθόδους, αλλά επίσης θα έχουμε και κάποιους περιορισμούς.

Ένας περιορισμός είναι ότι ένα αντικείμενο του παιχνιδιού είναι κάτι περισσότερο από ένα απλό sprite. Περιέχει πρόσθετα δεδομένα, όπως η κατεύθυνση και η ταχύτητά του. Χρειάζεται επίσης πιο προηγμένες συμπεριφορές, όπως να πυροβολεί σφαίρες ή να παίζει ήχους. Οι περισσότερες από αυτές τις πρόσθετες πληροφορίες και συμπεριφορές δεν παρέχονται από την κλάση Sprite, επομένως θα πρέπει να τις προσθέσουμε μόνοι μας.

Ένα άλλο ζήτημα είναι ότι το Pygame σχεδιάζει sprites ξεκινώντας από την επάνω αριστερή γωνία. Στο παιχνίδι μας, μπορεί να είναι ευκολότερο να αποθηκεύσουμε την κεντρική θέση ενός αντικειμένου με σκοπό τη μετακίνηση και την περιστροφή του. Σε αυτήν την περίπτωση, θα πρέπει να εφαρμόσουμε έναν τρόπο για να μετατρέψουμε αυτήν τη θέση σε μια επάνω αριστερή γωνία όπως απαιτείται από το Pygame.

Τέλος, αν και το Pygame έχει ήδη μεθόδους για τον εντοπισμό επικαλύψεων μεταξύ εικόνων, μπορεί να μην είναι καλές για τον εντοπισμό συγκρούσεων μεταξύ αντικειμένων. Ένα περιστρεφόμενο διαστημόπλοιο ή ένας αστεροειδής πιθανότατα δεν θα γεμίσει ολόκληρη την εικόνα, αλλά μάλλον τη στρογγυλή περιοχή μέσα σε αυτήν. Σε αυτήν την περίπτωση, η σύγκρουση θα πρέπει να λαμβάνει υπόψη μόνο αυτή τη στρογγυλή περιοχή, όχι ολόκληρη την επιφάνεια του sprite. Διαφορετικά, ενδέχεται να λάβουμε λανθασμένα αποτελέσματα:



Σε αυτό το παράδειγμα, τα sprites συγκρούονται, αλλά τα αντικείμενα του παιχνιδιού όχι.

Αυτό είναι πραγματικά το σημείο όπου η κλάση Sprite θα μπορούσε να βοηθήσει, αφού μπορούμε να τη χρησιμοποιήσουμε με τη μέθοδο pygame.sprite.collide_circle().

Αυτή η μέθοδος ανιχνεύει συγκρούσεις μεταξύ δύο sprites χρησιμοποιώντας κύκλους με κέντρο τις επιφάνειές τους.

Ευτυχώς, ο εντοπισμός μιας σύγκρουσης κύκλων δεν είναι μια πολύ περίπλοκη διαδικασία και μπορούμε να την εφαρμόσουμε μόνοι μας.

Δεδομένων αυτών των θεμάτων, γίνεται γρήγορα προφανές ότι η ενσωματωμένη κλάση Sprite του Pygame πρόκειται να επαυξηθεί κι όχι απλώς να χρησιμοποιηθεί από μόνη της.

Στην περίπτωση του παιχνιδιού μας, τα Pygame sprites παρέχουν λίγες χρήσιμες λειτουργίες. Ίσως θα ήταν καλή ιδέα να εφαρμόσουμε μια προσαρμοσμένη κλάση για τα αντικείμενα παιχνιδιού. Αυτό θα μας δώσει περισσότερο έλεγχο και θα μας βοηθήσει να κατανοήσουμε ορισμένες έννοιες αφού θα τις εφαρμόσουμε μόνοι μας.

21.1.8 Κλάση GameObject

Σε αυτήν την ενότητα, θα παρουσιάσουμε την κλάση GameObject.

Θα ενσωματώσει κάποια γενική συμπεριφορά και δεδομένα για όλα τα άλλα αντικείμενα του παιχνιδιού. Οι κλάσεις που αντιπροσωπεύουν συγκεκριμένα αντικείμενα (όπως το διαστημόπλοιο) θα κληρονομήσουν από αυτήν και θα την επεκτείνουν με τη δική τους συμπεριφορά και χαρακτηριστικά.

Η κλάση GameObject θα αποθηκεύσει τα ακόλουθα δεδομένα:

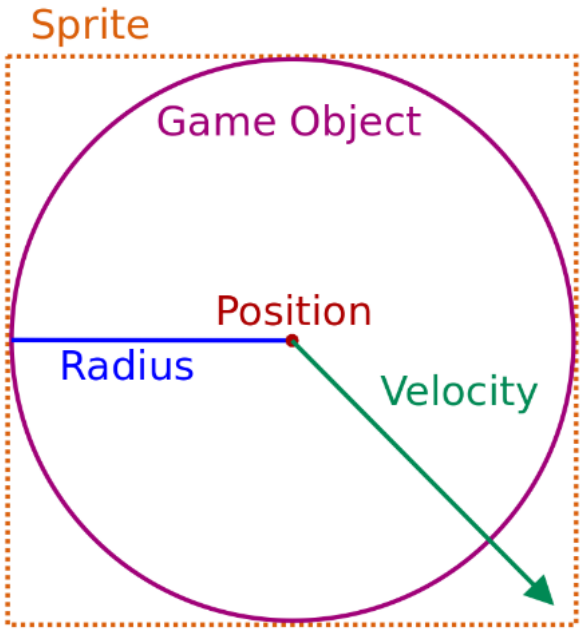
position: Ένα σημείο στο κέντρο του αντικειμένου στην οθόνη 2D

sprite: Μια εικόνα που χρησιμοποιείται για την εμφάνιση του αντικειμένου

radius: Μια τιμή που αντιπροσωπεύει τη ζώνη σύγκρουσης γύρω από τη θέση του αντικειμένου

velocity: Μια τιμή που χρησιμοποιείται για κίνηση

Ακολουθεί μια γραφική αναπαράσταση ενός αντικειμένου του παιχνιδιού:



Εξήγηση της GameObject

Το sprite είναι μια επιφάνεια που φορτώθηκε με την load_sprite() ήδη. Το radius είναι ένας ακέραιος αριθμός που υποδεικνύει τον αριθμό των pixel από το κέντρο του αντικειμένου μέχρι την άκρη της ζώνης σύγκρουσης. Ωστόσο, το position όπως και το velocity θα χρειάζονται έναν νέο τύπο αναπαράστασης: ένα **διάνυσμα** .

Τα διανύσματα είναι παρόμοια με τα tuples. Σε έναν κόσμο 2D (όπως αυτός στο παιχνίδι μας), τα διανύσματα αντιπροσωπεύονται από δύο τιμές που υποδεικνύουν συντεταγμένες x και y.

Αυτές οι συντεταγμένες μπορούν να δείχνουν μια θέση, αλλά μπορούν επίσης να αντιπροσωπεύουν κίνηση ή επιτάχυνση προς μια δεδομένη κατεύθυνση. Τα διανύσματα μπορούν να προστεθούν, να αφαιρεθούν ή ακόμα και να πολλαπλασιαστούν για γρήγορη ενημέρωση της θέσης ενός sprite.

ΤΕΛΟΣ Α' ΜΕΡΟΥΣ

ΤΟ ΚΑΛΟΚΑΙΡΑΚΙ ΣΥΝΕΧΙΖΕΤΑΙ!

ΓΙΑ ΛΙΓΟ ΑΚΟΜΑ (ΔΥΣΤΥΧΩΣ!)