**Flappy Bird Clone - Python**

Acesta este un proiect Flappy Bird clonat, dezvoltat în Python folosind biblioteca Pygame. Scopul acestui joc este de a ghida o pasăre printr-o serie de obstacole fără a le atinge.

Cum se Rulează Jocul

Pentru a rula jocul, asigurați-vă că aveți Python și Pygame instalate. Apoi, executați fișierul main.py pentru a începe jocul.

Structura Codului

Jocul este structurat în mai multe părți principale:

Inițializarea Jocului

* pygame.init(): Inițializează modulele necesare Pygame.
* clock = pygame.time.Clock(): Crează un ceas pentru a controla rata de cadre pe secundă.
* screen = pygame.display.set\_mode((screen\_width, screen\_height)): Setează dimensiunea ferestrei de joc.
* pygame.display.set\_caption(): Setează titlul ferestrei jocului.

Variabilele Jocului

* Aici sunt setate variabilele de bază ale jocului, cum ar fi viteza de defilare a solului, starea jocului, decalajul între țevi, frecvența țevilor, etc.

Încărcarea Imaginilor

* Încarcă imaginile de fundal, sol și butonul de restart.

Funcția draw\_text

* Este o funcție pentru a afișa text pe ecran.

Funcția reset\_game

* Resetează jocul la starea inițială după ce jucătorul pierde.

Definirea Claselor Principale

1. Clasa Bird

* Rolul clasei: Reprezintă pasărea controlată de jucător în joc.
* Atribute și metode principale:
* Constructor \_\_init\_\_(self, x, y): Inițializează pasărea. Stabilește poziția inițială, încarcă imaginile pentru animația de zbor și inițializează variabilele necesare pentru mișcare și stare.
* self.images: O listă de imagini utilizate pentru animația pasării.
* self.index: Indexul curent pentru imaginea pasării din animație.
* self.counter: Un contor pentru a controla viteza de schimbare a imaginilor în animație.
* self.rect: Un dreptunghi Pygame care reprezintă poziția și coliziunea pasării.
* self.vel: Viteza de deplasare verticală a pasării.
* self.clicked: Un flag pentru a verifica dacă jucătorul a apăsat click sau nu.
* Metoda update():

1. Gestionarea Gravitației:
   1. Când pasărea zboară (flying == True), viteza verticală (self.vel) este incrementată pentru a simula gravitația. Acest lucru face ca pasărea să cadă în jos pe ecran.
   2. Viteza este limitată pentru a preveni ca pasărea să cadă prea repede.
2. Actualizarea Poziției Verticale:
   1. Poziția verticală a pasării (self.rect.y) este actualizată în funcție de viteza actuală, mutând pasărea în jos pe ecran.
3. Gestionarea Săriturii:
   1. Dacă jucătorul apasă pe mouse și pasărea nu a "sărit" deja (self.clicked == False), viteza verticală este setată la o valoare negativă, provocând pasărea să se deplaseze în sus pe ecran.
4. Actualizarea Animației de Zbor:
   1. Animația pasării este controlată prin schimbarea imaginilor. Indexul imaginii (self.index) este actualizat periodic pentru a cicla prin imaginile de animație.
5. Rotația Pasării:
   1. Imaginea pasării este rotită în funcție de viteza ei verticală, dând impresia că pasărea se înclină în sus sau în jos pe măsură ce zboară sau cade.

2. Clasa Pipe

* Rolul clasei: Reprezintă țevile (obstacolele) prin care pasărea trebuie să treacă.
* Atribute și metode principale:
* Constructor \_\_init\_\_(self, x, y, position): Inițializează țeava. Stabilește poziția inițială și încarcă imaginea țevii. Poziția poate fi de sus sau de jos, determinând orientarea țevii.
* self.image: Imaginea țevii.
* self.rect: Un dreptunghi Pygame care reprezintă poziția și coliziunea țevii.
* Metoda update():
  1. Mișcarea Țevii:
     + Poziția orizontală a țevii (self.rect.x) este redusă cu viteza de defilare a solului (scroll\_speed). Acest lucru face ca țeava să se miște spre stânga ecranului, creând iluzia că pasărea se mișcă înainte.
  2. Verificarea și Eliminarea Țevii:
     + Dacă țeava a ieșit din ecran (adică self.rect.right < 0), aceasta este eliminată din grupul de sprite-uri (self.kill()), optimizând astfel performanța jocului prin eliminarea obiectelor care nu mai sunt necesare.

3. Clasa Button

* Rolul clasei: Reprezintă un buton interactiv, cum ar fi butonul de restart folosit la sfârșitul jocului.
* Atribute și metode principale:
  + Constructor \_\_init\_\_(self, x, y, image): Inițializează butonul. Setează poziția butonului pe ecran și încarcă imaginea acestuia.
  + self.image: Imaginea butonului.
  + self.rect: Un dreptunghi Pygame care reprezintă poziția și coliziunea butonului.
  + Metoda draw(): Desenează butonul pe ecran și verifică interacțiunile cu mouse-ul. Dacă mouse-ul este peste buton și este apăsat, metoda returnează True, indicând că butonul a fost apăsat.

Funcția jump

* Verifică Starea Jocului: Înainte de a permite săritura, funcția verifică dacă jocul nu este terminat (game\_over == False) și dacă pasărea nu a început deja să zboare (flying == False). Dacă pasărea nu zboară, atunci setează flying = True, ceea ce va activa logica de zbor în metoda update a pasării.
* Inițiază Săritura: Dacă jocul nu este terminat, viteza verticală a pasării (flappy.vel) este setată la -10. Aceasta este o valoare negativă, ceea ce înseamnă că pasărea se va deplasa în sus pe ecran, contrar forței de gravitație, creând efectul de săritură.

Ce se Întâmplă în Bucla while

* Bucla while run: este inima jocului, unde se desfășoară toată logica de joc și actualizarea ecranului. Iată principalele acțiuni:
* Controlul Ratei de Cadre: clock.tick(fps) asigură că jocul rulează la o rată constantă de cadre pe secundă.
* Afișarea Fundalului și a Solului: Fundalul și solul sunt desenate pe ecran la fiecare iterație a buclei.
* Desenarea și Actualizarea Grupurilor de Sprite-uri: bird\_group și pipe\_group sunt desenate și actualizate. Aceasta include gestionarea animației pasării, mișcarea țevilor, etc.
* Logica Scorului: Verifică dacă pasărea a trecut prin țevi fără a se ciocni de ele și actualizează scorul în consecință.
* Detectarea Coliziunilor: Verifică dacă există coliziuni între pasăre și țevi sau dacă pasărea a ieșit de pe ecran.
* Generarea Țevilor: Dacă s-a împlinit timpul necesar, se generează un nou set de țevi.
* Mișcarea Solului: Solul se mișcă continuu pentru a crea efectul că pasărea se deplasează prin joc.
* Resetarea Jocului: Dacă jocul s-a terminat, se afișează butonul de restart. Apăsând butonul resetează jocul pentru a începe o nouă încercare.
* Gestionarea Evenimentelor: Buclea verifică și răspunde la evenimentele de intrare, cum ar fi apăsările pe butoanele mouse-ului sau pe tastatură. Aici este apelată funcția jump() atunci când se detectează un eveniment de click sau o apăsare a tastei spațiu.
* Actualizarea Ecranului: După ce toate elementele au fost desenate și actualizate, ecranul este reîmprospătat pentru a reflecta modificările.
* Închiderea Jocului: Dacă se primește evenimentul de închidere a ferestrei (de exemplu, utilizatorul apasă pe butonul de închidere al ferestrei jocului), run devine False, ceea ce încheie bucla while și în final oprește jocul prin apelarea pygame.quit().