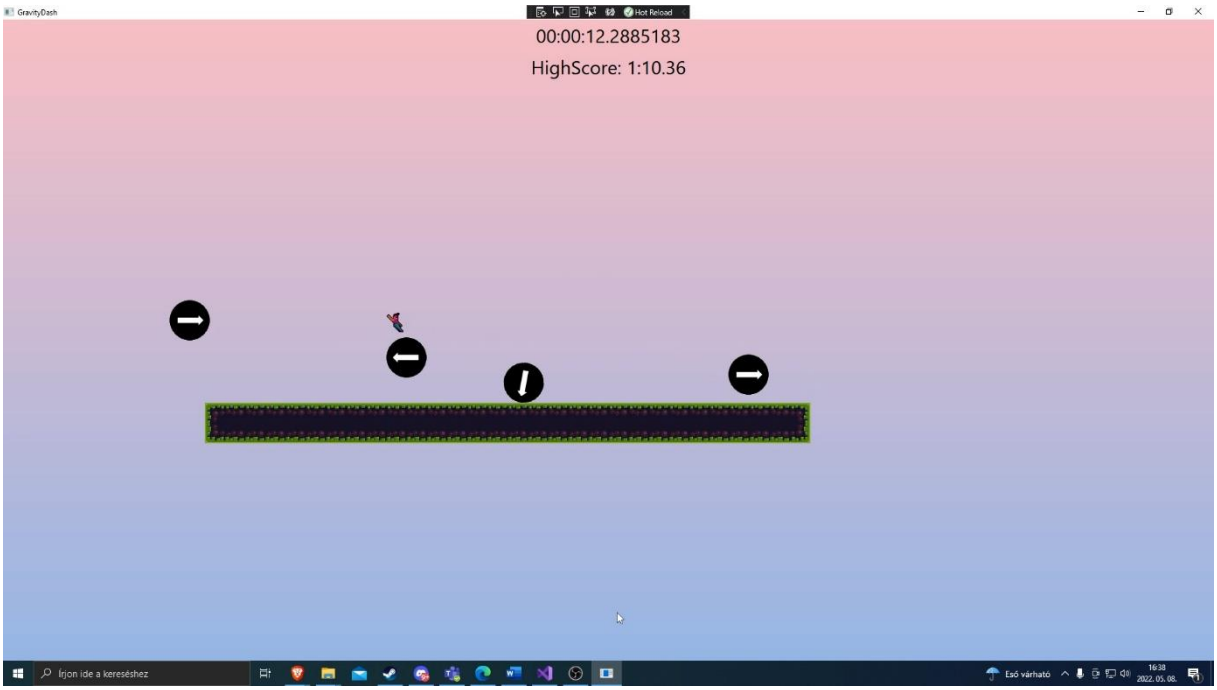
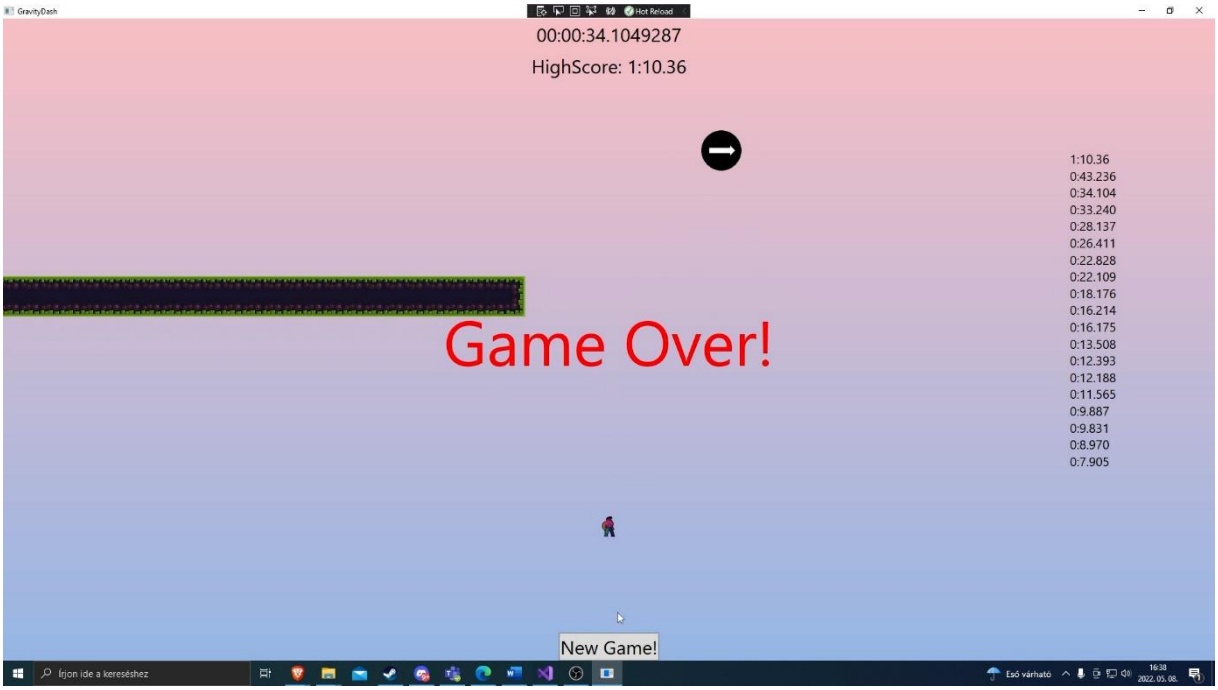
Gravity Dash dokumentáció

Játék rövid összefoglalója

A platformokon állva meg kell próbálni minél több ideig nem leesni a voidba, miközben a játék lövedékekkel bombáz minket, amelyek lelökhetnek minket, úgyhogy tanácsos ezeket elkerülni vagy elugrani előlük.





Models réteg (GravityDash.Models)

GameItem absztrakt osztály

Van egy Area nevű Geometry típusú adattagja (absztrakt, leszármazottak felülírják) és egy IsCollision nevű bool metódusa, amely egy másik GameItem példányt vár, és megnézi, hogy ez és a másik GameItem objektum geometriájának metszete nagyobb-e mint 0. (ha nagyobb, mint 0, akkor a geometriájuk összeér, ütköznek az objektumok)

IPosition interfész

Előírja két publikus double adattag létezését a megvalósító osztlyokban: X és Y koordináták, amelyek gettelhetők és settelhetők.

Player osztály

Megvalósítja: IPosition interfészt, leszármazik GameItem absztrakt ősosztályból.

Van egy ID egész szám adattagja, egy string típusú Name (Név) mező, egy 2 dimenziós sebességvektor és egy ImageBrush típusú karakterkinézet, amely a játék és karakter állapotának megfelelően folyamatosan változik a futásidőben (aktuális kinézetét egy statikus Skins listából nyeri).

Egy publikus Collision bool metódus pedig egy másik GameItemet várva megnézi, hogy a másik GameItem az egy Platform típusú-e, és ha igen beállítja, hogy a játékos megakadjon rajta. Számunkra az az eset a megfelelő, ha esés után fog meg minket a platform.

Illetve a játékos tud ugrani is, a földről vagy másik objektum felszínéről, illetve a levegőben ugrás ugrás is lehetséges, de csak egyszer és csak akkor, ha a játékos az ugrás előtt másik objektummal (canonball) érintkezett.

Cannonball osztály

A játékosokat veszélyeztető kilőtt lövedékek, amelyek levihetik őt a voidba. Megvalósítja: IPosition interfészt, leszármazik GameItem absztrakt ősosztályból.

Adattagjai egy karakterkinézet, két publikus X és Y double típusú pozíció, egy éppen aktuális szög, amelyben áll és egy 2 dimenziós sebességvektor.

Mozgató metódusa mindig megnöveli a pozícióját a sebességvektor megfelelő komponensével.

Egy publikus Collision bool metódus pedig egy másik GameItemet várva megnézi, hogy a másik GameItem egy másik lövedék-e, vagy egy platform vagy pedig a játékos. Amennyiben egy másik lövedék, a két lövedék összeütközik, és irányuk,

szögük megváltozik, amennyiben egy platform, a lövedék gurulni kezd rajta, ha pedig a játékos, akkor a játékost megfelelő sebességgel megfelelő irányban ellöki (megjegyzés: a játékos is tudja tologatni a lövedéket)

Level osztály

Van a szinteknek egy magassága és szélessége, egy listája a pályán található platformoknak és a pályán lévő lövedékeknek, illetve egy kétdimenziós egész szám tömb CollisionLayer, valamint egy szintén kétdimenziós egész szám tömb DrawingLayer.

Data réteg

Ebben a rétegben tárolja el a játék az elért eredményeket, időket. A játék beolvassa a „Scores.txt”-ből az eddigi eredményeket egy listába, hozzáfűzi egy új befejezett menet eredményét ehhez a listához, majd kiexportálja ezt a listát ugyanebbe a fájlba.

A GetHighScore metódus a legjobb eredmény megjelenítéséhez, míg a GetScoreList az összes eredmény megjelenítéséhez szolgáltatja az adatokat a főablak számára.

Main réteg

Ez a réteg felel a játék grafikus interfészéért. Itt tud a játékos az „Új játék” gombra kattintani, amely a NewGame() metódust futtatva elrejti az eredménytáblát, újra generálja a modellt, logikát és viewportot, majd ezeket átadja a Renderernek (megjelenítés).

Logic réteg

Bizonyos időközönként ágyúgolyót spawnol és az ütközéseket szimulálja/kezeli. Illetve kezeli az animációkat és a játékos mozgását.

Renderer réteg

Megjelenítésért felel, csak kapja az adatokat a gamemodelből és megjeleníti.

Repository réteg

Csak a levelt (szinteket) és a playert (játékosokat) kezel.

Ki mit csinált

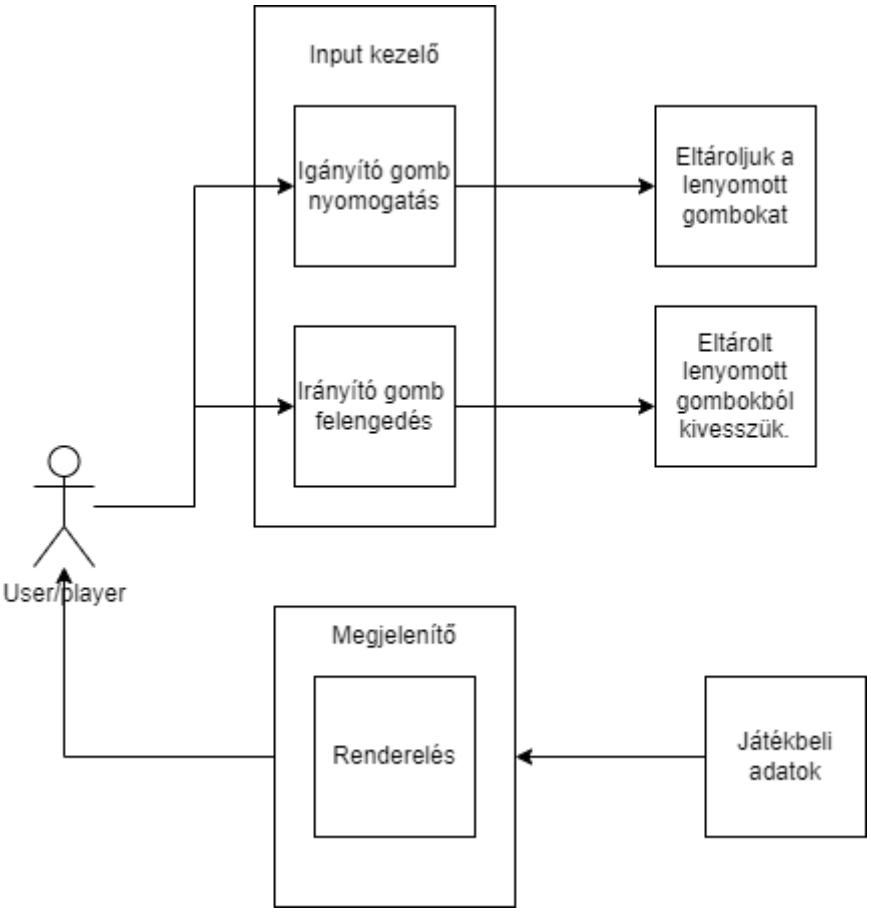
- Básthy Ádám László: Walking Skeleton készítés, Renderer réteg, Menu és HighScore rendszer
    - Nehezebb rész amire büszke vagyok: a Renderben az ágyúgolyók és a platform a kamerához (Viewport) relatív koordinátákon vannak kirajzolva, de a Fizika statikus globális koordinátákon alapszik.
- Griger Szabolcs: InGameLogic: physics (collision and movement) és karakter animációért felelős függvények és Taskok készítése
    - Nehezebb rész, amire büszke vagyok (1): A logic többszállas megvalósítása, melyet eleinte nehéz volt átlátni, megérteni.
    - Nehezebb rész, amire büszke vagyok (2): Az ágyúgolyók collisionjének realisztikus lekezelése hosszas vektorműveletekkel.
- Velenczei Ádám: Repository réteg & Models réteg készítése, dokumentáció írása, UML készítés
    - Nehezebb rész: dokumentáció megírása, főleg az UML

Hogyan ment a közös munka: Mindhármunknak ez volt az első alkalma, ami a multibranch fejlesztést illeti, és úgy érezzük, hogy a féléves feladat elkészítése során sikeresen megismerkedtünk a GitHub multibranch környezetével és annak alapvető használatával. Még architektünk is volt Básthy Ádám személyében, akinek a tervei alapján a munkát sikerült felosztanunk, és gond nélkül tudott dolgozni mindenki a részével.

Gameplay video linkje

https://youtu.be/ZdCAXnlq4cc

Use case Diagram



UML diagram