UNIVERSITATEA „POLITEHNICA” din BUCUREŞTI
Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei

SpaceDodge

Proiect Python

Nițu Rareș-Mihai

Iordache Mihai

Grupa 421A

2023

Cuprins

Descriere

Proiectul a avut ca scop dezvoltarea unuia dintre primele jocuri, numit SpaceDodge. Acesta reprezintă o experiență captivantă de acțiune spațială, punând la încercare reflexele și abilitățile de evitare într-un mediu plin de pericole intergalactice. Realizarea acestui joc a fost facilitată de utilizarea bibliotecii pygame, ceea ce a permis o implementare simplă, dar captivantă.

Instrucțiuni de joc

1. Obiectivul Jocului:
   - Obiectivul principal al jocului SpaceDodge este ca jucătorul să supraviețuiască cât mai mult timp posibil în spațiul cosmic.
   - Pe măsură ce timpul trece, nivelul de dificultate crește, iar obstacolele devin din ce în ce mai complexe.
2. Controlul Navei Spațiale:
   - Tasta săgeată stânga: Deplasează nava spațială la stânga.
   - Tasta săgeată dreapta: Deplasează nava spațială la dreapta.
   - Tasta săgeată sus: Deplasează nava spațială în sus.
   - Tasta săgeată jos: Deplasează nava spațială în jos.
3. Evitarea Obstacolelor:
   - Obstacolele, reprezentate sub forma de stele, vor apărea regulat pe ecran.
   - Numărul de stele crește o data cu timpul de joc.
4. Pierderea Jocului:
   - Dacă nava spațială intră în coliziune cu un obstacol, jocul se va încheia.
   - Un mesaj "You Lost!" va apărea pe ecran, iar jocul se va opri.

Instrucțiuni de instalare

Pentru a rula acest cod în Visual Studio Code:

1. Instalează Python:
   a) Dacă nu ai Python deja instalat, descarcă și instalează versiunea cea mai recentă de Python de pe site-ul oficial Python.
   b) În timpul instalării, asigură-te că bifezi opțiunea "Add Python to PATH".
2. Instalează Visual Studio Code:
   a) Descarcă și instalează Visual Studio Code.
3. Deschide Folderul cu Codul:
   a) Creează un nou folder pentru proiectul tău SpaceDodge.
   b) Deschide acest folder în Visual Studio Code.
4. Instalează Extensia Python:
   a) În Visual Studio Code, mergi în meniul de extensii și caută "Python".
   b) Instalează extensia "Python" oferită de Microsoft.
5. Configurare Interpret Python:
   a) După instalare, deschide fișierul main.py.

b) Deasupra editorului, în partea dreaptă, ar trebui să vezi o notificare că un interpret Python nu a fost găsit. Dă click pe acea notificare.
c) Alege un interpret Python instalat pe sistemul tău.

6. Instalează Biblioteca Pygame:
   a) Deschide terminalul în Visual Studio Code (View -> Terminal).
   b) În terminal, scrie **pip install pygame** și apasă Enter pentru a instala biblioteca Pygame necesară pentru acest joc.
7. Rulează Codul:
   a) După instalarea Pygame, poți să rulezi codul apăsând F5 sau apăsând pe butonul "Run Python File in Terminal" din partea de sus a editorului.

Pentru a rula codul în IDLE 3.0, urmează aceste instrucțiuni:

1. Instalează Python 3:
   a. Dacă nu ai Python 3.0 sau o versiune mai recentă instalată, descarcă și instalează Python de pe site-ul oficial Python.
   b. În timpul instalării, asigură-te că bifezi opțiunea "Add Python to PATH".
2. Deschide IDLE 3.0:
   a. După instalare, deschide IDLE 3.0. Poți face acest lucru căutând "IDLE" în meniul de start sau prin intermediul comenzii în terminal (idle).
3. Deschide sau Creează Fișierul cu Cod:
   a. Deschide sau creează un nou fișier în IDLE și copiază întregul cod SpaceDodge în acest fișier.
4. Salvează Fișierul:
   a. Salvează fișierul cu o extensie .py. De exemplu, poți să-l salvezi ca spacedodge.py.
5. Instalează Biblioteca Pygame:
   a. Deschide un terminal în IDLE (IDLE -> Run -> Python Shell).
   b. În terminal, scrie pip install pygame și apasă Enter pentru a instala biblioteca Pygame necesară pentru acest joc.
6. Rulează Codul:
   a. În IDLE, alege "Run Module" din meniul "Run" sau apasă F5 pentru a rula codul.

Descrierea codului

La inceput importăm modulele pe care urmează să le folosim:

```
import pygame
import time
import random
import pygame.mask
import sys
```

- **Pygame** – Modulul Python este o bibliotecă dedicată pentru dezvoltarea de jocuri în Python. Este esențial în acest cod pentru crearea și gestionarea interfeței de utilizator a jocului, desenarea pe ecran, gestionarea evenimentelor și a sunetului.

- **Time** - Modulul time oferă funcționalități legate de gestionarea timpului. Este utilizat aici pentru a măsura timpul scurs de la începutul jocului și pentru a crea întârzieri pentru a regla viteza de generare a obstacolelor.
- **Random** - Modulul random este utilizat pentru a genera numere aleatoare. În acest cod, este folosit pentru a alege poziții aleatoare pentru obstacole (stele) în cadrul jocului.
- **Pygame.mask** - Acest modul face parte din biblioteca Pygame și furnizează funcționalități pentru gestionarea măștilor utilizate în coliziuni. În acest caz, este utilizat pentru a verifica coliziunile între nava spațială și obstacole (stele).
- **Sys** - Modulul sys oferă acces la funcționalități specifice sistemului. În acest cod, este folosit pentru a încheia corect jocul prin oprirea execuției programului.

Apelăm modulele Pygame referitoare la fonturi si sunet:

```
pygame.font.init()
pygame.mixer.init()
```

- **Pygame.font.init()** - Această instrucțiune inițializează modulul Pygame responsabil pentru manipularea fonturilor
- **Pygame.mixer.init()** - Această instrucțiune inițializează modulul Pygame pentru manipularea sunetului (mixer)

Definim dimensiunile ferestrei:

```
WIDTH = 1000
HEIGHT = 800
WIN = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Space Dodge")
```

Incarcăm o variabilă cu imaginea de fundal:

```
BG = pygame.transform.scale(pygame.image.load("bg.jpg"), (WIDTH, HEIGHT))
```

Definim dimensiunile navei spațiale si creăm o maskă pentru aceasta:

```
SPACESHIP_WIDTH = 80
SPACESHIP_HEIGHT = 120
SPACESHIP = pygame.transform.scale(pygame.image.load("spaceship.png"),
(SPACESHIP_WIDTH, SPACESHIP_HEIGHT))
#definim o maska pentru a avea dimensiunile cat mai exacte ale navei spatiale
SPACESHIP_MASK = pygame.mask.from_surface(SPACESHIP)
```

Definim dimensiunile si viteza obstacolelor:

```
STAR_WIDTH, STAR_HEIGHT = 10, 20
STAR_VEL = 3
```

Incărcăm fișierul cu muzică de fundal:

```
pygame.mixer.music.load("background_music.mp3")
pygame.mixer.music.set_volume(0.5)  # Adjust the volume (0.0 to 1.0)
pygame.mixer.music.play(-1)  # -1 means loop indefinitely
```

Realizăm o funcție de desenare a elementelor pe ecran:

```
def draw(spaceship, elapsed_time, stars):
    WIN.blit(BG, (0, 0))

    #afisam timpul de cand a inceput jocul pe ecran in coltul din stanga sus
    time_text = FONT.render(f"Time: {round(elapsed_time)}s", 1, "white")
    WIN.blit(time_text, (10, 10))

    #afisarea pe ecran a navei spatiale
    WIN.blit(SPACESHIP, (spaceship.x, spaceship.y))

    #afisarea pe ecran a obstacolelor
    for star in stars:
        pygame.draw.rect(WIN, "white", star)

    pygame.display.update()
```

- Desenarea Fundalului și Timpului:
  - **WIN.blit(BG, (0, 0))** - Această linie desenează imaginea de fundal (BG) pe întregul ecran al jocului la coordonatele (0, 0). Astfel, imaginea de fundal este afișată în fiecare cadru.
  - **time_text = FONT.render(f"Time: {round(elapsed_time)}s", 1, "white")** - Se creează un obiect de tip text cu timpul scurs de la începutul jocului și se alege fontul, dimensiunea și culoarea textului.
  - **WIN.blit(time_text, (10, 10))** - Acestă linie plasează textul timpului în colțul din stânga sus al ecranului la coordonatele (10, 10).
- Desenarea Navei Spațiale:
  - **WIN.blit(SPACESHIP, (spaceship.x, spaceship.y))** - Această linie plasează imaginea navei spațiale pe ecran la coordonatele (spaceship.x, spaceship.y). Astfel, poziția și aspectul navei spațiale sunt actualizate în fiecare cadru în funcție de mișcările jucătorului.
- Desenarea Obstacolelor (Stelelor):
  - **for star in stars: pygame.draw.rect(WIN, "white", star) -** Această buclă parcurge lista de obiecte de tip stea (obstacole) și desenează un dreptunghi alb pe ecran pentru fiecare dintre ele. Aceasta este o metodă simplă de a reprezenta obstacolele, iar poziția și dimensiunile acestora sunt actualizate în fiecare cadru.
- Actualizarea Ecranului:
  - **pygame.display.update()** - Această linie actualizează ecranul jocului pentru a reflecta schimbările efectuate în cadrul funcției draw. Fără această actualizare, schimbările nu ar fi vizibile pe ecran.

Funcția principală a programului:

```
def main():
    run = True

    #initializam variabile pentru nava si timp
```

```
    spaceship = pygame.Rect(200, HEIGHT - SPACESHIP_HEIGHT, SPACESHIP_WIDTH,
SPACESHIP_HEIGHT)
    clock = pygame.time.Clock()
    start_time = time.time()
    elapsed_time = 0

    #initializam variabilele pentru generarea obstacolelor
    star_add_increment = 2000
    star_count = 0

    #creeam o lista pentru a salva obstacolele
    stars = []

    hit = False

    while run:
        #actualizam contorul pentru a adauga stele si pentru a afla timpul scurs
        star_count += clock.tick(60)
        elapsed_time = time.time() - start_time

        #generam obstacole la intervale regulate
        if star_count > star_add_increment:
            for _ in range(3):
                star_x = random.randint(0, WIDTH - STAR_WIDTH)
                star = pygame.Rect(star_x, -STAR_HEIGHT, STAR_WIDTH, STAR_HEIGHT)
                stars.append(star)

            #reducem intervalul de generare al stelelor pentru a creste dificultatea
            star_add_increment = max(200, star_add_increment - 50)
            star_count = 0

        #verificam daca fereastra a fost inchisa
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                run = False
                break

        #realizarea miscarii navetei spatiale prin intermediul tastelor apasate
        keys = pygame.key.get_pressed()
        if keys[pygame.K_LEFT] and spaceship.x - SPACESHIP_WIDTH + 80>= 0:
            spaceship.x -= SPACESHIP_WIDTH * 0.125
        if keys[pygame.K_RIGHT] and spaceship.x + SPACESHIP_WIDTH <= WIDTH:
            spaceship.x += SPACESHIP_WIDTH * 0.125
        if keys[pygame.K_UP] and spaceship.y - SPACESHIP_HEIGHT + 60 >= 0:
            spaceship.y -= SPACESHIP_HEIGHT * 0.1
        if keys[pygame.K_DOWN] and spaceship.y + SPACESHIP_HEIGHT + 10 <= HEIGHT:
            spaceship.y += SPACESHIP_HEIGHT * 0.1


#realizam miscarea obstacolelor
        for star in stars[:]:
            star.y += STAR_VEL
            if star.y > HEIGHT:
                stars.remove(star)
            else:
```

```
                #pentru fiecare obstacol de pe ecran se creeaza o maska pentru a
putea verifica coliziunea
                star_rect = pygame.Rect(star.x, star.y, STAR_WIDTH, STAR_HEIGHT)
                star_mask = pygame.mask.from_surface(pygame.Surface((STAR_WIDTH,
STAR_HEIGHT)))
                offset = (star_rect.x - spaceship.x, star_rect.y - spaceship.y)
                #verificam daca a avut loc coliziunea intre obstacol si nava spatiala
                if SPACESHIP_MASK.overlap(star_mask, offset):
                    stars.remove(star)
                    hit=True
                    break

            #daca naveta a fost atinsa de un obstacol oprim jocul
            if hit:
                lost_text = FONT.render("You Lost!", 1, "white")
                WIN.blit(lost_text, (WIDTH/2 - lost_text.get_width()/2, HEIGHT/2 -
lost_text.get_height()/2))
                pygame.display.update()
                pygame.mixer.music.stop()  # Stop the music when the game ends
                pygame.time.delay(1000)
                sys.exit()

        #desenam obiectele pe ecran
        draw(spaceship, elapsed_time, stars)

    pygame.quit()
```

- Inițializarea Variabilelor:
    - **Run = True** - O variabilă care controlează starea jocului. Dacă run este True, jocul continuă să ruleze.
    - **Spaceship** - Un obiect care reprezintă poziția și dimensiunile navei spațiale la începutul jocului.
    - **Clock** - Un obiect ce monitorizează timpul.
    - **Start_time** - Timpul în care a început jocul, măsurat în secunde.
    - **Elapsed_time** - Timpul scurs de la începutul jocului, actualizat la fiecare rulare.
    - **Star_add_increment** - Un contor care determină intervalul de timp la care se adaugă noi obstacole.
    - **Star_count** - Contor pentru a urmări numarul obstacole generate.
    - **Stars** - O listă care va stoca obiectele de tip obstacol (stele).
    - **Hit** - Un indicator care arata daca a avut loc o coliziune între navă și un obstacol.
- Bucla Principală:
    - **while run** - menține jocul în execuție, pana la coliziunea dintre rachetă si obstacole.
- Generarea Obstacolelor:
    - obstacolele (stelele) sunt generate la intervale regulate în funcție de star_add_increment. Se generează trei obstacole la fiecare interval, iar intervalul este redus treptat pentru a crește dificultatea.

- Mișcarea Navei Spațiale:

        - Gestionarea mișcării navei spațiale pe ecran în funcție de tastele apăsate de utilizator (**pygame.key.get_pressed()**). Mișcarea are loc în funcție de direcția tastelor săgeată.
- Mișcarea Obstacolelor și Detectarea Coliziunilor:
    - Obstacolele se deplasează în jos pe ecran, iar dacă trec dincolo de marginea de jos a ecranului, sunt eliminate din lista. Se verifică, de asemenea, coliziunile între nave și obstacole.
- Încheierea Jocului:
    - Dacă are loc o coliziune, se afișează textul "You Lost!" și se oprește jocul. Apoi, programul se încheie după o scurtă întârziere.
- Desenarea Obiectelor pe Ecran:
    - Se apelează funcția draw pentru a desena obiectele pe ecran și a actualiza vizualizarea jocului în fiecare cadru.
- Încheierea Programului:
    - După ce utilizatorul pierde jocul, se încheie programul utilizând **pygame.quit()** și **sys.exit()**.

Apelăm funcția principală:

```
if __name__ == "__main__":

    main()
```

- **if __name__ == "__main__":** - Această declarație condițională verifică dacă scriptul este executat direct (nu este importat ca modul). Dacă această condiție este adevărată, înseamnă că scriptul este programul principal care este executat.
- **main() -** Dacă scriptul este programul principal, se apelează funcția main(). Acesta este un obicei comun în Python pentru a organiza codul: plasarea logicii principale a scriptului într-o funcție numită main() și apoi apelarea acesteia doar atunci când scriptul este executat direct.

Modul de rulare al codului:

Codul explicat mai sus este rulat in felul urmator:

1. Se importă modulele (pygame, time, random, pygame.mask, și sys)
2. Se inițializează variabilele si dimensiunile acestora
3. Se definește funcția de desenare
4. Se creează o funcție main care o să ruleze până când jucătorul o să inchidă programul
    i. La fiecare rulare a buclei, se actualizează contorul pentru adăugarea de obstacole și pentru a se actualiza timpul scurs.
    ii. Se generează obstacole la intervale regulate, și se verifică coliziunile cu nava spațială.
    iii. Se verifică evenimentele Pygame, apăsarea tastelor, inclusiv închiderea ferestrei.
    iv. Se realizează mișcarea navei spațiale în funcție de tastele apăsate.
    v. Se realizează mișcarea obstacolelor și se verifică coliziunile cu nava spațială.
    vi. Dacă a avut loc o coliziune, jocul se oprește și se afișează un mesaj corespunzător.

vii. Se desenează elementele pe ecran utilizând funcția draw.

5. Rularea principală, se verifică dacă scriptul este rulat direct și, în acest caz, se apelează funcția principala pentru a începe jocul.

Referințe

- https://www.youtube.com/watch?v=waY3LfJhQLY
- https://github.com/techwithtim/Python-Game-Dev-Intro
- https://docs.python.org/3/library/sys.html
- https://www.pygame.org/docs/
- https://www.youtube.com/watch?v=tJiKYMQJnYg
- https://chat.openai.com/share/d8542ebf-443e-4616-93c4-51e3d024e4b3
- https://chat.openai.com/share/9f5b264d-2d22-4639-aead-96fd3bb98e29
- https://chat.openai.com/share/bfd4f60f-9935-469b-a3a4-4f2deb632084

Github Repository

https://github.com/RaresNitu03/SpaceDodge.py