Lecția 10 - Python Games

Trainer: Hrișcă Miruna

Aplicații folosite + alternative

01 PyCharm
Link - windows
Link - Apple

02 One Compiler
Link

03 VS Code
Link

04 Online GDB
Link

Biblioteca Turtle

1. `onscreenclick(funcție)`
   - Această funcție spune calculatorului:

     „Când cineva face click pe ecran, apelează această funcție!”

   - Este ca și cum ai spune: „Dacă apeși pe ecran, broasca țestoasă va reacționa.”
2. `hideturtle()`
   - Face broasca țestoasă **invizibilă**, astfel încât **vedem doar desenul**, nu și broasca.
   - Imaginează-ți că broasca țestoasă este un **pix invizibil** care desenează fără să o vedem.
3. `tracer(False)`
   - Această funcție **oprește animația** desenului, ca să nu mai vedem broasca țestoasă mișcându-se pas cu pas.
   - Astfel, desenul apare **instantaneu**, mult mai repede.

Biblioteca FreeGames

1. `floor`
   - Funcția `floor` ne dă **cel mai mare număr întreg mai mic sau egal** cu un număr dat.
   - Gândește-te că ai o scară și te afli pe treapta 3.7 → `floor(3.7)` te duce **înapoi pe treapta 3**, adică **rotunjește în jos**.
2. `square`
   - Funcția `square` desenează **pătrate** pe ecran cu ajutorul broaștei țestoase.
   - Poți spune: „Desenează un pătrat roșu, cu centrul la punctul X,Y și latura de 50 de pixeli.”
3. `vector`
   - Un **vector** este ca un **săgeată care arată direcția și distanța** pe ecran.
   - În FreeGames, vectorii ne ajută să știm **unde se află obiectele**, cum se mișcă și cum să le mutăm.

Biblioteca Time

- Biblioteca **time** este ca un **ceas magic în Python**.
- Ne ajută să **măsurăm timpul** sau să facem lucrurile să se întâmple **după o anumită întârziere**.

1. `sleep()`

- Funcția `sleep()` spune calculatorului:
  „Oprește-te puțin și nu face nimic pentru X secunde.”
- Este ca atunci când zicem: „Așteaptă 3 secunde și apoi continuăm jocul.”

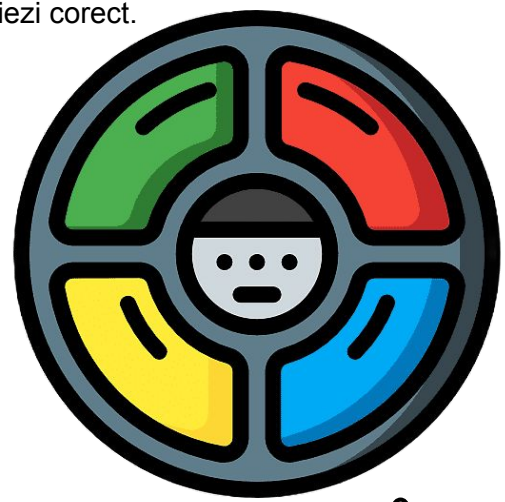
Biblioteca Random

Funcția `choice(seq)`

- Imaginează că ai o **cutie cu bomboane de diferite culori**: roșu, galben, verde și albastru.
- Dacă vrei să alegi **o bomboană la întâmplare**, fără să te uiți, folosești `choice()`.
- Calculatorul alege **o culoare la întâmplare** din listă.

Despre jocul Simon Says

- **Simon Says** este un joc în care trebuie să **urmărești și să repeți o secvență** de culori sau cărămizi.
- Este ca un **joc de memorie**: calculatorul îți arată un model, iar tu trebuie să-l copiezi corect.



Cum funcționează:

1️⃣ **Începem jocul**

- Apăsăm pe ecran ca să pornim.
- Calculatorul va afișa **prima secvență** de culori sau cărămizi.

2️⃣ **Trebuie să urmăm modelul**

- Apoi, apăsăm pe **cărămizi în aceeași ordine** în care le-a arătat calculatorul.

3️⃣ **Modelul crește**

- De fiecare dată când **nimerești secvența corect**, calculatorul adaugă **un pas în plus** la secvență.
- Așa jocul devine **tot mai greu**, iar tu trebuie să-ți folosești memoria și atenția.

Programarea jocului

```
from random import choice
from time import sleep
from turtle import *
from freegames import floor, square, vector
```

- `choice` din `random` – alege aleator un element dintr-o listă.
- `sleep` din `time` – pune pauză în program pentru un anumit număr de secunde.
- `turtle` – folosit pentru desen, animarea grafice simple pe ecran.
- `floor` – rotunjește la cel mai apropiat multiplu al unui număr.
- `square` – desenarea unui pătrat pe ecran.
- `vector` – reprezintă coordonate `(x, y)` pentru poziția fiecărui pătrat.

Programarea jocului

```
pattern = []
guesses = []
tiles = {
    vector(0, 0): ('red', 'dark red'),
    vector(0, -200): ('blue', 'dark blue'),
    vector(-200, 0): ('green', 'dark green'),
    vector(-200, -200): ('yellow', 'khaki'),
}
```

- `pattern` – lista cu secvența de culori pe care trebuie să o memoreze jucătorul.
- `guesses` – lista cu încercările jucătorului, pentru a verifica dacă se potrivesc cu `pattern`.
- `tiles` – dicționar care leagă coordonatele fiecărui pătrat de culorile lui:
  - primul element (`glow`) e culoarea aprinsă când "clipesc" pătratul.
  - al doilea element (`dark`) e culoarea normală.

Programarea jocului

```
def grid():
    """Draw grid of tiles."""
    square(0, 0, 200, 'dark red')
    square(0, -200, 200, 'dark blue')
    square(-200, 0, 200, 'dark green')
    square(-200, -200, 200, 'khaki')
    update()
```

- Desenează cele patru pătrate pe ecran.
- `square(x, y, size, color)` – desenează un pătrat la coordonatele `(x, y)` cu dimensiunea `size` și culoarea `color`.
- `update()` – actualizează ecranul pentru a afișa pătratele.

Programarea jocului

```
def flash(tile):
    """Flash tile in grid."""
    glow, dark = tiles[tile]
    square(tile.x, tile.y, 200, glow)
    update()
    sleep(0.5)
    square(tile.x, tile.y, 200, dark)
    update()
    sleep(0.5)
```

Programarea jocului

- `def flash(tile):` – definim o funcție numită `flash`, care primește **un parametru** numit `tile`.
  - Parametrul `tile` reprezintă pătratul pe care vrem să îl facem să "clipească" pe ecran.
- `tiles[tile]` caută în dicționarul `tiles` perechea de culori pentru pătratul dat.
    a. Exemplu: `tiles[vector(0,0)]` → `('red', 'dark red')`.
  - `glow, dark` – despachetează perechea de culori:
    a. `glow` → culoarea aprinsă, care apare când pătratul „clipeste”.
    b. `dark` → culoarea normală a pătratului.
- `square(tile.x, tile.y, 200, glow)` – desenează un pătrat la coordonatele `(tile.x, tile.y)` cu latura 200 și culoarea `glow`.
  - Practic schimbăm culoarea pătratului la cea aprinsă.
- `update()` – actualizează ecranul pentru a afișa pătratul aprins.
- `sleep(0.5)` – așteaptă jumătate de secundă, ca jucătorul să observe culoarea aprinsă.
- `square(tile.x, tile.y, 200, dark)` – redesenează pătratul cu culoarea lui normală (`dark`).
- `update()` – actualizează ecranul pentru a afișa pătratul stins.
- `sleep(0.5)` – pauză pentru ca schimbarea să fie vizibilă.

Programarea jocului

```
def grow():
    """Grow pattern and flash tiles."""
    tile = choice(list(tiles))
    pattern.append(tile)

    for tile in pattern:
        flash(tile)

    print('Pattern length:', len(pattern))
    guesses.clear()
```

Programarea jocului

- `list(tiles)` – ia toate pătratele disponibile (coordonatele lor) și le pune într-o listă.
- `choice(list(tiles))` – alege **un pătrat la întâmplare** din listă.
- `tile` – pătratul ales, care va fi următorul în secvență.
- `pattern` – lista care stochează secvența completă de culori pe care jucătorul trebuie să o urmeze.
- `append(tile)` – adaugă pătratul ales la sfârșitul secvenței.
- `for tile in pattern: flash(tile)` - Parcurge fiecare pătrat din `pattern`. Apelează funcția `flash(tile)` pentru a face pătratul să „clipească” (să se aprindă și să se stingă). Astfel, jucătorul vede **toată secvența**, inclusiv pătratul nou adăugat.
- `print('Pattern length:', len(pattern))` - `len(pattern)` – calculează câte pătrate are secvența. `print(...)` – afișează pe ecranul consolei lungimea secvenței.
- `guesses` – lista încercărilor jucătorului.
- `clear()` – șterge toate elementele din listă, astfel încât jucătorul să înceapă cu lista goală pentru noua rundă.

Programarea jocului

```
def tap(x, y):
    """Respond to screen tap."""
    onscreenclick(None)
    x = floor(x, 200)
    y = floor(y, 200)
    tile = vector(x, y)
    index = len(guesses)
    if tile != pattern[index]:
        exit()

    guesses.append(tile)
    flash(tile)
    if len(guesses) == len(pattern):
        grow()
    onscreenclick(tap)
```

Programarea jocului

- `def tap(x, y):` – definim funcția `tap`, care **primește coordonatele `x` și `y` ale punctului apăsat pe ecran**.
- `onscreenclick(None)` - Dezactivează temporar click-urile pe ecran, pentru a evita probleme dacă jucătorul apasă foarte repede.
- `floor(x, 200)` – rotunjește coordonata x la cel mai apropiat multiplu de 200.
- `floor(y, 200)` – face același lucru pentru y.
- `vector(x, y)` – creează un obiect vector cu coordonatele ajustate, astfel încât să știm exact pe care pătrat s-a apăsat.
- `index` reprezintă **poziția în secvența pe care jucătorul trebuie să o urmeze**.
- `len(guesses)` ne spune câte încercări corecte a făcut deja jucătorul.
- Exemplu: dacă jucătorul a apăsat deja 2 pătrate corect, următorul trebuie să fie `pattern[2]`
- `if tile != pattern[index]: exit()` - Compară pătratul apăsat (`tile`) cu pătratul corect din secvență (`pattern[index]`). Dacă e greșit, jocul se oprește (`exit()`).
- `guesses.append(tile)` – adaugă pătratul apăsat corect la lista încercărilor.
- `flash(tile)` – aprinde și stinge pătratul apăsat pentru feedback vizual.
- `if len(guesses) == len(pattern): grow()` - Dacă jucătorul a apăsat toate pătratele corect, secvența s-a terminat. Funcția `grow()` va adăuga un pătrat nou în secvență și va arăta întregul pattern.
- `onscreenclick(tap)` - Setează din nou funcția `tap` să răspundă la click-uri. Acum jucătorul poate apăsa următorul pătrat.

Programarea jocului

```
def start(x, y):
    """Start game."""
    grow()
    onscreenclick(tap)
```

- Inițiază jocul.
- Apelează `grow()` pentru prima secvență și setează funcția `tap` ca răspuns la click.

Programarea jocului

```
setup(420, 420, 370, 0)
hideturtle()
tracer(False)
grid()
onscreenclick(start)
done()
```

- `setup(420, 420, 370, 0)` – setează fereastra de joc: lățime, înălțime, poziție.
- `hideturtle()` – ascunde cursorul.
- `tracer(False)` – oprește animația automată, astfel încât să putem controla când se actualizează ecranul.
- `grid()` – desenează tabla de joc.
- `onscreenclick(start)` – începe jocul când se face primul click.
- `done()` – păstrează fereastra deschisă până când utilizatorul o închide.