

Școala  
Colegiul Național “Elena Ghiba Birta”

LUCRARE DE ATESTAT  
DISCIPLINA INFORMATICĂ

Candidat,  
Grozav Maya-Medeea

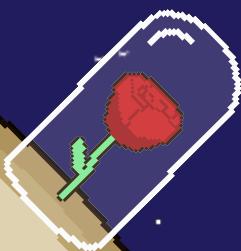
Coordonator,  
Prof. Măgureanu Marieta

# The Little game



# Cuprins

<u>Introducere</u>	4
<u>Logica jocului</u>	5
<u>Variabile. Constante</u>	6
<u>Afișarea și controlarea caracterului</u>	7
<u>Funcții</u>	8
<u>Elemente de design</u>	10
<u>Concluzii</u>	11
<u>Webografie</u>	12



# Introducere

## De ce am ales această temă?

Nu am mai creat un joc până acum, dar toată viața mi-au plăcut provocările din care pot învăța ceva, astfel, am ales să fac un joc în Pygame pentru a învăța un limbaj nou de programare într-un mod distractiv.

Am ales să fac un joc bazat pe cartea *Micul prinț* de Antoine de Saint-Exupéry deoarece este una dintre cărțile mele preferate, având ca mesaj principal importanța iubirii, a conexiunii și a capacitatei de a vedea dincolo de suprafață pentru a aprecia adevărata esență a lucrurilor. Cu ajutorul jocului îmi pot împărtăși entuziasmul pentru carte într-un mod distractiv și ușor de înțeles, sperând să stârnească o curiozitate în jucători.

## Mediul de lucru

### Hardware

Procesor: 1,5 GHz sau mai rapid.

Memorie: 4 GB (4.096 MB) RAM.

Spațiu HDD: 3 GB.

### Software

OS: Windows 7 sau o versiune ulterioară pe 64 de biți

Python 3, Pygame

Programul este realizat cu ajutorul mediului integrat de dezvoltare Visual Studio Code

Programul este împărțit în 4 fișiere: *ATESTAT.py*, *functions.py*, *constants.py* și *pics.py*.

**ATESTAT.py** - codul principal al jocului, care este rulat pentru a-l porni

**functions.py** - funcțiile create de mine pentru joc (ex. funcția pentru afișarea replicilor)

**constants.py** - constantele folosite (ex. lățimea și înălțimea ecranului, numărul de cadre pe secundă, viteza caracterului, etc.)

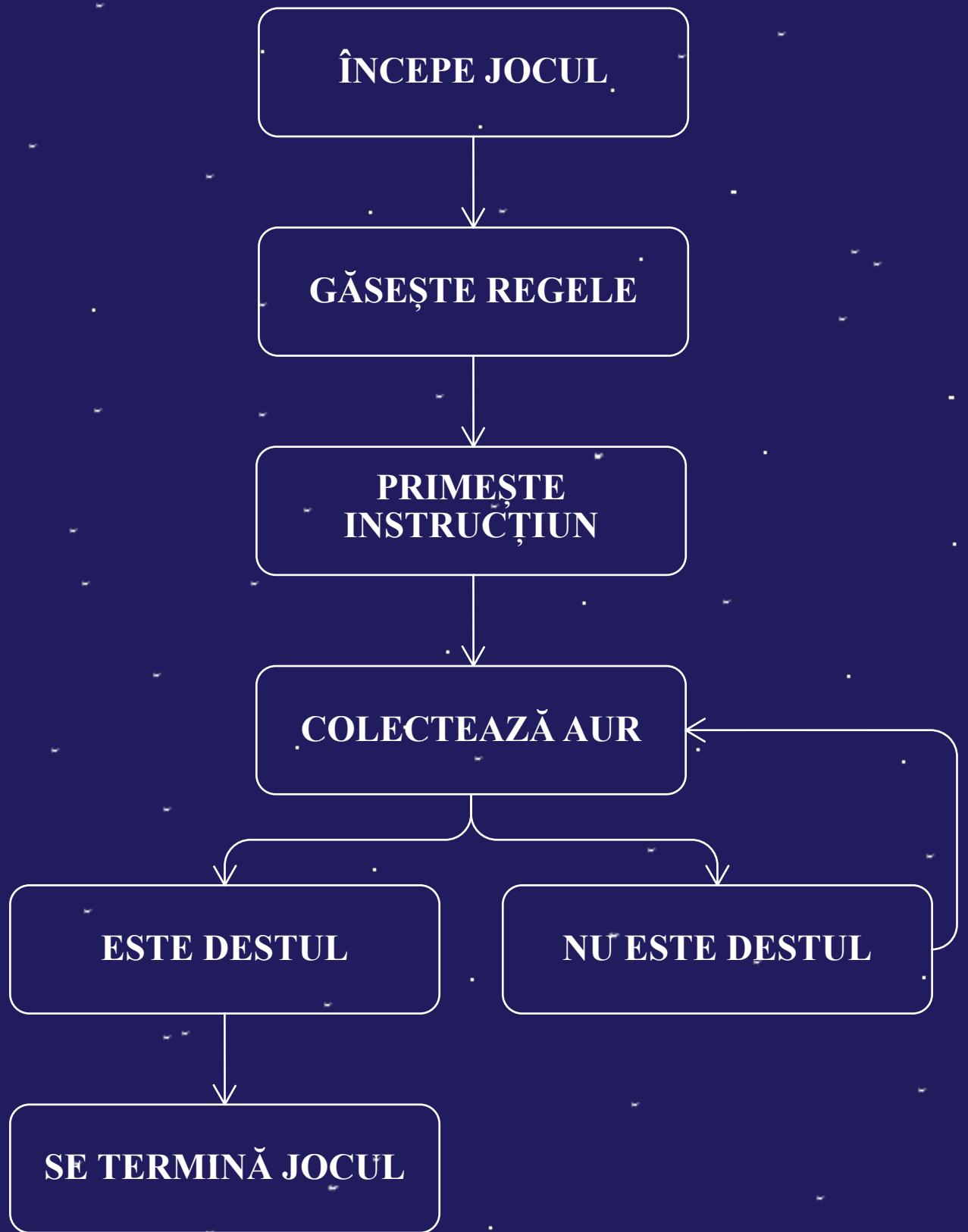
**pics.py** - inițializiază imaginile folosite

Pe lângă aceste patru fișiere mai există și folderul *assets/*, unde se află imaginile și fonturile folosite

## Python/Pygame

Python este un limbaj de programare clar, flexibil și puternic, care este cunoscut pentru multitudinea de moduri în care poate fi folosit. Pygame este o bibliotecă Python specializată în dezvoltarea de jocuri și aplicații multimedia 2D. Cu o sintaxă simplă și o largă varietate de funcții, este modul perfect de a începe programarea jocurilor.

# Logica jocului



# Variabile

La începutul fișierului se initializează variabile globale, fonturi, evenimente de tip timer folosite pentru animații de care va fi nevoie pe tot parcursul programului. Tot jocul are loc într-un while loop care rulează până la închiderea jocului. În acest loop se rulează diferite instrucțiuni în funcție de pagina curentă. La începutul instrucțiunilor fiecărei pagini se initializează, o singură dată, toate variabilele și obiectele necesare pentru pagina respectivă, după care se parcurge codul propriu-zis.

**level** int. - reține numărul paginii curente, în funcție de care se rulează codul principal (0 - pagina de început, 1 - jocul propriu-zis, 1.5 - meniul de sfârșit)

**initialized** boolean - reține dacă au fost sau nu initialize variabilele necesare paginii curente (False - la rularea unei pagini noi -> se initializează variabilele și obiectele necesare -> devine True; True - nu mai initializează variabilele pentru a nu pierde valorile deja modificate)

**screen** pygame.Surface - ecranul pe care rulează jocul

**event** Event - reține un eveniment la întâmplarea acestuia (ex. timer, apăsarea unei taste) pentru a rula instrucțiunile dorite

**main\_lines\_done** list - o listă de buleene care reține dacă au fost sau nu afișate instrucțiunile principale; doar după ce toate valorile din listă sunt adevărate apar itemele care trebuie ulterior colectate

**do\_i\_display\_find\_king** boolean - adevărată pentru primele 5 secunde de când începe jocul, cât timp se afișează instrucțiunea de a găsi regele

**rows** list - lista în care vor fi împărțite replicile pe linii pentru a putea fi afișate

**letters** list - lista în care vor fi puse literele care trebuie afișate; cu ajutorul acestei variabile se va crea animația replicilor

**CHANGE500** Event - eveniment de tip timer care devine adevărat la fiecare 500 de milisecunde; folosită pentru unele animații

**birds\_rect** rect - reține dimensiunea și poziția imaginii păsărilor; dimensiunea rămâne constantă de-a lungul programului, însă poziția se schimbă cu ajutorul funcției *change\_animation\_coords()* pentru a crea animația de mișcare a păsărilor

# Constante

**starting\_point** int. - extrema jocului în partea stângă; nu există nimic dincolo de aceste coordonate

**item\_starting\_point** int. - coordonatele de unde încep să fie plasate itemele

**distance** int. - distanța minimă între două iteme; la această valoare se adaugă numădul furnizat de funcția *random.randint()*

**how\_much\_does\_a\_king\_need** int. - numărul de iteme care trebuie duse la rege pentru a termina jocul

# Afișarea și controlarea caracterului

## Controale:

**W/SPACE/** - sărit; poate fi acționat doar când caracterul este pe pământ

**A/** - deplasare spre stânga

**D/** - deplasare spre dreapta

Gama largă de taste care pot fi folosite pentru a controla caracterul este menită să creeze o ușoară utilizare a programului, în funcție de deprinderile jucătorului (</^> pentru oamenii mai puțin experimentați în domeniul jocurilor video; A/W/D/SPACE pentru oamenii cu mai multă experiență în jocuri video)

Caracterul controlat de jucător este static, având poziția fixă la mijlocul ecranului. Impresia de mișcare este dată de mutarea fundalului, pământului și restul obiectelor spre dreapta la apăsarea tastei '<', respectiv spre stânga, la apăsarea tastei '>'.

Pentru afișarea corectă a direcției și animațiilor personajului am folosit mai multe variabile:

**walking** - adeverată în timp ce se acționează butoanele de direcție

**direction** - 0 dacă este apăsată tasta '<', 1 dacă este apăsată tasta '>'

**on\_ground** - 0 în cazul în care caracterul sare, 1 în caz contrar

**char\_image** - variază între 0 și 1, valoarea schimbându-se la fiecare 500 de milisecunde, creând o animație de mișcare

Cu ajutorul acestor variabile, se realizează afișarea caracterului. Spre exemplu, în cazul în care caracterul este pe pământ, una dintre tastele '</'>' este apăsată și caracterul 'merge' spre dreapta, atunci se afișează imaginea cu caracterul îndreptat spre dreapta, prima sau a doua, în funcție de valoarea reținută în *char\_image*.



char\_image = 0  
walking = 1  
direction = 0  
on\_ground = 1



char\_image = 1  
walking = 1  
direction = 0  
on\_ground = 1



direction = 0  
on\_ground = 0



walking = 0  
direction = 1  
on\_ground = 1



char\_image = 0  
walking = 1  
direction = 1  
on\_ground = 1

# Functii

**find\_x** - este apelată la apăsarea tastei ‘p’; cauță prin toate itemele, returnează id-ul itemului care se află la mijlocul ecranului ( $\pm 50\text{px}$ ), sau -1 dacă nici un item nu se află la mijlocul ecranului

**pick\_up** - apelată dacă valoarea returnată de *find\_x* este diferită de -1; ridică item-ul de pe jos, îl adaugă în inventar

```
def give(to_give, to_get, max):
    while to_give and (to_get < max):
        to_get += 1
        to_give -= 1
    return to_get, to_get
```

**give** - mută obiecte din inventarul jucătorului în pseudo-inventarul regelui, până când inventarul jucătorului este gol sau inventarul regelui este plin

**convert\_into\_rows** - desparte un text în rânduri pentru a putea fi afișat (Pygame nu are funcții de afișare a textului pe mai multe rânduri); folosește funcția *convert\_into\_rows\_beta*

**convert\_into\_rows\_beta** - plasează litere pe rândul curent până când ajunge la numărul maxim de caractere; în cazul în care după ultimul caracter de pe rând urmează, în replica inițială, altceva decât spațiu (nu are loc tot cuvântul pe rând) se sterg caractere de la sfârșit către început până când se găsește spațiu, iar cuvântul șters va fi plasat pe rândul următor

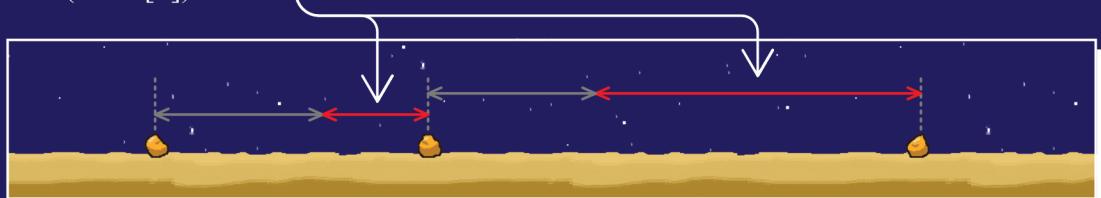
**what\_letters\_do\_i\_display** - calculează numărul de litere care trebuie afișate într-un moment dat și le pune, pe rânduri, într-o listă; este esențială în procesul de creare a animațiilor replicilor

```
def what_letters_do_i_display(main_k, chars_on_row, rows, max):
    x = []
    k = 0
    isit = 0
    poz = 0
    while k < main_k and poz < len(rows):
        if k + chars_on_row[poz] >= main_k:
            x.append(rows[poz][:main_k - k+1])
        else:
            x.append(rows[poz][:chars_on_row[poz]+1])
        k += chars_on_row[poz]
        poz += 1
    if main_k == max:
        isit = 1
    return x, isit
```

**talk** - afișează pe ecran chenarul, fața regelui și, pe rând, literele replicii

**manage\_items** - afișează itemele; totodată afișează și instrucțiunea de a apăsa tasta ‘P’ pentru a le ridica la primele două

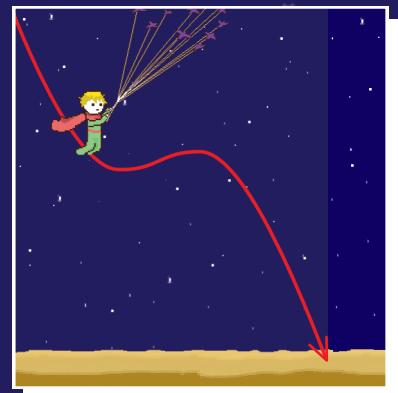
**initialize\_items** - inițializează itemele; variabila *items* reține: coordonate (*items[0]*), generate cu ajutorul *random.randint()*, imaginea item-ului(*items[1]*) și dacă este sau nu pe pământ(*items[2]*)



**change\_animation\_coords** - crează traectoria player-ului în animația de început:

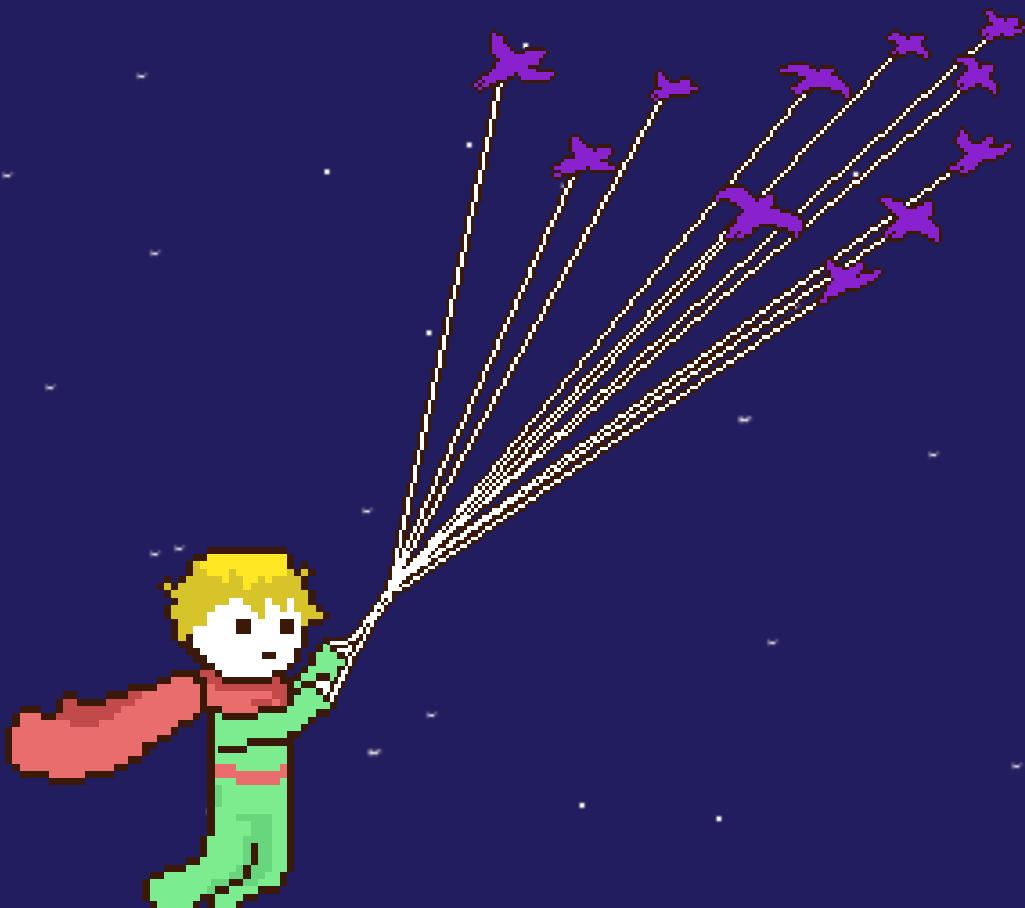
```
rect1.x += 1  
rect1.y = rect1.x + 100 * math.sin(rect1.x / 72)
```

unde *rect1* reprezintă poziția păsărilor, iar caracterul este dispus în funcție de aceasta; la fiecare secundă abscisa crește cu 1 și ordonata devine  $x + 100 * \sin(x / 72)$



**button click** - verifică dacă valorile coordonatelor click-ului se află în interiorul suprafeței butonului

**hm\_chars\_on\_row** - calculează numărul de caractere care se află pe un rând



# Elemente de design

Toate imaginile folosite în program sunt o creație personală. Inspiratia a fost luată din jocurile din anii '80, când prospera stilul *pixelart*. Astfel se poate resimți o ușoară nostalgie de cei ce s-au jucat jocuri video în copilărie.

Design-ul personajelor este în mare parte bazat pe design-ul personajelor aşa cum apar în carte, creat de însuși autorul cărții.

**Micul prinț** apare de-a lungul ilustrațiilor cărții cu același nume în mai multe înfățișări: de la haine simple cu papion roșu sau cu fular galben, la un costum de prinț cu sabie. Am ales în defavoarea înfățișării din urmă din cauza complexității costumului și inutilității unei săbi în acest joc. Astfel, am ales să-l reprezin în haine simple și fular (pentru a crea mai multă mișcare în animația de umblat), roșu în loc de galben (pentru un contrast mai mare al culorilor).



**Regele** are un design fidel celui original, fiind doar ușor simplificat. Din moment ce regele are replici, a fost nevoie și de imaginea capului său, aceasta fiind mai mare și mai detaliată. Culoarea pielii regelui este aceeași culoare cu cea a bucătelelor de aur, reprezentând măreția acestuia. Prima replică („Ah! Here is a subject!”) este luată din carte, fiind primul lucru pe care îl spune regele micului prinț. Restul repeticilor au un caracter de finalitate, reflectând personalitatea regelui aşa cum apare în opera originală.



**Aurul** nu apare în carte. Ideea a fost preluată din folclorul românesc, în care nu există rege care să nu dețină aur. Astfel adunatul și adusul de aur regelui este misiunea perfectă: nici prea complexă, nici complet liniară. Pentru un design mai bun și o complexitate puțin mai ridicată a codului nu există o singură imagine a aurului, ci trei. Acestea sunt însușite itemelor ciclic.



# Concluzii

Dacă ar fi să existe versiunea 2.0 a jocului, ar putea fi adăugate încă 6 nivele, fiecare reprezentând una dintre cele 7 planete pe care micul prinț le vizitează în carte. Cu fiecare nivel ar deveni tot mai grele misiunile, toate fiind antrenament pentru ultimul nivel. Un alt lucru pe care l-aș adăuga în versiunea 2.0 ar fi muzica și efectele sonore. În versiunea 3.0 aş adăuga mai multe design-uri pentru caracter, astfel încât jucătorul să-și poată personaliza caracterul după propriul plac.

În concluzie, prin începerea acestui proiect am deschis poarta către o lume nouă pe care nu o știam, dar am ajuns să o înțeleg în doar câteva săptămâni. Pe lângă faptul că am reușit să învăț bazele limbajului de programare *Python*, modul în care funcționează *Pygame* și elemente de bază pentru programarea jocurilor, mi-am și dezvoltat modul de gândire prin crearea unor funcții complexe și foarte diferite de ceea ce știam până acum.

## **Webografie**

**pixilart.com**

**geeksforgeeks.org**

**pygame.org**

**w3schools.com**

**1001fonts.com**