Játék leírás:

Egy többjátékos módra támaszkodó játékot fogunk elkészíteni.

Az alábbi tervezetnek az „első rész” fejezetet adnám, mert ha ezzel készen lennénk idő előtt, több elemet is szándékoznék elkészíteni.

Ezek a kulcsszavak és technológiák jöttek elő a többjátékos mód gondolatára:

- Nancyfx (Api)
- MongoDB (adattárolás)
- websockets (kliens – szerver kommunikáció)

A környezet egy txt alapján fog létrejönni. Az első fejezetben a következő dolgokat fogja tartalmazni: föld, egyes elemek a levegőben, amikre fel lehet ugrani.

A karakter képes ugrani, mozogni és lőni. A játékosoknak egymást kell majd eltalálniuk ezáltal veszítenek az életerejükből vagy lépnek szintet és válnak erősebbé.

Talán random megjelenő, felvehető életet is kéne biztosítani a pályán.

Játék bezárásakor a karakter állapota elmentésre kerül.

A játékos egy fiókhoz lesz kötve, amit a kliens megnyitásakor elérhet vagy létrehozhat.

Mindenek előtt a kliens megnyílásakor létrehozunk a kliens oldalon egy socket objektumot, ami csatlakozni fog a szervezhez.

A szerver figyel ezekre a csatlakozásokra és egy csatlakozásnál létrehoz egy socketet, ami egy új szálon figyel a kliens utasításaira, és elteszi ezt a socketet egy olyan listába, ahol az összes csatlakozott klienshez tartozó socket van. Pl. arra az esetre, ha mindenkinek egyszerre kellene kiküldeni valamilyen információt.

Bejelentkezésnél kezdetlegesen a játékos legutóbbi pozícióját és állapotát kapja meg a kliens és a szerver.

Ezek eltárolásához mongoDB adatbázist használunk, mert ingyen van a próbaverzió és könnyű csatlakozni hozzá, kezelni, nem igényel lokális adatbázist.

A Nancyfx-et találtam alkalmasnak az egyszerűbb kliens szerver kommunikációhoz, ami json fromában fogja az adatokat fogadni és visszaküldeni. Csak a bejelentkezéshez és a fiók készítéséhez fogjuk használni.

A többi kliens-szerver kommunikáció a kliens és a szerver által eltárolt socketek feladata lesz, ami byte streambe alakított object-eket fog küldözgetni. Ezek az object típusok egy Common-class-library-ben lesznek eltárolva, hogy elérhetőek legyenek a kliens és a szerver számára is.

Mikor a név és a jelszó megfelelőnek bizonyul, eltároljuk a játékost az aktuális adataival egy „GameState”-be. Központilag a szerveren fog ez az adathalmaz eltárolódni és egyes változtatások alkalmával ezt fogja a szerver a csatlakozott klienseknek kiküldeni.

Játékos mozgatásánál a kliens küld a szervernek információt arról, hogy merre szeretne a játékos mozogni, a szerveren a „GameState”-ben meg lesz változtatva az adott játékos pozíciója, és ez a módosított state fog kikerülni minden klienshez, a kliensek az adat érkezésekor a kapott state alapján fognak újra renderelni egyet.

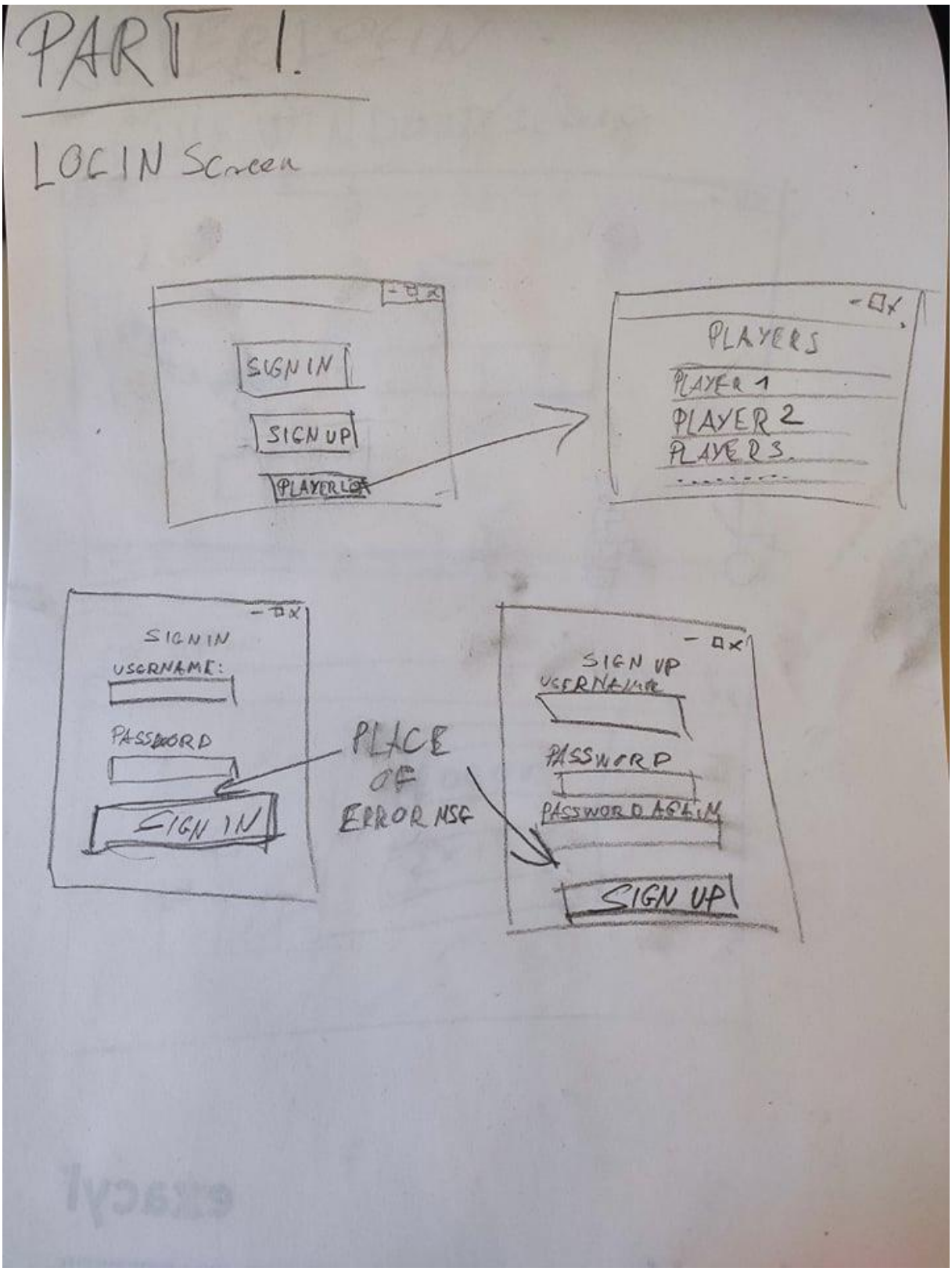
Hasonló elven szeretnénk elkészíteni a játék többi funkcióját, azonban még nem egyértelmű tisztán bizonyos feladatok megvalósítása. Amik még tervezésre szorulnak:

- karakter ugrása
- lövedék repülése

A feladatok részei és beosztásai:

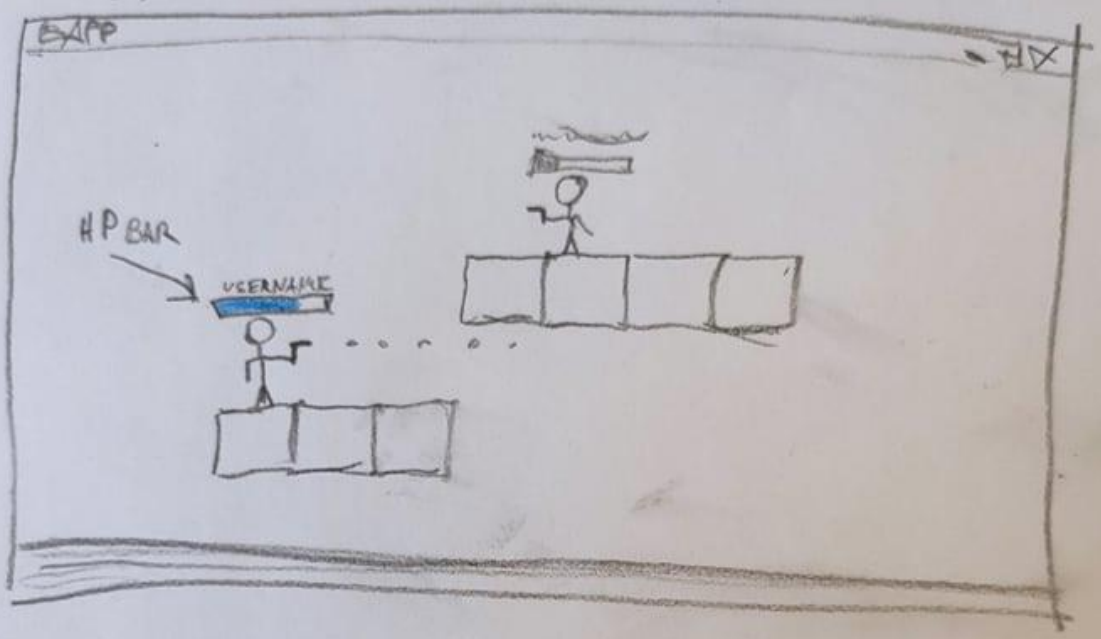
- api, adatbázis műveletek (Nagy Mátyás)
- szerver, szerver-logika, kommunikáció (Mészáros Zsolt, Balogh Bence)
- kliens, megjelenítés, kliens-logika (Mészáros Zsolt, Balogh Bence)

Csatolok néhány kezdetleges látványtervet is az elképzelésekről. A pálya elemei és a karakter kinézete még eldöntésre kerülnek.



AFTER LOGIN

GAME WINDOW/Screen



Esc ACTION