2. Rendszerterv

Dokumentum verzió: 1.0

Kiadás dátuma: 2025.11.16

Készítette: Infinite Loopers fejlesztőcsapat

1. Bevezetés

A rendszertervezési dokumentum az Infinite Loopers játék műszaki megvalósítását írja le. A H3-ban elkészített SRS (Software Requirements Specification) dokumentum határozza meg a funkcionális és nem-funkcionális követelményeket, míg jelen rendszerterv azt mutatja be, hogy ezek a követelmények hogyan valósulnak meg konkrét architektúrában, modulokban és komponensekben. A cél, hogy a fejlesztők számára egyértelmű, részletes terv készüljön, amely irányt mutat az implementációhoz, és alapot ad a későbbi bővítésekhez, karbantartáshoz.

A dokumentum leírja a réteges architektúrát, bemutatja a főbb komponenseket és azok felelősségeit, részletezi a modulok közötti kommunikációt, valamint ismerteti a választott tervezési mintákat és technológiai környezetet. Emellett kitér a külső interfészekre és a fejlesztés során figyelembe vett korlátozásokra is.

2. Architektúra áttekintés

Az Infinite Loopers játék architektúrája réteges felépítést követ, amelyet a logikai felelősségi körök elválasztása és az átláthatóság érdekében alakítottunk ki. A rétegek egymásra épülnek: a legfelső szinten a megjelenítési réteg biztosítja a grafikus UI-t és a HUD-ot, alatta az alkalmazási réteg koordinálja a játékmenetet, az üzleti logika réteg valósítja meg a játék szabályait, míg a legalacsonyabb szinten az adatkezelési réteg intézi a pontszámok és beállítások tartós tárolását.

Az alábbi ábra szemlélteti a réteges architektúrát:

2.1 Rétegek részletes leírása

Megjelenítési réteg (UI / HUD / Menük): ez a réteg felel a játék során megjelenő grafikus elemekért, ideértve a játékos karakter sprite-ját, az akadályokat, a háttérgrafikát, a pontszámokat jelző HUD-ot, valamint a különböző menüképernyőket (főmenü, beállítások, Game Over). A réteg a Pygame grafikus API-jára épül, és közvetlenül rajzol a képernyő bufferére.

Alkalmazási réteg (GameManager, InputController): itt valósul meg a játékmenet irányítása és az események koordinálása. A GameManager felelős a játék állapotainak kezeléséért (menü, játék, szünet, game over), a fő frissítési ciklusért és a többi modul közötti kommunikációért. Az InputController a Pygame eseménykezelésére építve feldolgozza a billentyűleütéseket és más bemeneteket, majd továbbítja azokat a megfelelő komponenseknek.

Üzleti logika réteg (Player, ObstacleManager, fizikai logika): ebben a rétegben találhatók a játék lényegi szabályai és működése. A Player osztály kezeli a karakter mozgását, ugrását, gravitációját és ütközéseit. Az ObstacleManager generálja és frissíti az akadályokat, biztosítva a folyamatos kihívást. A fizikai logika, mint például a sebességek és gyorsulások számítása, szintén ebben a rétegben valósul meg.

Adatkezelési réteg (ScoreSystem, fájlkezelés): ez a réteg felel a pontszámok, high score és esetleges beállítások tartós tárolásáért. Az adatkezelés egyszerű szöveges fájlok (pl. highscore.txt) segítségével történik, így elkerülhető egy teljes értékű adatbázis bevezetésének többletterhe. A ScoreSystem modul hivatott arra, hogy írási és olvasási műveleteket végezzen, miközben leválasztja az üzleti logikát a konkrét tárolási megoldásról.

3. Komponens-terv

A komponens-terv az egyes főbb modulok, csomagok és osztályok felelősségeit ismerteti. A cél az, hogy minden komponensnek jól meghatározott feladata legyen, minimális átfedéssel és laza csatolással, ezzel támogatva a karbantarthatóságot és a bővíthetőséget.

3.1 GameManager

A GameManager a játék szíve, amely a fő játékhurok (game loop) futtatásáért és a játékmenet egészéért felel. Fő feladatai közé tartozik az inicializáció (Player, akadályok, pontszám lenullázása), az állapotkezelés (menü, játék, szünet, game over), valamint a modulok közötti kommunikáció koordinálása. Minden képkockánál meghívja a frissítési metódusokat (update), ellenőrzi az ütközéseket, és gondoskodik a megfelelő képernyő kirajzolásáról.

3.2 Player

A Player komponens a játékos karakterének megvalósítása. Ez az osztály tárolja a pozíciót, a sebességet, az animációs állapotot, valamint megvalósítja az ugrás, esés és ütközés kezelésének logikáját. A Player az InputControllertől érkező vezérlőjeleket (pl. ugrás gomb) alakítja mozgássá, és minden frissítési ciklusban kiszámítja az új pozíciót a fizikai szabályok figyelembevételével.

3.3 ObstacleManager

Az ObstacleManager a pályán megjelenő akadályok létrehozásáért, életciklusáért és törléséért felel. Időzítés alapján dönt arról, mikor kell új akadályt létrehozni, majd gondoskodik azok folyamatos mozgatásáról és a képernyőt elhagyó objektumok eltávolításáról. Szoros kapcsolatban áll a GameManagerrel és a Playerrel az ütközések felismerése és kezelése érdekében.

3.4 ScoreSystem

A ScoreSystem komponens a pontszám és a high score kezelését végzi. Önálló felelősségi körébe tartozik a pontszám növelése (pl. megtett távolság, elkerült akadályok alapján), a high score betöltése a fájlból a játék elején, majd mentése a játék végén. A GameManager hívja meg a ScoreSystem megfelelő metódusait, így a pontkezelés logikailag elkülönül a játékmenet irányításától.

3.5 MenuSystem

A MenuSystem kezeli a főmenüt, beállítások menüt és Game Over képernyőt, valamint a különböző menüpontok között történő navigációt. A felhasználói választások alapján jelzi a GameManager számára, ha új játékot kell indítani, vissza kell lépni a főmenübe vagy ki kell lépni az alkalmazásból. A menük megjelenítéséhez a Megjelenítési réteg komponenseit használja.

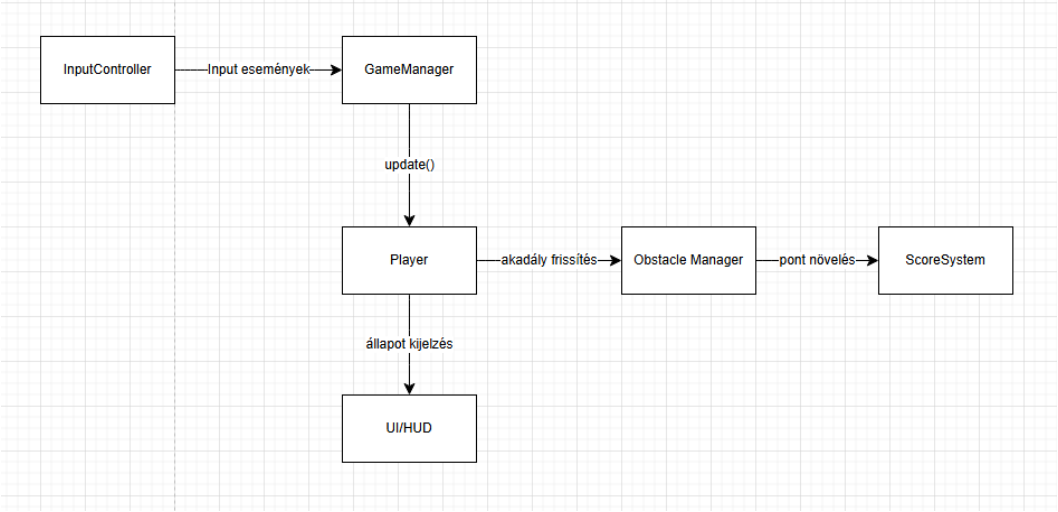
3.6 SoundManager

A SoundManager a hanghatásokért és a háttérzene lejátszásáért felel. Elkülönítve kezeli a rövid hangeffekteket (ugrás, ütközés, menü gombnyomás) és a folyamatosan szóló

háttérzenét. Kapcsolatban áll a GameManagerrel, amely események (pl. játék indítása, game over) alapján utasítja a megfelelő hangok lejátszására vagy leállítására.

4. Interakciós terv

Az interakciós terv azt írja le, hogyan kommunikálnak egymással a különböző komponensek a játék futása közben. A legfontosabb folyamat a game loop, amely minden képkockánál meghatározza, hogy milyen sorrendben hívódnak meg a frissítési és rajzolási metódusok, valamint hogyan kezelődnek a felhasználói bemenetek és az ütközések.



Az alábbi ábra egy magas szintű adatáramlást és modulközi kommunikációt mutat:

A game loop indulásakor az InputController lekérdezi a Pygame eseménykezelőjét, majd a felhasználói inputokat (pl. ugrás gomb) továbbítja a GameManager felé. A GameManager frissíti a Player és az ObstacleManager állapotát, majd az ütközések vizsgálata után értesíti a ScoreSystemet, ha pontszámot kell növelni, illetve a MenuSystemet, ha a játék véget ér. Végül a Megjelenítési réteg komponensei kirajzolják az aktuális állapotot a képernyőre.

5. Tervezési minták és technológiai környezet

5.1 Használt tervezési minták

A rendszerterv több ismert tervezési mintát alkalmaz. Az egyik legfontosabb az egyrészt MVC-szerű elrendezés, ahol a Megjelenítési réteg a "View", a GameManager és a vezérlő modulok a "Controller" szerepét töltik be, míg a Player, ObstacleManager és ScoreSystem bizonyos szempontból a "Model" szerepére hasonlítanak. Emellett Singleton-szerű megoldás alkalmazható a ScoreSystem és GameManager komponensekre, hogy ezekből csak egy példány legyen a futás során. A modulok közötti laza csatolást egyszerű interfészekkel és függőség-átadással biztosítjuk.

5.2 Technológiai környezet

A játék Python programozási nyelven készül, a Pygame könyvtár felhasználásával, amely a 2D-s grafika, eseménykezelés és hangkezelés működését biztosítja. A fejlesztés többnyire asztali környezetre (Windows, Linux, macOS) célzott, ahol a Python értelmező és a Pygame telepítése után a játék futtatható. Az adatkezelési réteghez egyszerű szöveges fájlok használata történik, extra adatbázis-réteg bevezetése nélkül.

5.3 Külső interfészek, API-k

A rendszer elsődleges külső függősége a Pygame könyvtár, amely grafikai és hang API-kat biztosít. A grafikus alrendszer feladata a képernyőfrissítés, a sprite-ok és háttérképek kirajzolása, míg a hang API a zenei és hangeffekt csatornákat kezeli. A fájlrendszerrel való kapcsolat a Python beépített I/O műveletein keresztül valósul meg (open, read, write). Nincs távoli szerverrel vagy webes API-val való kommunikáció, a játék teljesen offline környezetben fut.

6. Korlátozások és feltételezések

A tervezés során feltételezzük, hogy a játékot asztali gépen vagy laptopon futtatják, amely képes stabil FPS érték mellett futtatni a Pygame-alapú alkalmazást. A teljesítményoptimalizálás egyszerű technikákra korlátozódik (sprite-ok újrarajzolása, felesleges számítások elkerülése), nem alkalmazunk fejlett profilozást vagy párhuzamosítást. További feltételezés, hogy a felhasználó rendelkezik az alapvető Python-környezet telepítéséhez szükséges jogosultságokkal.

Korlátozás továbbá, hogy nincs beépítve többjátékos mód, online ranglista vagy szerveralapú adatkezelés. Az adatkezelés lokális fájlokra korlátozódik, így a high score csak az adott gépen érvényes. A tervezés ugyanakkor úgy épül fel, hogy a későbbiekben viszonylag egyszerűen bővíthető legyen hálózati kommunikációval vagy komplexebb adatbázis-réteggel, amennyiben erre igény mutatkozik.