# LUCRARE PENTRU ATESTAREA COMPETENȚELOR PROFESIONALE LA INFORMATICĂ

Prof. Coordonator: Cîrdeiu Emilia

Candidat: Baltă Alexandru

# - Space Dodge-

**Cuprins**

[Motivația alegerii temei 5](#_Toc133946779)

[Limbaj de Programare folosit 6](#_Toc133946780)

[**Python** 6](#_Toc133946781)

[**Pygame** 10](#_Toc133946782)

[Descrierea softului utilizat 11](#_Toc133946783)

[Prezentarea aplicației 13](#_Toc133946784)

[Structura aplicației 16](#_Toc133946785)

[**1.** **main.py :** 16](#_Toc133946786)

[**2.** **player.py** 16](#_Toc133946787)

[**3.** **asteroid.py** 17](#_Toc133946788)

[**4.** **screen.py** 17](#_Toc133946789)

[**5.** **sprite\_sheet.py** 17](#_Toc133946790)

[Cod sursă 18](#_Toc133946791)

[1. Codul sursă al paginii ‘main.py’ 18](#_Toc133946792)

[2. Codul sursă al paginii ‘player.py’ 19](#_Toc133946793)

[3. Codul sursă al paginii ‘asteroid.py’ 22](#_Toc133946794)

[4. Codul sursă al paginii ‘screen.py’ 24](#_Toc133946795)

[5. Codul sursă al paginii ‘sprite\_sheet.py’ 31](#_Toc133946796)

[Concluzii 32](#_Toc133946797)

[Bibliografie 33](#_Toc133946798)

[Youtube videos: 33](#_Toc133946799)

[Websites: 33](#_Toc133946800)

# Motivația alegerii temei

**Tema acestui proiect la informatică este «*Joc Retro*».**

**Sunt de părere că toți dintre noi am auzit de jocuri vechi, retro. Jocuri precum Tetris, Mario Bros, Space Invaders și multe altele. Aceste jocuri am făcut furori la lansarea acestora, fiind niște jocuri care ne-au făcut copilăria mai frumoasă multora dintre noi.**

**Țin minte că eram copil și într-o zi a venit un prieten de-a lui tata acasă. Eu la vârsta de 6 ani, nu prea știam ce face acel domn, dar știu că înainte să plece m-a condus spre o măsuța din camera mea. La început nu știam ce este schmbat la măsuță, dar stau, mă uit mai bine și văd o cutie gri chiar în stânga măsuței. Eram surprins, dar oarecum intrigat de prezența acelei cuti gri. După aceea acel domn a scos dintr-o punga un ecran cu piciorușe și l-a pus pe măsuță. Și dupa ce l-am deschis am fost uimit. Mi-am zis «Uau! Un televizor pe care pot sa fac eu ce vreau. Ce grozav!».**

**La inceput nu aveam conexiune la internet pe acel computer, dar totusi era un folder pe acel calculator. Iar in acel folder erau o multime de jocuri. Curios fiind am apasat pe o iconita care mi a atras atentia, iar aplicatia care s-a deschis a fost un joc cu un omulet care se ferea de niste asteroizi care picau din cer. Mi s-a parut o idee geniala de joc si m-am jucat vreo saptamana numai acel joculet, char daca calculatorul venea si cu alte jocuri.**

**Asa ca pentru aceast atestat am ales sa recreez char acel joculet. Nu mai stiu numele jocului, dar am o oarecare imagine in memorie cu el si am incercat sa o recreez in acest joculet. Nu am ales alt joc deoarece acesta a fost prima mea interactiune cu jocurile video si chiar daca erau alte jocuri mai interesante si jucate de mai multi oameni, acesta are un loc special in inima mea.**

# Limbaj de Programare folosit

## **Python**

Python este un limbaj de programare de înalt nivel, interpretat și orientat-obiect. A fost creat în anii 1990 de Guido van Rossum și este unul dintre cele mai populare limbaje de programare în prezent. Python este utilizat într-o gamă largă de domenii, inclusiv dezvoltarea web, știința datelor, inteligența artificială, automatizarea proceselor și multe altele. Una dintre caracteristicile cheie ale Python este sintaxa sa clară și concisă, care face codul mai ușor de citit și de înțeles decât în alte limbaje de programare. Python utilizează spații și indentări pentru a indica blocurile de cod, în locul acoladelor sau parantezelor, ceea ce face codul mai curat și mai ușor de citit.

Python are și o comunitate puternică de dezvoltatori care lucrează la îmbunătățirea limbajului și dezvoltarea de noi instrumente și biblioteci.Una dintre cele mai mari avantaje ale Python este ușurința de a învăța și de a începe să programezi. Sintaxa clară și concisă face ca codul să fie ușor de înțeles și de scris, ceea ce face din Python o alegere populară pentru începătorii în programare.Python este, de asemenea, un limbaj flexibil și extensibil, care poate fi utilizat pentru o gamă largă de sarcini. Python poate fi utilizat pentru dezvoltarea de aplicații desktop, dezvoltarea web, procesarea datelor, automatizarea proceselor și multe altele.Python este, de asemenea, un limbaj popular în domeniul inteligenței artificiale și al învățării automate. Python are o serie de biblioteci și instrumente specializate pentru aceste domenii, cum ar fi TensorFlow, Keras și PyTorch.Python are o serie de caracteristici cheie care îl fac un limbaj puternic și eficient pentru dezvoltatori. Una dintre aceste caracteristici este gestionarea automată a memoriei, care ajută la evitarea erorilor de memorie și face codul mai sigur și mai ușor de utilizat.Python este, de asemenea, un limbaj orientat-obiect, ceea ce înseamnă că permite dezvoltatorilor să organizeze codul în obiecte reutilizabile, ceea ce face codul mai ușor de întreținut și de modificat în timp. Python este, de asemenea, un limbaj interpretat, ceea ce înseamnă că codul este rulat direct de către interpretorul Python, fără a fi compilat înainte de a fi executat. Acest lucru face ca procesul de dezvoltare să fie mai rapid și mai eficient, deoarece nu este nevoie să se aștepte timpul de compilare.

Python are o gamă largă de biblioteci și framework-uri disponibile, care permit dezvoltatorilor să creeze aplicații rapid și eficient. Python are o bibliotecă standard puternică, care include o gamă largă de funcții și module pentru diverse sarcini, cum ar fi lucru cu fișiere, procesare de text, networking, criptare și multe altele. Python are și o gamă largă de biblioteci de terțe părți, care sunt disponibile prin intermediul PyPI (Python Package Index), inclusiv NumPy, Pandas, Django, Flask și multe altele. Python este, de asemenea, un limbaj portabil și multiplatformă, care poate fi rulat pe o gamă largă de sisteme de operare, inclusiv Windows, Mac OS X și diverse distribuții Linux. Acest lucru face din Python o alegere populară pentru dezvoltatorii care doresc să creeze aplicații care rulează pe diverse platforme.Python este, de asemenea, un limbaj puternic pentru procesarea datelor și analiza datelor. Python are o gamă largă de biblioteci și instrumente specializate pentru aceste domenii, cum ar fi Pandas, NumPy, SciPy, Matplotlib și multe altele.Python are o sintaxă clară și concisă, care îl face ușor de învățat și de utilizat. Python utilizează indentarea pentru a indica blocurile de cod, în locul acoladelor sau parantezelor, ceea ce face codul mai curat și mai ușor de citit. Python are, de asemenea, o gamă largă de funcții și module, care permit dezvoltatorilor să creeze aplicații complexe și sofisticate, folosind un număr relativ mic de linii de cod.

Python este, de asemenea, un limbaj flexibil și extensibil, care poate fi utilizat pentru o gamă largă de sarcini. Python poate fi utilizat pentru dezvoltarea de aplicații desktop, dezvoltarea web, procesarea datelor, automatizarea proceselor și multe altele. Python are, de asemenea, o gamă largă de framework-uri și biblioteci disponibile pentru aceste sarcini, inclusiv Flask, Django, NumPy, Pandas, TensorFlow și multe altele. Python este un limbaj open-source, ceea ce înseamnă că poate fi utilizat gratuit de către oricine și poate fi modificat și distribuit în mod liber. Acest lucru face din Python o alegere populară pentru dezvoltatorii care doresc să lucreze cu tehnologii open-source.

Python este unul dintre cele mai popluare limbaje în domeniul inteligenței artificiale și al învățării automate. Python are o serie de biblioteci și instrumente specializate pentru aceste domenii, cum ar fi TensorFlow, Keras, PyTorch, Scikit-learn și multe altele. Aceste biblioteci și instrumente permit dezvoltatorilor să creeze modele de învățare automată și de inteligență artificialPython este, de asemenea, un limbaj popular în domeniul dezvoltării de jocuri. Python are o serie de biblioteci și framework-uri specializate pentru dezvoltarea de jocuri, cum ar fi Pygame, PyOpenGL și Panda3D. Aceste biblioteci și framework-uri permit dezvoltatorilor să creeze jocuri 2D și 3D, folosind Python ca limbaj principal.Python este, de asemenea, un limbaj popular pentru dezvoltarea de aplicații web. Python are o serie de framework-uri specializate pentru dezvoltarea web, cum ar fi Flask, Django, Pyramid și Bottle. Aceste framework-uri permit dezvoltatorilor să creeze aplicații web complexe și scalabile, folosind Python ca limbaj principal.Python este, de asemenea, un limbaj popular pentru automatizarea proceselor. Python poate fi utilizat pentru crearea de scripturi de automatizare a proceselor, cum ar fi procesarea de fișiere, transferul de fișiere, extragerea de date și multe altele. Python are o serie de module specializate pentru aceste sarcini, cum ar fi os, shutil și pathlib.

Python este, de asemenea, un limbaj popular în domeniul securității informatice. Python poate fi utilizat pentru crearea de instrumente de securitate, cum ar fi scanere de vulnerabilități, analizatoare de pachete și multe altele. Python are o serie de biblioteci specializate pentru aceste sarcini, cum ar fi Scapy și PyCrypto. Python este, de asemenea, un limbaj popular pentru dezvoltarea de aplicații pentru dispozitive mobile. Python poate fi utilizat pentru crearea de aplicații mobile, folosind framework-uri precum Kivy și BeeWare. Aceste framework-uri permit dezvoltatorilor să creeze aplicații mobile cross-platform, care rulează pe iOS și Android.

Documentația Python este ușor de înțeles și de navigat, ceea ce face ca învățarea și utilizarea limbajului Python să fie mai ușoare și mai eficientă.Python este un limbaj interpretat, ceea ce înseamnă că nu trebuie să fie compilat înainte de a fi rulat. Acest lucru face ca dezvoltarea și testarea aplicațiilor să fie mai rapidă și mai eficientă decât în cazul altor limbaje de programare, cum ar fi C++ sau Java.Python are și o bibliotecă standard foarte puternică, care oferă o gamă largă de funcționalități. Această bibliotecă standard include module pentru procesarea de fișiere, procesarea de date, comunicarea cu rețelele, criptografie, și multe altele. Acest lucru face ca dezvoltarea de aplicații în Python să fie mai ușoară și mai rapidă, deoarece dezvoltatorii nu trebuie să scrie cod de la zero.Python este un limbaj open-source, ceea ce înseamnă că poate fi utilizat gratuit de către oricine și poate fi modificat și distribuit în mod liber.În plus, Python este un limbaj care se dezvoltă continuu. Versiunile noi ale limbajului Python sunt lansate frecvent, adăugând noi caracteristici și funcționalități. Acest lucru face ca Python să fie un limbaj viabil și actual pentru dezvoltarea de aplicații în viitor.Un alt avantaj major al limbajului Python este versatilitatea sa. Python poate fi utilizat într-o gamă largă de domenii, cum ar fi dezvoltarea de aplicații web, dezvoltarea de jocuri, analiza datelor, inteligența artificială și machine learning, automatizarea proceselor, și multe altele. Acest lucru se datorează în mare parte bibliotecilor și framework-urilor disponibile în Python, care oferă funcționalități specifice pentru diferite domenii.Python este un limbaj simplu și intuitiv, care îl face potrivit pentru începători în programare, dar și pentru programatori experimentați. De asemenea, Python este un limbaj care încurajează bunele practici de programare, cum ar fi utilizarea indentării pentru structurarea codului și scrierea de cod clar și concis.Python este un limbaj cross-platform, ceea ce înseamnă că poate fi utilizat pe mai multe sisteme de operare, cum ar fi Windows, Linux și macOS. Acest lucru face ca dezvoltarea de aplicații în Python să fie mai ușoară și mai accesibilă pentru dezvoltatorii care lucrează pe diferite platforme.

În plus, Python are o gamă largă de module și framework-uri disponibile, care permit dezvoltatorilor să creeze aplicații complexe și scalabile. Framework-uri precum Flask și Django sunt utilizate pentru dezvoltarea de aplicații web, în timp ce biblioteci precum NumPy și Pandas sunt utilizate pentru analiza datelor și machine learning.

Mai mult decat atat, Python este un limbaj care încurajează bunele practici de programare și este ușor de citit și de întreținut. Python este un limbaj open-source și gratuit, ceea ce face ca dezvoltarea de aplicații în Python să fie accesibilă pentru oricine. Aceste caracteristici și avantaje ale limbajului Python îl fac unul dintre cele mai populare și puternice limbaje de programare din lume.

În concluzie, Python este un limbaj de programare popular și puternic, care este utilizat într-o gamă largă de aplicații. Python are o sintaxă clară și concisă, care îl face ușor de învățat și de utilizat. Python are o comunitate mare și activă de dezvoltatori, care contribuie la dezvoltarea și îmbunătățirea limbajului Python. Python are, de asemenea, o documentație excelentă și o bibliotecă standard puternică, care fac dezvoltarea de aplicații mai rapidă și mai eficientă. Python este un limbaj interpretat, ceea ce face ca dezvoltarea și testarea aplicațiilor să fie mai rapidă și mai eficientă decât în cazul altor limbaje de programare. Python este un limbaj open-source, ceea ce înseamnă că poate fi utilizat gratuit de către oricine și poate fi modificat și distribuit în mod liber.

## **Pygame**

Pygame este o bibliotecă de programare open source, specializată pentru dezvoltarea de jocuri și aplicații multimedia. Pygame a fost dezvoltat pentru a fi folosit în limbajul de programare Python și oferă o serie de funcționalități pentru a crea jocuri și alte aplicații interactive. Acest document va prezenta o descriere detaliată a pygame, includând istoricul, arhitectura, caracteristicile și exemple de utilizare.

*Istoric:* Pygame a fost inițial dezvoltat în anul 2000 de Pete Shinners, care a dorit să creeze o bibliotecă de programare Python pentru dezvoltarea de jocuri. El a folosit biblioteca SDL (Simple DirectMedia Layer) pentru a crea o interfață între Python și hardware-ul calculatorului. Pygame a fost dezvoltat inițial pentru a fi folosit pe platforma Linux, dar acum este disponibil pe mai multe sisteme de operare, inclusiv Windows, macOS și iOS.

*Arhitectura:* Pygame utilizează biblioteca SDL pentru a crea o interfață între Python și hardware-ul calculatorului. Aceasta permite accesul la funcții de afișare grafică, sunet și intrare utilizator. Pygame oferă o serie de module, cum ar fi module pentru grafică, sunet și intrare, fiecare oferind o serie de funcții pentru a crea aplicații multimedia.

*Caracteristici:* Pygame oferă o serie de caracteristici care permit dezvoltatorilor să creeze jocuri și aplicații multimedia de înaltă calitate. Unele dintre aceste caracteristici includ:

*Grafică:* Pygame oferă o serie de funcții pentru a afișa imagini, animații și texte pe ecran. Dezvoltatorii pot utiliza aceste funcții pentru a crea interfețe utilizator atractive, jocuri cu grafică 2D și alte aplicații multimedia.

*Sunet:* Pygame oferă funcții pentru a reda sunete și muzică, precum și pentru a crea efecte sonore pentru jocuri și alte aplicații multimedia.

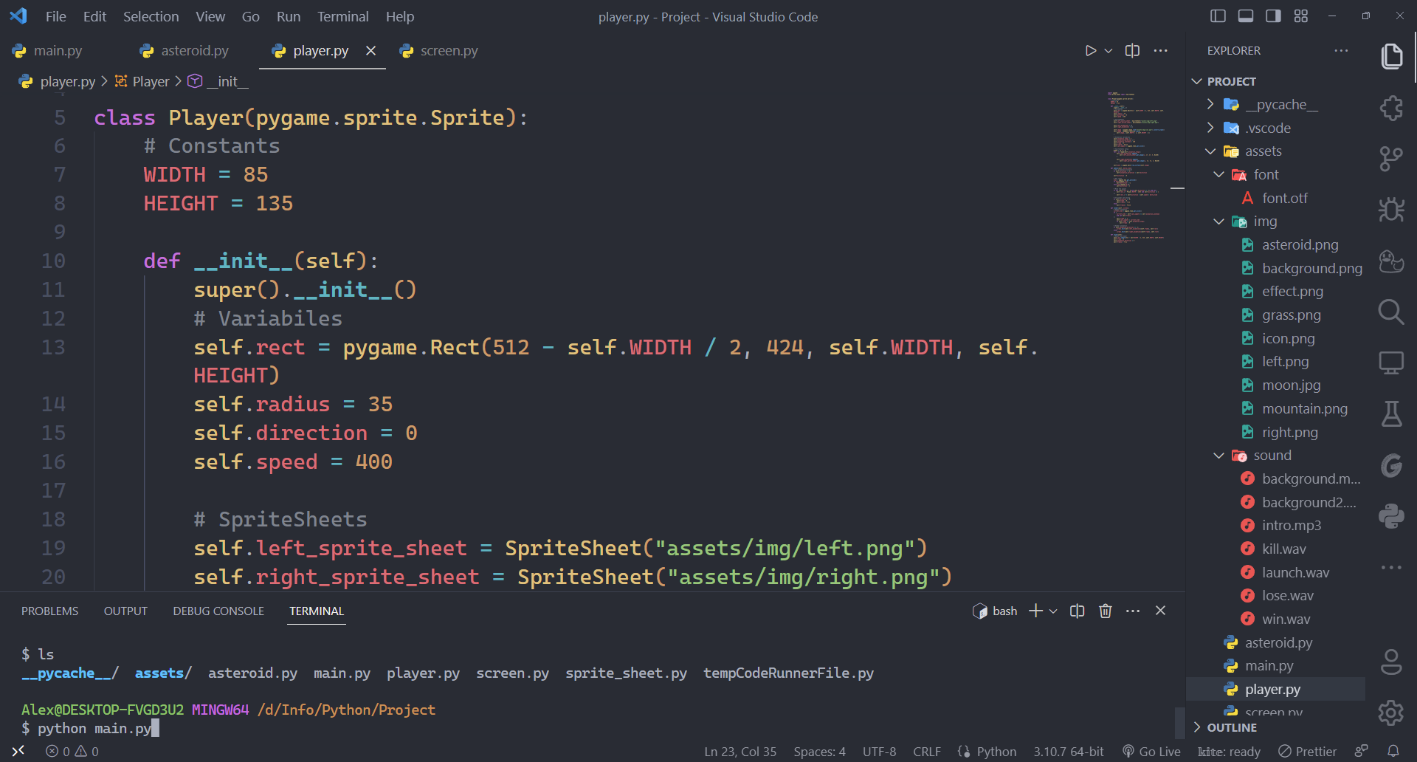
*Intrare:* Pygame oferă funcții pentru a detecta intrările utilizator, cum ar fi tastatura, mouse-ul și gamepad-urile. Dezvoltatorii pot utiliza aceste funcții pentru a crea interacțiune cu utilizatorul în jocuri și alte aplicații multimedia.

*Coliziuni:* Pygame oferă funcții pentru a detecta coliziunile între obiecte pe ecran, ceea ce face posibilă crearea de jocuri cu detecție de coliziune

# Descrierea softului utilizat

Visual Studio Code este un editor de text gratuit, cross-platform, dezvoltat de Microsoft. Acesta devine rapid un favorit datorită performanței sale excelente și exuberantului număr de caracteristici pe care le oferă.

La fel ca majoritatea IDE-urilor, VSCode are o piață de extindere care conține mii de pluginuri cu o calitate diferită. Pentru a vă ajuta să alegeți cele care merită descărcate, am făcut această colecție de extensii pe care le-am găsit cele mai utile și mai distractive.



Prettier

Prettier este cel mai popular formatator de coduri în web dev chiar acum. Permite codului echipei dvs. să arate la fel, indiferent cine a scris-o. Această extensie face posibilă aplicarea automată a Prettier și formarea rapidă a documentelor JS și CSS.

TODO Highlight

Acest instrument marchează toate comentariile TODO din codul dvs., ceea ce face mai ușor să urmăriți orice afacere neterminată înainte de a împinge la producție. În mod implicit, acesta caută cuvinte cheie TODO și FIXME, dar puteți adăuga și expresiile dvs. Personalizate.

Themes

Desigur, cele mai importante dintre toate extensiile sunt temele. Veți căuta în fiecare zi editorul dvs. VSCode, de ce să nu faceți mai frumos? Există tone de pluginuri de personalizare care schimba schema de culori și pictogramele din bara laterală.

Audio Preview

**O altă extensie importantă este Audio Preview care te lasă să auzi în aplicație fișierele de tipul .mp3 sau .wav. Acest lucru nu este în Visual Studio Code, așa că această extensie implementează acest lucru.**

# Prezentarea aplicației

Aplicația este recrearea mea a unui joc jucat de mine în copilărie, joc care are următoarea logică. Ești un astronaut și te afli pe o altă planetă. Iar scopul tău este să supraviețuiești în timp ce asteroizi cad din cer. Tu trebuie să îi eviți mișcându-te stânga-dreapta. Asteroizi cad la fiecare jumătate de secundă.

Acum că am vorbit de logica jocului, hai să începem cu începutul și anume ecranul de pornire. Când se deschide jocul rulează 2 secunde un ecran în care este specificat creatorul jocului.



**Acest pop-up apare pentru 2 secunde, iar mai apoi se trece la ecranul de pornire unde este prezent mare, pe centrul ecranului, titlul jocului, care se mișcă sus și jos, și o indicație pe care scrie “Press Enter to play”.**



După ce jucătorul apasă ‘enter’, o să fie trimis pe ecranul în care începe jocul propriu-zis. Pe acel ecran o să se observe scorul în partea din stânga sus. Jucătorul o să observe în partea din mijloc jos un astronaut, iar acesta trebuie să-l controleze cu A, dacă vrea să meargă în stânga, și cu D dacă vrea să meargă în dreapta. La scurt timp după asta, asteroizii cad la un interval de jumătate de secundă unul de altul, iar jucătorul trebuie să încerce să stea cât mai mult în viață.

Pe ecran se poate observa că locația clar nu este pe pământ, ci mai degrabă pe o altă planetă, poate chiar pe lună, observându-se munți de o culoare gri în spatele jucătorului.

Scorul reține fiecare secundă pe care jucătorul o petrece încercând să reziste. Se poate vedea o animație pe care jucătorul o are atunci când merge, de parcă ar topăi cumva, dar jocul, cum are tematica “Retro Games”, animația este făcută sacadat și drăguț în același timp.



**Din momentul în care un asteroid intră în contact cu jucătorul, se oprește muzica și se aude un ‘Game Over’ pe fundal. Nu doar că se aude, dar este și scris mare acesta pe ecran și încă o indicație care îi sugerează jucătorului că, dacă vrea să reînceapă jocul, trebuie să apese ‘enter’. Cadrul este pus pe pauză în momentul în care jucătorul este lovit și se poate vedea clar contactul cu asteroidul.**



**După ce jucătorul apasă pe 'enter', acesta este pus iar în ipostaza de a juca jocul, cu scorul evident resetat. Și tot așa. Acest joc nu are final, nu poți câștiga, este doar vorba despre scor și dacă ai reușit să ai un scor mai mare decât prietenii tăi. Dacă dorești să ieși, poți apăsa pe butonul 'x' pe care îl au toate aplicațiile sau să apeși pe 'esc' atunci când ești pe ecranul de Game Over.**

# Structura aplicației

**Aplicația a fost realizată în cea mai mare parte în limbajul Python, prin programare direct în cod sursă. Scrierea liniilor de comandă s-a făcut cu Visual Studio Code, vizualizarea codului putându-se realiza cu orice editor text.**

Aplicația este împărțită în 5 fișiere cheie. Fiecare fișier are un rol diferit, dar toate sunt necesare pentru a rula aplicația. O să explic, pe scurt, care sunt fișierele, denumirea acestora și ce face fiecare.

## **main.py :**

**Primul fișier necesar este main.py. In general toate aplicațiile folosesc un fișier main de unde incepe rularea codului. Câteva limbaje de programare au implementat chiar o funcție care se cheamă main(), dar din pacate python-ul nu dispune de asa ceva. Așa că am implementat eu o functie main. In principiu in fișier îmi inițializez pygame-ul și câteva variabile cum ar fi clock– pentru crearea unui framerate constant, screen – pentru a creea obiectul de tip Screen. Mai apoi am facut un loop care permite programului să ruleze la nesfârșit sau pană îi spun eu să se oprească. Dau Update la ecran si apoi Render la acesta, iar apoi setez fps-ul și creez delta time, care este timpul care trece dupa fiecare frame.**

## **player.py**

**Programul meu are nevoie de un player, asa ca am decis sa fac o classa de Player, in care îmi creez o schița pentru un asa zis obiect de tip Player. Aceasta clasa are câteva funcții/metode interesante, pritre care și Init, care initializeaza variabilele pe care le contine clasa, Update, unde setez directia jucatorului și il fac sa se miște in functie de aceasta, Render, unde animez jucatorul și il afisez pe ecran și Reset unde resetez player-ul la pozitia initiala, dupa ce pierde.**

## **asteroid.py**

**Programul meu are nevoie de asemenea de obiecte care să cada din cer aka. asteroizi. Asa ca am decis sa creez o classa de Asteroid unde imi initializes toate variabile pe care un asteroid ar trebui sa le aibe, cum ar fi imagine, marime, etc. Am creat si o classa pe nume Asteroids unde ma ocup cu crearea asteroiziolor si manipularea acestora pe ecran. Clasa mea de Asteroids are, la fel ca Player, aceleași funții. Metodele fac cam acelasi lucru ca si la Player, in sensul ca Update se ocupa cu miscarea asteroizilor, Reder, cu afisarea lor pe ecran, etc.**

## **screen.py**

**Acest fisier este defapt cel ce face cam toata treaba. De altfel este și cel mai lung. In acesta am câteva clase draguțe care ma ajuta să realizez ecranele, adica ecranul de la inceputul jocului, cel prezent in timpul in care joci și ecranul in momentul in care pierzi. Am realizat acest lucru făcând câteva classe cum ar fi - Screen: Unde creez blueprint-ul pentru ecranul in sine și schimb intre ecrane. - StartScreen: Aici este eveident dupa nume ca creez ecranul de pornire. - PlayScreen: Este pentru ecranul din timpul in care joci. - GameOverScreen: Unde realizez ecranul, in momentul in care cade un asteroid pe tine. Toate aceste clase realizeaza ce isi propun fiecare prin anumite functii si variabile, functii care se repeta la nivelun fiecarei clase. De exemplu functia Update si functia Render.**

## **sprite\_sheet.py**

**Acest fisier este un fisier de dimensiune mica in care imi creez o clasa ajutatoare care ma ajuta sa creez sprite-uri, adica sa iau imagile pentru player si pentru astaroid.**

# Cod sursă

## Codul sursă al paginii ‘main.py’

import pygame

import sys

from screen import Screen

FPS = 60

def **main**():

    pygame.**init**()

    screen = Screen()

    clock = pygame.time.Clock()

    previous\_time = pygame.time.**get\_ticks**()

    while True:

        for event in pygame.event.**get**():

            if event.type == pygame.QUIT:

                pygame.**quit**()

                sys.**exit**()

        present\_time = pygame.time.**get\_ticks**()

        delta\_time = (present\_time - previous\_time) / 1000

        previous\_time = present\_time

        screen.**update**(delta\_time)

        screen.**render**()

        clock.**tick**(FPS)

if \_\_name\_\_ == "\_\_main\_\_":

**main**()

## Codul sursă al paginii ‘player.py’

import pygame

from sprite\_sheet import SpriteSheet

class Player(pygame.sprite.Sprite):

    # Constants

    WIDTH = 85

    HEIGHT = 135

    def **\_\_init\_\_**(self):

        super().**\_\_init\_\_**()

        # Variabiles

        self.rect = pygame.Rect(512 - self.WIDTH / 2, 424, self.WIDTH, self.HEIGHT)

        self.radius = 35

        self.direction = 0

        self.speed = 400

        # SpriteSheets

        self.left\_sprite\_sheet = SpriteSheet("assets/img/left.png")

        self.right\_sprite\_sheet = SpriteSheet("assets/img/right.png")

        self.left\_animations = []

        self.right\_animations = []

        self.image = pygame.image.**load**("assets/img/icon.png").**convert\_alpha**()

        self.image = pygame.transform.**scale**(

            self.image, (self.WIDTH / 2, self.HEIGHT / 2)

        )

        # Animation variabiles

        self.animation\_steps = 3

        self.animation\_direction = 0

        self.animation\_cooldown = 150

        self.frame = 0

        self.freeze = False

        self.last\_update = pygame.time.**get\_ticks**()

        # Spritesheets setup

        BLACK = (0, 0, 0)

        for i in range(self.animation\_steps):

            self.left\_animations.**append**(

                self.left\_sprite\_sheet.**get\_image**(i, 17, 27, 5, BLACK)

            )

            self.right\_animations.**append**(

                self.right\_sprite\_sheet.**get\_image**(i, 17, 27, 5, BLACK)

            )

        self.mask = pygame.mask.**from\_surface**(self.image)

    def **update**(self, delta\_time):

        # Set animation direction

        if self.direction != 0:

            self.animation\_direction = self.direction

        self.direction = 0

        # Get input

        keys = pygame.key.**get\_pressed**()

        if keys[pygame.K\_a]:

            self.direction = -1

        elif keys[pygame.K\_d]:

            self.direction = 1

        # Move the player

        if not (self.rect.x < 0 and self.direction == -1) and not (

            self.rect.x + Player.WIDTH > 1024 and self.direction == 1

        ):

            self.rect.x += self.direction \* self.speed \* delta\_time

        # If player not moving

        if self.direction == 0:

            self.frame = 0

            self.freeze = True

else:

            self.freeze = False

    def **render**(self, screen):

        # Update animation

        current\_time = pygame.time.**get\_ticks**()

        if (

            current\_time - self.last\_update >= self.animation\_cooldown

            and not self.freeze

        ):

            self.frame += 1

            self.last\_update = current\_time

            if self.frame >= self.animation\_steps:

                self.frame = 0

        # Render animation

        if self.animation\_direction == -1:

            screen.blit(self.left\_animations[self.frame], self.rect)

        else:

            screen.blit(self.right\_animations[self.frame], self.rect)

    def **reset**(self):

        # Reset the player

        self.rect.**update**(512 - self.WIDTH / 2, 424, self.WIDTH, self.HEIGHT)

        self.frame = 0

        self.animation\_direction = 0

        self.freeze = True

## Codul sursă al paginii ‘asteroid.py’

import pygame

import random

class Asteroid(pygame.sprite.Sprite):

    # Constant

    SCALE = 150

    def **\_\_init\_\_**(self, x, y):

        super().**\_\_init\_\_**()

        # Image

        self.image = pygame.transform.**scale**(

            pygame.image.**load**("assets/img/asteroid.png"),

            (self.SCALE, self.SCALE),

        )

        # Variabiles

        self.rect = self.image.**get\_rect**()

        self.rect.**update**(x, y, self.SCALE, self.SCALE)

        self.radius = 65

        self.mask = pygame.mask.**from\_surface**(self.image)

class Asteroids:

    # Constants

    SCALE = 150

    SPEED = 600

    def **\_\_init\_\_**(self):

        # Variabiles

        self.asteroid\_list = []

        self.spawn\_time = 0.5

        self.time = 0

    def **update**(self, delta\_time):

        self.time += delta\_time

        # Spawn an asteroid

        if self.time > self.spawn\_time:

            self.asteroid\_list.**append**(

                Asteroid(random.randint(0, 1024 - self.SCALE), -300)

            )

            # Delete an asteroids when goes offscreen

            if self.asteroid\_list[0].rect.y > 500:

                self.asteroid\_list.**pop**(0)

            self.time = 0

        # Move asteroids

        for asteroid in self.asteroid\_list:

            asteroid.rect.y += delta\_time \* self.SPEED

    def **render**(self, screen):

        # Draw asteroids

        for asteroid in self.asteroid\_list:

            screen.blit(asteroid.image, asteroid.rect)

    def **reset**(self):

        # Clear asteroids from list

        self.time = 0

        self.asteroid\_list.**clear**()

## Codul sursă al paginii ‘screen.py’

import pygame

import math

import sys

from player import Player

from asteroid import Asteroids

class Screen:

    # Constants

    global WIDTH, HEIGHT, GRASS, BLACK, WHITE

    WIDTH = 1024

    HEIGHT = 640

    GRASS = 128

    BLACK = (0, 0, 0)

    WHITE = (255, 255, 255)

    def **\_\_init\_\_**(self):

        # Screen

        global screen

        screen = pygame.display.**set\_mode**((WIDTH, HEIGHT))

        pygame.display.**set\_icon**(pygame.image.**load**("assets/img/icon.png"))

        pygame.display.**set\_caption**("Space Dodge")

        # Images

        global background\_tex, mountains\_tex, grass\_tex

        background\_tex = pygame.transform.**scale**(

            pygame.image.**load**("assets/img/background.png"), (WIDTH, HEIGHT)

        )

        mountains\_tex = pygame.transform.**scale**(

            pygame.image.**load**("assets/img/mountain.png"), (WIDTH, 262)

        )

        grass\_tex = pygame.transform.**scale**(

            pygame.image.**load**("assets/img/grass.png"), (GRASS, GRASS)

        )

        # Fonts

        global font\_150, font\_69

        font\_150 = pygame.font.Font("assets/font/font.otf", 150)

        font\_69 = pygame.font.Font("assets/font/font.otf", 69)

        # Sounds

        global launch\_sfx, lose\_sfx, start\_sfx, intro\_sfx

        launch\_sfx = pygame.mixer.Sound("assets/sound/launch.wav")

        lose\_sfx = pygame.mixer.Sound("assets/sound/lose.wav")

        start\_sfx = pygame.mixer.Sound("assets/sound/win.wav")

        intro\_sfx = pygame.mixer.Sound("assets/sound/intro.mp3")

        pygame.mixer.music.**load**("assets/sound/background2.mp3")

        pygame.mixer.music.**set\_volume**(0.05)

        # Objects

        self.start\_screen = StartScreen()

        self.play\_screen = PlayScreen()

        self.gameover\_screen = GameOverScreen()

        # Enum :))

        global START\_SCREEN, PLAY\_SCREEN, GAMEOVER\_SCREEN

        START\_SCREEN, PLAY\_SCREEN, GAMEOVER\_SCREEN = 0, 1, 2

        # Variabiles

        global current\_screen

        current\_screen = START\_SCREEN

    def **update**(self, delta\_time):

        # Check the type of screen and updates it

        if current\_screen == START\_SCREEN:

            self.start\_screen.**update**(delta\_time)

        elif current\_screen == PLAY\_SCREEN:

            self.play\_screen.**update**(delta\_time)

        elif current\_screen == GAMEOVER\_SCREEN:

            self.gameover\_screen.**update**()

    def **render**(self):

        # Check the type of screen and draws it

        if current\_screen == START\_SCREEN:

            self.start\_screen.**render**()

        elif current\_screen == PLAY\_SCREEN:

            self.play\_screen.**render**()

        elif current\_screen == GAMEOVER\_SCREEN:

            self.gameover\_screen.**render**()

        pygame.display.**flip**()

class PlayScreen:

    def **\_\_init\_\_**(self):

        # Objects

        global player, asteroids

        player = Player()

        asteroids = Asteroids()

        self.seconds = 0

        self.minutes = 0

        self.score = 0

    def **check\_collision**(self):

        for ast in asteroids.asteroid\_list:

            if pygame.sprite.**collide\_circle**(player, ast):

                global current\_screen

                current\_screen = GAMEOVER\_SCREEN

                self.seconds = 0

                self.score = 0

                self.minutes = 0

                pygame.mixer.music.**stop**()

                pygame.mixer.Sound.**play**(lose\_sfx)

    def **update\_score**(self, delta\_time):

        self.score += delta\_time

        if self.score >= 1:

            self.seconds += 1

            self.score = 0

        if self.seconds == 60:

            self.seconds = 0

            self.minutes += 1

    def **render\_score**(self):

        if self.seconds < 10:

            self.zero = "0"

        else:

            self.zero = ""

        self.score\_white = font\_69.**render**(

            f"SCORE: {str(self.minutes)}:{self.zero}{str(self.seconds)}", True, WHITE

        )

        self.score\_black = font\_69.**render**(

            f"SCORE: {str(self.minutes)}:{self.zero}{str(self.seconds)}", True, BLACK

        )

        screen.**blit**(self.score\_black, (10, 5))

        screen.**blit**(self.score\_white, (5, 0))

    def **update**(self, delta\_time):

        self.**update\_score**(delta\_time)

        # Updates player and asteroid

        player.**update**(delta\_time)

        asteroids.**update**(delta\_time)

        self.**check\_collision**()

    def **render**(self):

        # Render backgraound

        screen.**blit**(background\_tex, (0, 0))

        screen.**blit**(mountains\_tex, (0, 284))

        asteroids.**render**(screen)

        # Render grass

        for i in range(8):

            screen.**blit**(grass\_tex, (i \* 128, 524))

        player.**render**(screen)

        self.**render\_score**()

class GameOverScreen:

    def **\_\_init\_\_**(self):

        # Strings of text

        self.gameover\_white = font\_150.**render**("GAME OVER", True, WHITE)

        self.gameover\_black = font\_150.**render**("GAME OVER", True, BLACK)

        self.press\_enter\_white = font\_69.**render**("Press Enter", True, WHITE)

        self.press\_enter\_black = font\_69.**render**("Press Enter", True, BLACK)

    def **update**(self):

        keys = pygame.key.**get\_pressed**()

        # Resets the game and returns to the Play Screen

        if keys[pygame.K\_RETURN]:

            asteroids.**reset**()

            player.**reset**()

            global current\_screen

            current\_screen = PLAY\_SCREEN

            pygame.mixer.music.**play**(-1)

        # If you press esc it quits the game

        elif keys[pygame.K\_ESCAPE]:

            pygame.**quit**()

            sys.**exit**()

    def **render**(self):

        # Drawing the strings of text

        screen.**blit**(self.gameover\_black, (210, 135))

        screen.**blit**(self.gameover\_white, (200, 125))

        screen.**blit**(self.press\_enter\_black, (365, 305))

        screen.**blit**(self.press\_enter\_white, (360, 300))

class StartScreen:

    def **\_\_init\_\_**(self):

        # string of texts

        self.space\_white = font\_150.**render**("Space", True, WHITE)

        self.space\_black = font\_150.**render**("Space", True, BLACK)

        self.dodge\_white = font\_150.**render**("Dodge", True, WHITE)

        self.dodge\_black = font\_150.**render**("Dodge", True, BLACK)

        self.startMessage\_black = font\_69.**render**("Press enter to play", True, BLACK)

        self.startMessage\_white = font\_69.**render**("Press enter to play", True, WHITE)

        self.splashMessage\_white = font\_69.**render**("CREATED BY BALTA ALEX", True, WHITE)

        self.splashMessage\_black = font\_69.**render**("CREATED BY BALTA ALEX", True, BLACK)

        # Variables

        self.splash\_timer = 0

        self.splash\_sound = False

        self.entry\_sound = False

    def **update**(self, delta\_time):

        # Splash Screen animation

        if self.splash\_timer < 2:

            self.splash\_timer += delta\_time

            if self.splash\_sound == False:

                pygame.mixer.Sound.**play**(start\_sfx)

                self.splash\_sound = True

        else:

            # Play launch sound

            if self.entry\_sound == False:

                pygame.mixer.music.**play**(-1)

                # pygame.mixer.Sound.play(launch\_sfx)

                self.entry\_sound = True

            keys = pygame.key.**get\_pressed**()

            # Starts the game and goes to the Play Screen

            if keys[pygame.K\_RETURN]:

                global current\_screen

                current\_screen = 1

                pygame.mixer.Sound.**play**(launch\_sfx)

            # If you press esc it quits the game

            elif keys[pygame.K\_ESCAPE]:

                pygame.**quit**()

                sys.**exit**()

    def **render**(self):

        # Render Splash Screen

        if self.splash\_timer < 2:

            self.**splash\_screen**()

        # Render Start Screen

        else:

            screen.**fill**(WHITE)

            screen.**blit**(background\_tex, (0, 0))

            screen.**blit**(mountains\_tex, (0, 284))

            for i in range(8):

                screen.**blit**(grass\_tex, (i \* 128, 524))

            self.**floating\_text**(self.space\_black, -140, 10)

            self.**floating\_text**(self.space\_white, -150)

            self.**floating\_text**(self.dodge\_black, -10, 10)

            self.**floating\_text**(self.dodge\_white, -20)

            self.**plain\_text**(self.startMessage\_black, 150, 6)

            self.**plain\_text**(self.startMessage\_white, 153)

    def **splash\_screen**(self):

        screen.**fill**(WHITE)

        screen.**blit**(background\_tex, (0, 0))

        self.**plain\_text**(self.splashMessage\_black, 0, 6)

        self.**plain\_text**(self.splashMessage\_white)

    def **floating\_text**(self, text, height=0, amount=0):

        screen.**blit**(

            text,

            (

                WIDTH / 2 - text.get\_width() / 2 + amount,

                HEIGHT / 2

                - text.get\_height() / 2

                + math.**sin**(pygame.time.**get\_ticks**() \* 0.005) \* 10

                + height,

            ),

        )

    def **plain\_text**(self, text, height=0, amount=0):

        screen.**blit**(

            text,

            (

                WIDTH / 2 - text.get\_width() / 2 + amount,

                HEIGHT / 2 + height - text.get\_height() / 2 + amount,

            ),

        )

## Codul sursă al paginii ‘sprite\_sheet.py’

import pygame

class SpriteSheet:

    def **\_\_init\_\_**(self, filename: str):

        self.filename = filename

        self.sheet = pygame.image.**load**(self.filename).**convert\_alpha**()

    def **get\_image**(self, frame, width, height, scale, color):

        image = pygame.Surface((width, height))

        image.**set\_colorkey**(color)

        image.**blit**(self.sheet, (0, 0), ((frame \* width), 0, width, height))

        image = pygame.transform.**scale**(image, (width \* scale, height \* scale))

        return image

# Concluzii

În concluzie, vreau să zic că m-am distrat mult făcând acest cod în Python, și de altfel am învățat multe. Este primul meu proiect relativ mare în Python și am intrat în el fără experiență prea multă. Chiar am realizat un vis de-al meu din copilărie, acela de a recrea jocul care m-a adus în lumea calculatoarelor.

Pot să zic, codul nu în totalitate ce voiam. Dacă mai aveam timp implementam un meniu unde poți să-ți alegi skin-uri și unde poți să setezi volumul, etc. Dar cred că mai important, încercam să îi dau o însemnătate jocului. Cum aș face asta? Simplu. Retin scorul și afișez high score-ul ca să ai de ce să continui să joci. Aș fi făcut ca la un moment dat asteroizi să cadă mai repede sau mai încet. Aș fi dat power-up-uri la player ca să aibă scor 2x sau să se miște asteroizi mai greu, etc. Sunt multe de îmbunătățit, dar totuși jocul este în regulă și așa.

Nu este cel mai fain joc, dar este făcut în totalitate de mine. Nu am luat cod de pe internet, doar m-am uitat pe documentația de la Pygame. Am vrut ca jocul asta să fie o provocare pentru mine.

Recomand tuturor oamenilor interesați de programare să învețe Python, pentru că este una dintre cele mai căutate limbi de programare în domeniul informaticii, mai ales în domeniul analizei datelor. Este un limbaj ușor, cu o sintaxă extraordinară, foarte citibil și perfect pentru începători.

# Bibliografie

## Youtube videos:

* [Youtube building a game in pygame](https://www.youtube.com/watch?v=6gLeplbqtqg&t=5757s&pp=ygULZ2FtZSBweXRob24%3D&ab_channel=freeCodeCamp.org)
* [Youtube tutorial pygame](https://www.youtube.com/watch?v=FfWpgLFMI7w&t=293s&pp=ygULZ2FtZSBweXRob24%3D&ab_channel=freeCodeCamp.org)
* [Youtube Python Competion](https://www.youtube.com/watch?v=PC_pAgJopIA&ab_channel=PolyMars)

## Websites:

* [Pygame Documentation](https://www.pygame.org/docs/)
* [GitHub](https://github.com/)
* [StackOverflow](https://stackoverflow.com/)
* [ChatGpt](https://chat.openai.com/)