Házi Feladat

Programozás alapjai 2.

Takács András

1 Feladat:

A feladat egy 2 dimenziós játék, amiben kettő játékos versenyzik azon, hogy a másik játékos tetejére ugorjon ezzel pontot szerezzen. A játékosokat a billentyűzetről lehet irányítani és egy grafikus ablakban jelenik meg. A játék rendelkezik előre elkészített pályákkal, de a felhasználó is készíthet extra pályákat.

2 Feladatspecifikáció:

A feladat egy két fős játék elkészítése, melyben a két játékos egy-egy színes négyzetet irányít és a játék célja az, hogy a játékos a másik játékos által irányított négyzet tetejére ugorjon, ezzel pontot szerezve.

Amikor egy játékos pontot szerezt azt a játék egy képernyő közepére, a játékos színével megegyező, kiírt szöveggel jelzi, kiírja a játékos pontjainak a számát és egy rövid idő után visszaállítja a két játékost az eredeti helyükre. Amennyiben egyik játékos eléri a pályához tartozó győzelemhez szükséges pontszámot a program ezt szintén kiírással jelzi és egy új pályát tölt be nullázott pontokkal.

A két játékos egyazon számítógépről játszik, egy billentyűzeten osztozkodva, a kék játékos az A és D billentyűket használhatja horizontális mozgáshoz, a space-t ugráshoz és az S-et egy azonnali lefele gyorsuláshoz, míg a piros játékos esetében ezek megfelelője a két oldal nyíl a bal shift és a lefele nyíl.

A pálya, a játékosok és a szövegek grafikusan az SDL3 könyvtár segítségével vannak megjelenítve és a billentyűzet kezelésében is ez a könyvtár fog segíteni. A két játékos fizika szimulációjához a kód számontartja a gyorsulásukat és sebességüket, emellett biztosítja a helyes ütközéseket a másik játékossal és a pályaelemekkel. Ha egy játékos kimegy a pálya oldalán akkor a pálya másik oldalán bukkan fel, ha kiesik a pálya alján akkor a pálya tetejére kerül. Mivel a két játékos és a pályák is téglalapok, ezért az ütközések detektáláshoz elég azt ellenőrizni, hogy tetszőleges két tengelyhez képest el nem forgatott téglalap metszi-e egymást. Egyes pályaelemekről az ütközéskor visszapattannak a játékosok, más pályaelemek megölik a játékost.

A program a grafikus megjelenítés mellett egy parancssorban rögzíti a fontosabb eseményeket/fellépő hibákat. Egy ilyen lehetséges hiba például egy hibás pályafájl, ebben az esetben a program a terminálban jelzi, hogy melyik fájl, mely sorában keresendő a hiba. A pályákat „.gamemap” kiterjesztésű fájlokból az induláskor tölti be a program, minden a játékkal egy mappában lévő ilyen fájlt betöltésével megpróbálkozik. A formátum pontosabb specifikációja majd a részletesebb tervben lesz megtalálható.

3 Pontosított specifikáció:

A pályafájlok formátuma az alábbiaknak fog megfelelni:

*Pálya magassága(double, a tényleges magasság ennek a kétszeres)*

*Pálya háttérszíne(3 x byte)*

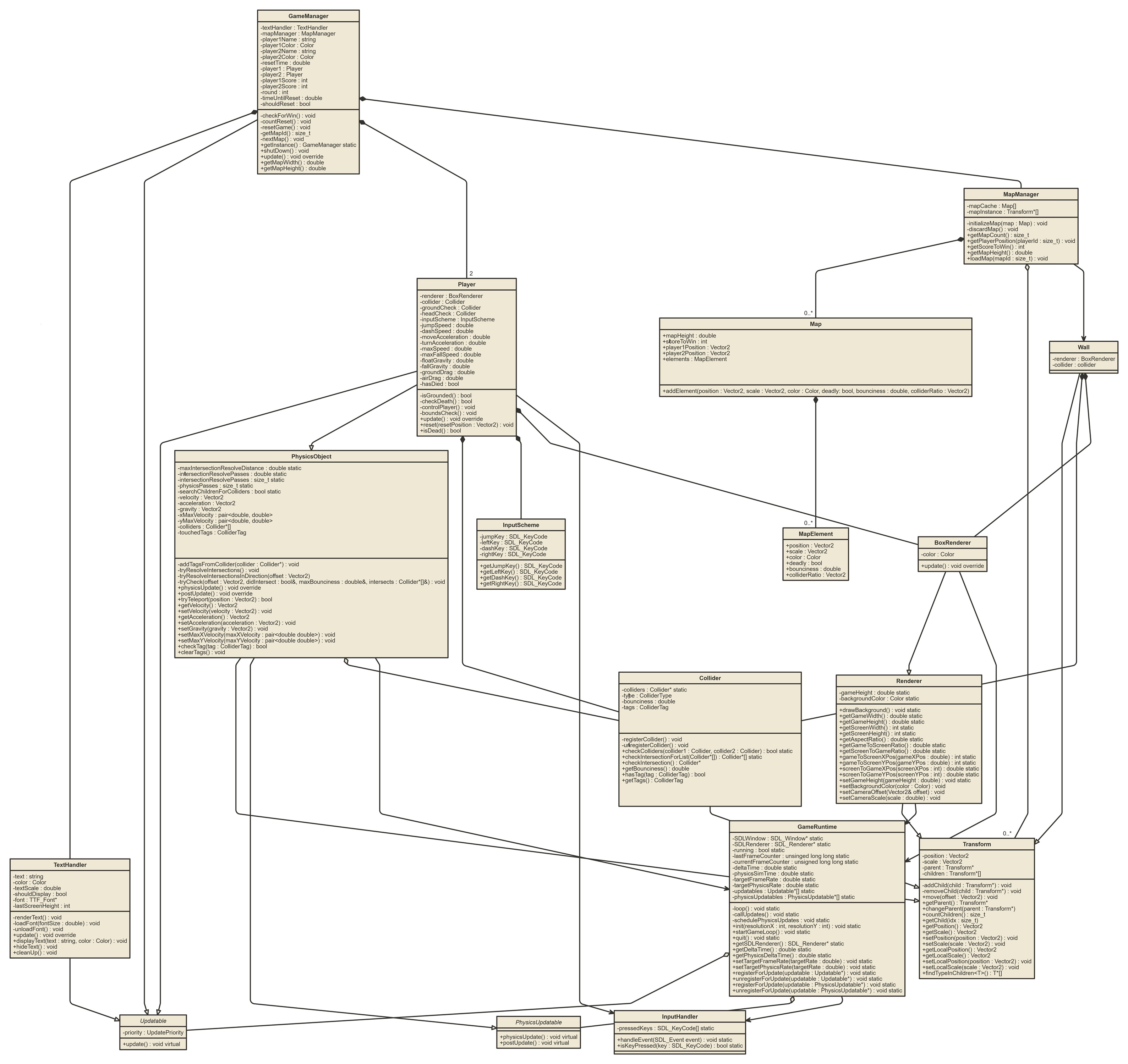
*Első játékos kezdőkoordinátái(float float)*

*Második játékos kezdőkoordinátái(float float)*

*N db falelem:(pozíció: float float, méret: float float, szín: 3 x byte, visszapattanás mértéke: float, fizikai kiterjedés aránya: float, float)*

4 Terv:

Az UML diagramot a <https://www.nomnoml.com> segítségével hoztam létre, részletezi a projekthez tartozó osztályokat és a közöttük lévő kapcsolatokat.



(Bele kell zoomolni hogy olvasható legyen)

5 Fontosabb algoritmusok:

5.1: Kettő el nem forgatott téglalap metszi-e egymást

Ezt az algoritmust használom arra, hogy ellenőrizzem, hogy két játékbeli objektum ütközik-e.

Tudjuk: center1, oldalhosszak1, center2 oldalhosszak2

//ha az első téglalap alja magasabban van a második téglalap tetejénél, így biztos nem metszgetik egymást

IF center1.y - oldalhosszak1.y > center2.y + oldalhosszak2.y:

RETURN HAMIS

//ha az első téglalap teteje alacsonyabban van a második téglalap aljánál, így biztos nem metszhetik egymást

IF center1.y + oldalhosszak1.y < center2.y – oldalhosszak2.y:

RETURN HAMIS

//ha az első téglalap bal széle jobbra van a második téglalap jobb oldalánál, így biztos nem metszhetik egymást

IF center1.x - oldalhosszak1.x > center2.x + oldalhosszak2.x:

RETURN HAMIS

//ha az első téglalap jobb széle balra van a második téglalap bal oldalánál, így biztos nem metszhetik egymást

IF center1.x + oldalhosszak1.x < center2.x - oldalhosszak2.x:

RETURN HAMIS

//különben kizárásos alapon a kettő téglalap metszi egymást

RETURN IGAZ

5.2: Fizikai objektum szimulációja:

A fizikai objektum rendelkezik sebesség- és gyorsulásvektorral, minden szimulációs lépés elején a sebességhez hozzáadja a gyorsulást szorozva a szimulációk gyakoriságának reciprokával, hogy konzisztens viselkedést biztosítson a képkockák számától függetlenül. Mivel a gyorsulás négyzetes kapcsolatban van a megtett úttal a szimulációs rátát konzisztensen tartjuk, hogy az ebből adódó hibákat elkerüljük.

Kiszámoljuk a lehetséges mozgatás maximumát: ez sebességvektor szorozva a szimulációk gyakoriságának reciprokával.

Felvesszük a próbamozgatás nagyságát: először ½, aztán 1/4 … ½^n

Először a x tengely mentén mozgatjuk a próbamozgatás x komponensével, ha nem ütközik semmi mással maradhat, ha nem visszaléptetjük a jelenlegi próbamozgatással

Ezt az y tengely mentén is megtesszük

Ezután felezzük a próbamozgatást, amíg el nem érünk ½^n-ig

Amennyiben a feljebb lévő algoritmus során ütköztünk valamivel, nullázzuk a sebességet és gyorsulást, amennyiben volt az ütközések között visszapattanós tulajdonságú objektum, akkor ennek arányában megfordítjuk a sebességet.

Az algoritmus garantálja, hogy az objektumok relatíve jól hozzásimulnak egymáshoz, n számától a lebegőpontos számítás hibáitól függően.