4. Rendszer vezérlés és architektúra

Ebben a fejezetben bemutatjuk, hogyan valósul meg az Infinite Loopers játék rendszervezérlése, a komponensek közötti kommunikáció, valamint a réteges architektúra. A leírás a H4 rendszerterv, analízis modell és UI terv alapján készült.

4.1 Réteges architektúra áttekintése

A játék architektúrája **logikai rétegekre** osztható, a H4 rendszertervben leírtak szerint:

1. **Megjelenítési réteg (UI / HUD / Menük)**
    - Ide tartoznak a grafikus elemek: háttér, platform, Player sprite, AnswerCube-ok, Coin-ok, HUD elemek, főmenü, Pause menü, Game Over képernyő.
    - A réteg a Pygame grafikus API-jára épül, közvetlenül a képernyő bufferre rajzol.
2. **Alkalmazási réteg (vezérlés)**
    - Fő komponens: **GameManager**-nek megfelelő game_loop() függvény és a menülogikát megvalósító menu() függvény.
    - Kezeli a játék fő állapotgépet: **Főmenü → Játék → Pause → Game Over / Win → Főmenü**.
    - Koordinálja a Player, AnswerCube, Coin, ScoreSystem és a HUD közötti együttműködést.
3. **Üzleti logika réteg (játéklogika)**
    - A Player mozgása, ugrás, gravitáció, ütközéslogika.
    - A matematikai feladatok és válaszkockák generálása (generate_levels(), setup_level()).
    - A combo rendszer, pontszámítás, időzítés és életkezelés.
4. **Adatkezelési réteg**
    - A High Score és egyéb tartós adatok fájlalapú tárolása (highscore.txt, későbbi bővíthetőség Settings irányába).
    - Az adatkezelés a ScoreSystem logikájába integrálva történik (a jelenlegi prototípusban ez még minimális, de a H4-ben definiált szerep már kijelölt).

4.2 Aktív és passzív objektumok

A H5 útmutató elvárása szerint megkülönböztetjük az **aktív** és **passzív** objektumokat.

Aktív objektumok / komponensek

Ezek saját frissítési logikával rendelkeznek, minden frame-ben (játékciklusban) meghívódnak:

- **GameManager (game_loop)**
    - A fő **játékciklus** megvalósítója.
    - Minden iterációban: eseménykezelés → logikai frissítés → ütközésvizsgálat → rajzolás.

- **Input-kezelés (Pygame event loop)**
    - Folyamatosan figyeli a billentyűzet és ablak eseményeit (QUIT, KEYDOWN, KEYUP).
    - Az eseményeket a GameManager dolgozza fel (ugrás, Pause, menüválasztás stb.).
- **Player**
    - update(keys, dt) metódusban kezeli a bemeneteket, ugrást, gravitációt és az állapotgépét (Földön, Ugrik, Leesik, Ütközött).
- **Lebegő objektumok**
    - AnswerCube, Coin, FloatingText, Puff – mind rendelkeznek update() metódussal, amely a lebegő mozgást, élettartamot, villogást, fade-out-ot kezeli.

Passzív objektumok

Ezeknek önálló „időbeli" viselkedésük nincs, csak adatot szolgáltatnak vagy egyszerűen kirajzolódnak:

- **Level definíciók** (feladatok, helyes/hibás válaszok listái).
- **Konstansok** (SCORE_CORRECT, SCORE_WRONG, SCORE_COIN, max combo idő, gravity stb.).
- **Adatfájlok** (highscore.txt, későbbi settings fájlok).

4.3 Vezérlési folyamat – fő játékciklus

A vezérlés központi eleme a **main loop (GameManager)**, amely a H4 rendszertervben bemutatott game loop mintát követi.

A folyamat fő lépései:

1. **Inicializálás**
    - Pygame inicializálása, ablak (screen) és óra (clock) létrehozása.
    - Főmenü meghívása: menu(screen, clock).
    - Játék indításakor: Player, AnswerCube-ok, Coin-ok, pontszám, életek, időzítők, combo, effekt-listák inicializálása.
2. **Eseménykezelés (InputController szerep)**
    - pygame.event.get() ciklusban:
        - QUIT → alkalmazás bezárása.
        - KEYDOWN:
            - **ESC**: kilépés / Pause menü megnyitása.
            - **SPACE / ↑**: Player ugrásának indítása.
            - **R**: Game Over / Win állapotban újraindítás.

        - KEYUP: ugrás gomb felengedése → player.release_jump().

3. **Logikai frissítés**
    - **Player.update()**: mozgás, ugrás, animációváltás.
    - **Időzítők frissítése**:
        - timer csökkentése (szintidő),
        - combo_timer frissítése (combo bontása inaktivitásnál).
    - **Objektumok frissítése**:
        - for puff in puffs: puff.update()
        - for cube in cubes: cube.update(t) – lebegés, flash effektek
        - for coin in coins: coin.update(t) – lebegés
        - for ft in floating_texts: ft.update(dt) – felugró szövegek mozgása.
4. **Ütközésdetektálás és állapotváltás** (üzleti logika)
    - **Player ↔ AnswerCube**
        - Ha helyes válasz kockával ütközik (c.correct == True):
            - combo növelése, pontszám = alap pont * combo, Level teljesítése → következő Level vagy Win állapot.
        - Ha hibás kockával ütközik:
            - lives csökkentése, pontlevonás, combo reset, esetleg Game Over.
    - **Player ↔ Coin**
        - Érme eltávolítása, pontszám növelése, FloatingText hozzáadása.
    - **Idő lejárta**
        - Ha timer <= 0: élet csökken, újraszámolás vagy Game Over.
5. **Rajzolás (Megjelenítési réteg)**
    - Parallax háttér rétegek kirajzolása (ég, felhők, dombok, platform).
    - Puffs, Player, Coin-ok, AnswerCube-ok, FloatingText-ek kirajzolása.
    - HUD: Score, Lives, Time, Combo, aktuális egyenlet szövegének kiírása.
    - Overlay felületek:
        - Szint-intro,
        - Game Over / Win képernyő (újraindítás, főmenüre visszalépés).
6. **Állapotok kezelése**
    - A GameManager logikai állapotai (nem formális enum, de flag-ek formájában):

        - show_level_intro – rövid bevezető a feladathoz.
        - game_over – Game Over overlay, R-rel újrakezdhető.
        - win – győzelmi képernyő, R-rel újrakezdhető.
    - A menübe való visszatérés vagy programból való kilépés is a főciklus visszatérési értékein és a main() függvény logikáján keresztül történik.

4.4 Üzenetküldés és kommunikáció a komponensek között

A komponensek közötti kommunikáció **egyszerű, függvényhíváson és attribútum-hozzáférésen alapuló „üzenetküldés”** formájában valósul meg, külső hálózati vagy aszinkron rendszer nélkül.

Fő kommunikációs csatornák:

- **GameManager → Player / AnswerCube / Coin / FloatingText / Puff**
    - update() hívások (aktív frissítés),
    - draw() hívások (kirajzolás).
- **GameManager → pontkezelés (ScoreSystem logika)**
    - Pontszám növelése/csökkentése ütközés vagy esemény alapján.
    - (A H4 adatbázis-terv szerinti ScoreSystem később külön modulba szervezhető, jelenleg a main.py-ben integráltan szerepel.)
- **GameManager → UI / HUD**
    - Szövegek, élet, idő, combo értékek átadása rajzoló függvényeknek (draw_text, HUD kirajzolás).
- **Pygame event rendszer → GameManager (InputController szerep)**
    - Billentyűesemények továbbítása, amelyek alapján a GameManager állapotot vált (Pause, Game Over, menü navigáció), vagy a Player metódusait hívja (jump(), release_jump()).

A kommunikáció **egyirányú vezérlést** követ: a GameManager „hívogatja” a többi komponenst, azok nem hívják vissza a GameManager függvényeit (a logika így egyszerűbb és jól követhető marad).

4.5 Láthatóság és hozzáférési szabályok

A H4 rendszerterv szerint a komponensek közötti láthatóságot úgy alakítottuk ki, hogy a csatolás laza maradjon, és az architektúra később is bővíthető legyen.

- **GameManager**
    - Látja a Player, AnswerCube, Coin, Puff, FloatingText objektumokat és az aktuális HUD-értékeket (score, lives, timer, combo).
    - Központi „orchestrator”, de a konkrét fizikai és vizuális logikát a specializált osztályokra bízza.

- **Player**
    - Nem tart közvetlen referenciát a GameManagerre; a GameManager adja át neki a billentyűzet-állapotot és a delta időt (dt).
    - Csak a saját mozgását és állapotát kezeli.
- **AnswerCube / Coin / Puff / FloatingText**
    - Saját belső adataikat kezelik (pozíció, lebegés, élettartam),
    - Nem férnek közvetlenül hozzá globális játékállapothoz (például score-hoz), csak a GameManager kérdezi le vagy módosítja ezeket a játéklogika részeként.
- **Adatkezelés (ScoreSystem / highscore)**
    - A fájlírás/olvasás a későbbiekben egy dedikált ScoreSystem modulba mozgatható, amelyhez csak a GameManager fér hozzá, így a GUI, Player stb. nem végez saját fájlműveleteket.

Ezzel a felosztással a vezérlés **átlátható**, a komponensek közötti függőségek **minimálisak**, és az architektúra alkalmas arra, hogy később új UI elemeket, extra játékmódokat vagy adatbázis-alapú mentést integráljunk anélkül, hogy a magjáték logikáját alapjaiban át kellene írni.