# Platformer Project

## Haladó programozás beadandó feladat

**Készítette:**

**Kotlár Botond (FUHZ21)**

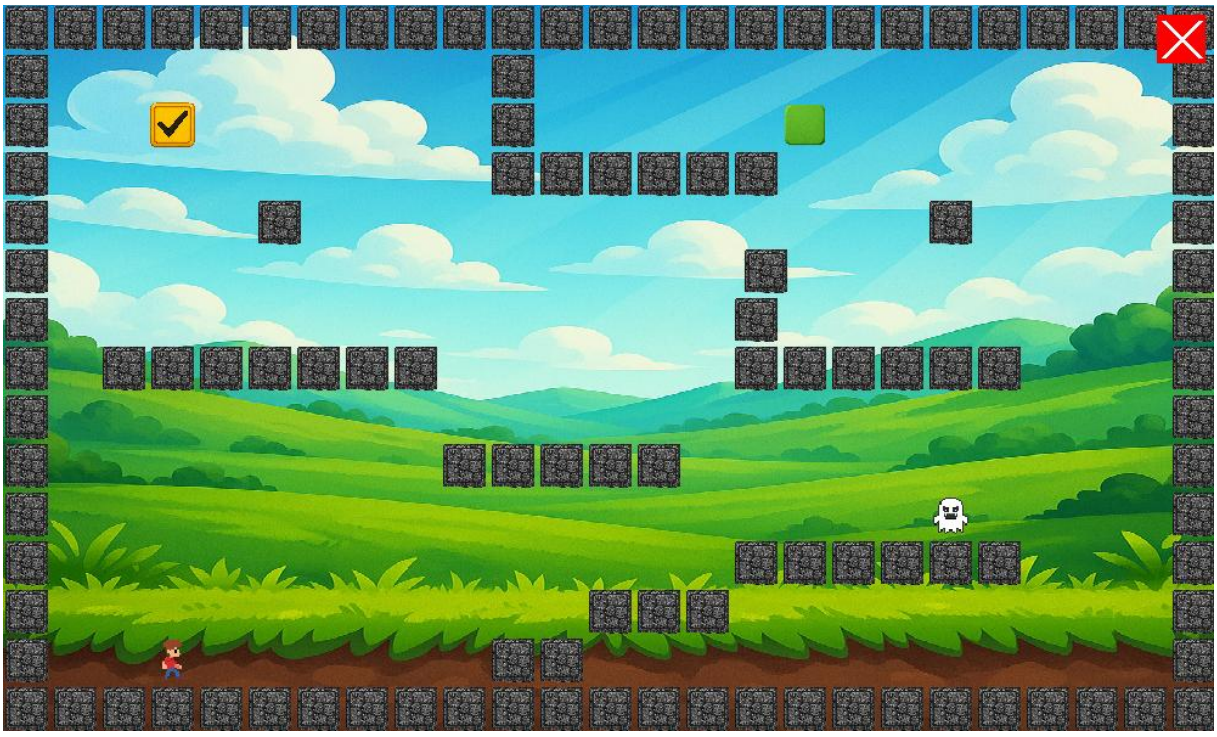
**Dobóczi Levente (JC84WP)**

**Krecsmarik Zalán (B1XTCZ)**

Github link: https://github.com/Zzalann/HaladoProgBeadando

### A feladat ismertetése:

A Platformer Project egy Python és Pygame alapú, 2D-s ügyességi videojáték. A játék célja, hogy a játékos végigjusson három különböző nehézségű pályán, miközben akadályokat kerül el, mozgó platformokat használ,megkerüli az ellenséget, és egyszerű logikai kvízekre válaszol. A projekt keretében megvalósult a menürendszer, a pályaválasztás, a tile-alapú pálya betöltése és a játékmechanikák működése.



## A játék célja:

A játékos feladata, hogy:

- teljesítse az összes pályát,
- elérje a célt jelző blokkot,
- aktiválja a checkpoint kvízeket,
- elkerülje az ellenségeket és a veszélyes elemeket (pl. láva).

A pályák lineárisan haladnak: a következő pálya csak akkor érhető el, ha az előzőt teljesítette a játékos.

## Felhasználói környezet:

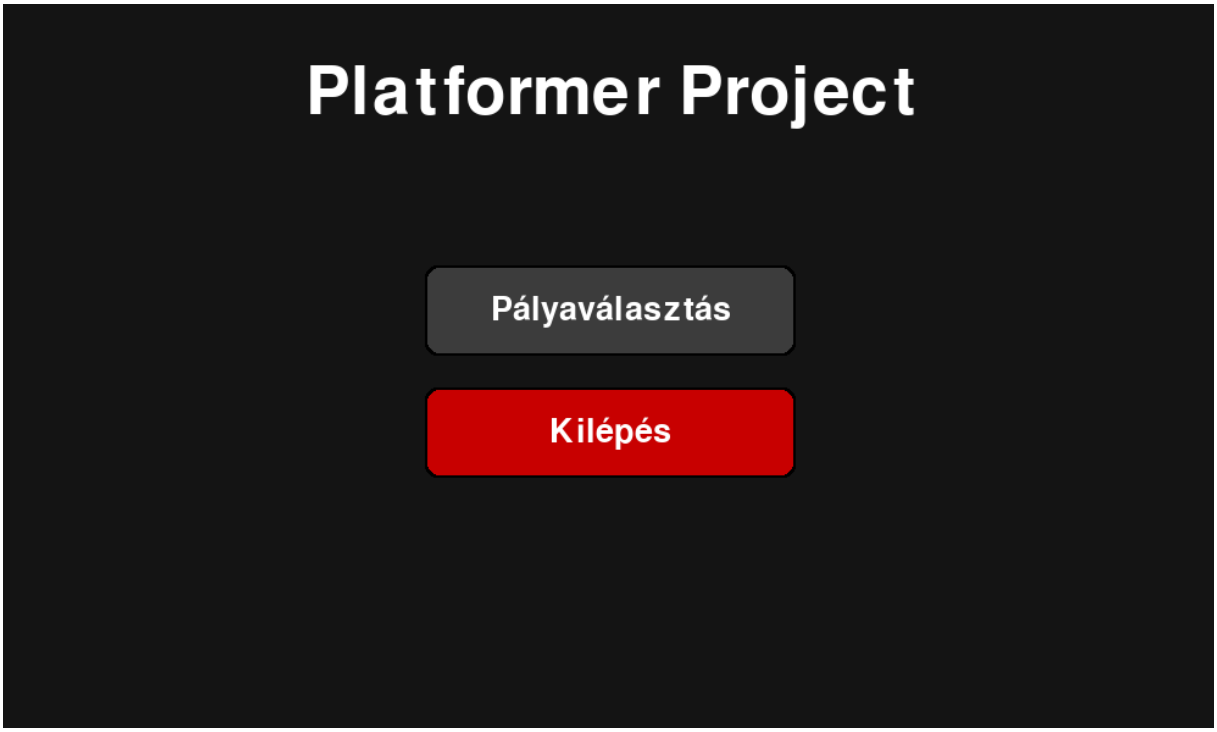
- Nyelv: Python 3.x
- Grafikus könyvtár: Pygame
- Felbontás: 1000 × 600 px
- Input eszközök: billentyűzet (nyilak,space), egér (menü)

## A játék felépítése:

**Főmenü**

- A játék induláskor két gombot mutat:

1. Pályaválasztás
2. Kilépés
    - A gombok interaktívak, egérrel navigálhatók, hover-effektet használnak.



**Pályaválasztó menü**

- Három pálya található a játékban:
    1. pálya: mindig elérhető
    2. pálya: csak az 1. teljesítése után
    3. pálya: csak a 2. teljesítése után
- A menü figyelmezteti a játékost, ha túl korai pályát választana.

## Pályarendszer:

A pályák .txt fájlokban tárolódnak (level1.txt, level2.txt, level3.txt).
Minden pálya mérete: 25 × 15 tile → 1000 × 600 px.

Tile-jelek és jelentésük:

| Karakter | Jelentés |
|---|---|
| # | Fal / platform |
| P | Player kezdőpozíció |
| G | Célpont |
| E | Ellenség pozíció |
| C | Checkpoint |
| L | Láva |
| M | Mozgó platform |
| . | Üres terület |

A program ezeket soronként beolvassa, létrehozza a megfelelő rect objektumokat, majd kirajzolja a sprite-okat.

```
#########################
#.........#.............#
#..C......#.....G.......#
#.........######........#
#....M.............#....#
#...............M.......#
#...............#.......#
#.#######......#######...#
#.......................#
#........#####..........#
#...................E....#
#...............#######...#
#..P........###.........#
#..........##............#
#########################
```

```
#########################
#.......................#
#.............#........#
#.............#........#
#.....M.........#....G...#
#..........###############
#.......................#
#....E..................#
#..############.........#
#......................C.#
#............M......####.#
#..............#.........#
#..............#.....P...#
#..............#.........#
#########################
```

```
#########################
#...........#...........#
#........C..#..G........#
#...M...##########.......#
#...........#...........#
#...........#...........#
######......#...#####...#
#...........#...........#
#.......########....M....#
#...........#...........#
######......#...........#
#...........#..#######...#
#..P....######..........#
#.......................E.#
#########################
```

## Játékmechanikák:

### Player vezérlés

- Balra / jobbra: ← és → gombok
- Ugrás: SPACE
- Valósághű gravitáció
- Pixelpontos ütközés detektálás blokkokkal és mozgó platformokkal

A mozgó platformok akkor is mozgatják a játékost, ha rajtuk áll.

### Ellenség AI

Az ellenség három állapotot használ:

- idle: alaphelyzet
- chase: ha a játékos 200 px-en belülre kerül
- return: visszasétál az eredeti pozíciójára

Ha a player hozzáér → halál.

### Checkpoint és kvíz

A játékos a checkpoint mezőre lépve:

1. kap egy véletlenszerű kvízkérdést
2. helyes válasz → checkpoint aktiválódik
3. rossz válasz → a játékos meghal
4. feladás → visszalép a menübe

Csak aktivált checkpoint után érhető el a cél és nyerhető meg a pálya.

**Halál**

Halál oka lehet:

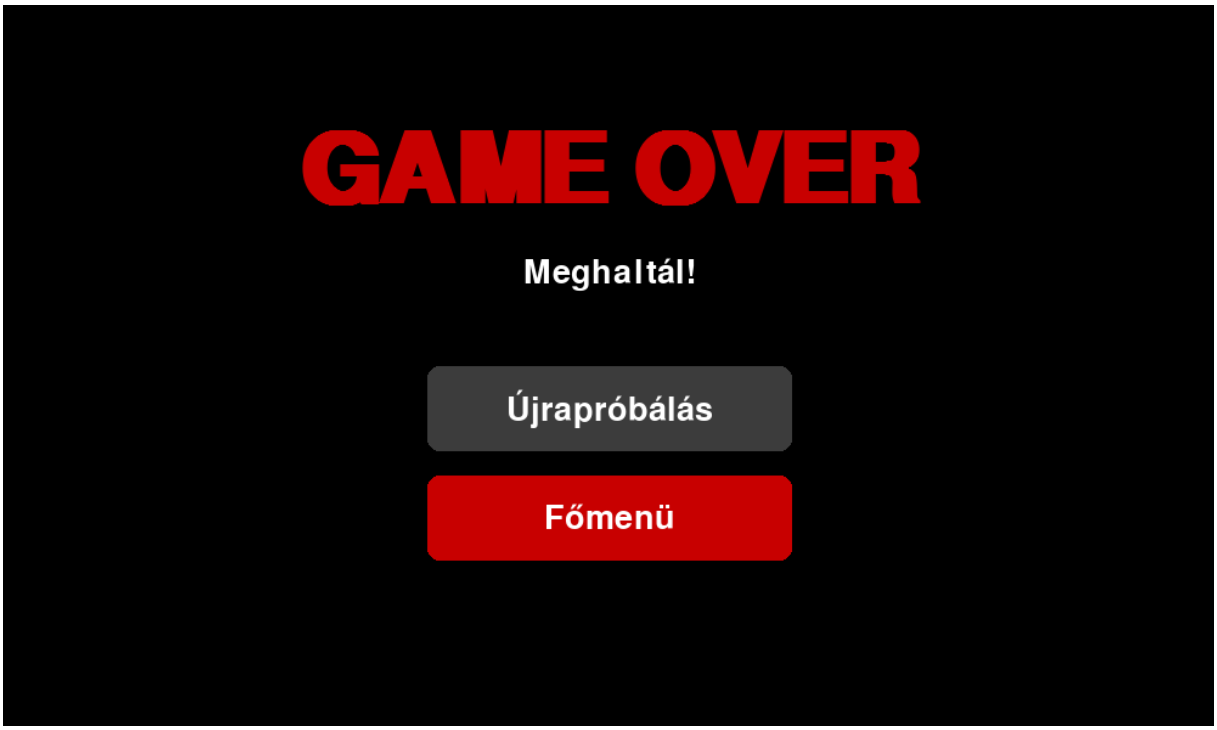
- ellenség érintése
- rossz válasz a kvíz során / feladás

Halál után a GAME OVER képernyő jelenik meg.

**Game Over képernyő**

Lehetőségek:

- Újrapróbálás
- Vissza a menübe



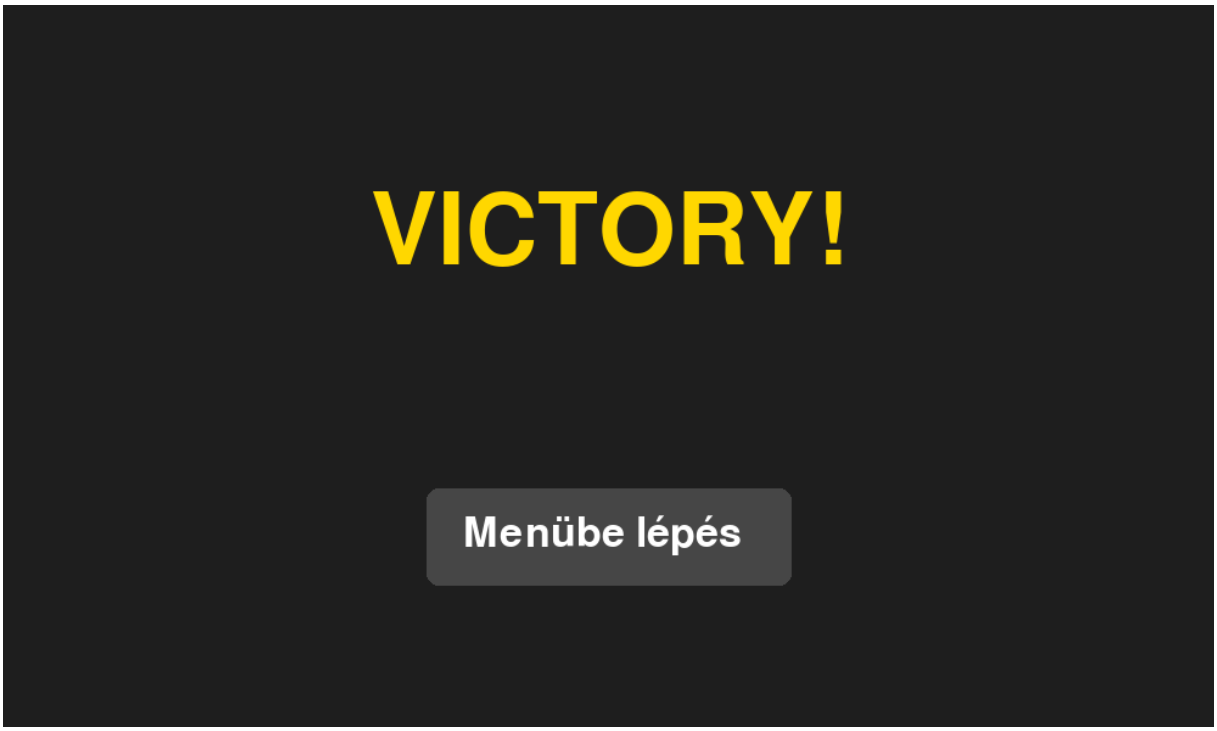
## Győzelem

### Win rendszer

A játék akkor jelöl győzelmet, ha:

- a játékos aktiválta a checkpointot,
- és hozzáér a célponthoz.

Ekkor a **Victory** képernyő jelenik meg.



## Kódszerkezet:

### main.py

- A játék fő futtató modulja.
- Állapotgép megvalósítása (menu, level_select, game, win, game_over).
- Pályákhoz tartozó teljesítettség nyilvántartása.

```
elif state == "level_select":
    selected_level = run_level_select(screen, completed_levels)

    if selected_level == "back":
        state = "menu"

    # 1. pálya mindig indítható
    elif selected_level == 1:
        state = "game"

    # 2. és 3. csak a megelőző pálya teljesítése után
    elif selected_level == 2 and completed_levels[1]:
        state = "game"

    elif selected_level == 3 and completed_levels[2]:
        state = "game"
```

```
elif state == "game":
    result = run_game(screen, selected_level)

    if result == "win":
        completed_levels[selected_level] = True
        state = "win_screen"

    elif result == "menu":
        state = "menu"

    elif result == "game_over":
        state = "game_over"

    elif result == "exit":
        running = False
```

**menu.py**

Tartalmazza:

- főmenü
- pályaválasztó menü
- game over képernyő
- win képernyő
- gombkirajzolás és eseménykezelés

```
screen.fill((30,30,30))
draw_title(screen, text: "Pályaválasztás")

# 1. pálya mindig elérhető
draw_button(screen, lvl1, text: "1. Pálya", GRAY, HOVER, WHITE)

# 2. pálya akkor nyitott, ha az 1. kész
col2 = GRAY if completed_levels[1] else (80,80,80)
draw_button(screen, lvl2, text: "2. Pálya", col2, HOVER, WHITE)

# 3. pálya akkor nyitott, ha a 2. kész
col3 = GRAY if completed_levels[2] else (80,80,80)
draw_button(screen, lvl3, text: "3. Pálya", col3, HOVER, WHITE)

draw_button(screen, back, text: "Vissza", RED, hover: (255,50,50), WHITE)
```

```
screen.fill(BLACK)

font_title = pygame.font.SysFont( name: None, size: 120, bold=True)
t_title = font_title.render( text: "GAME OVER", antialias: True, RED)
screen.blit(t_title, (WIDTH // 2 - t_title.get_width() // 2, 100))

font_msg = pygame.font.SysFont( name: None, size: 40)
t_msg = font_msg.render( text: "Meghaltál!", antialias: True, WHITE)
screen.blit(t_msg, (WIDTH // 2 - t_msg.get_width() // 2, 210))

draw_button(screen, retry_b, text: "Újrapróbálás", GRAY, HOVER, WHITE)
draw_button(screen, menu_b, text: "Főmenü", RED, hover: (255, 50, 50), WHITE)

pygame.display.flip()
clock.tick(60)

for event in pygame.event.get():
    if event.type == pygame.QUIT:
        return "exit"
    if event.type == pygame.MOUSEBUTTONDOWN:
        mx, my = event.pos

        if retry_b.collidepoint((mx, my)):
            return "retry"

        if menu_b.collidepoint((mx, my)):
            return "menu"
```

**game.py**

A játék legfontosabb modulja:

- pályabetöltés (load_level)
- player fizika (gravitáció, ütközés)
- ellenség mesterséges intelligencia
- mozgó platformok kezelése
- kvízrendszer
- halál és győzelem logikája
- sprite-ok rajzolása

```
player_img_right = pygame.transform.scale(pygame.image.load("sprites/player.png"), size: (40, 40))
player_img_left = pygame.transform.flip(player_img_right, flip_x: True, flip_y: False)

enemy_img = pygame.transform.scale(pygame.image.load("sprites/enemy.png"), size: (40, 40))
block_img = pygame.transform.scale(pygame.image.load("sprites/block.png"), size: (40, 40))
lava_img = pygame.transform.scale(pygame.image.load("sprites/lava.png"), size: (40, 40))
checkpoint_img = pygame.transform.scale(pygame.image.load("sprites/checkpoint.png"), size: (40, 40))
goal_img = pygame.transform.scale(pygame.image.load("sprites/goal.png"), size: (40, 40))

background_img = pygame.image.load("backgrounds/bg1.png")
background_img = pygame.transform.scale(background_img, size: (WIDTH, HEIGHT))
```

```
if enemy:
    dist = abs(enemy.x - player.x)

    if dist < 200:
        enemy_state = "chase"

    elif enemy_state == "chase" and dist >= 200:
        enemy_state = "return"

    if enemy_state == "chase":
        vx, vy = move_towards(enemy, player, speed: 2)
        enemy.x += vx
        enemy.y += vy

    elif enemy_state == "return":
        vx, vy = move_towards(enemy, enemy_home, speed: 2)
        enemy.x += vx
        enemy.y += vy

        if abs(enemy.x - enemy_home.x) < 3:
            enemy_state = "idle"
```

## Grafikai elemek:

A játék saját sprite-okat használ:

- Player (jobbra és balra néző állapot):

  Jobbra:  ,Balra: 
- Ellenség:

  
- Block:

  
- Checkpoint:

  
- Cél:

  
- Háttérkép

A sprite-ok minden pályán teljes képernyős háttérrel jelennek meg.

## Összegzés:

A Platformer Project egy teljes, működő prototípus, amely:

- működő menürendszerrel,
- tile-alapú pályával,
- interaktív kvízekkel,
- ellenség- és platformmechanikákkal,
- játékosvezérléssel,
- győzelem/halál logikával

rendelkezik.

A játék moduláris felépítésű, könnyen bővíthető:

- további pályákkal
- új kvízkérdésekkel
- több ellenségtípussal
- új mozgó platformokkal
- power-up rendszerekkel

Források:

https://chatgpt.com/

https://www.pygame.org/docs/

https://stackoverflow.com/questions