A programozás alapjai 3

Házi feladat: Breakout-klón

Név: Telek István

Neptun-kód: DVCYQK

1. rész: Specifikáció

A feladat leírása:

Házi feladatnak egy Breakout-klón elkészítését választottam, ami egy egyszerű, egyszemélyes játék. Kiindulásként az Arkanoid (https://en.wikipedia.org/wiki/Arkanoid) játékot vettem alapul, egy ehhez hasonló játékot szeretnék megvalósítani.

A játék lényege, hogy a játékos egy labdát pattogtat egy ütővel, a labda pedig az útjába kerülő téglákról lepattan, amivel azokat összetöri. A játék nehézsége, hogy a labdát elveszítjük, ha nem tudjuk eltalálni az ütővel. Többféle tégla is elképzelhető (pl. más pontszámot kapunk az összetörésükért).

A játék tartalmazni fog egy eredményjelzőt, amit egy fájlba automatikusan elmentünk. A játékos pontszámai így nyomon követhetők maradnak.

A játék felépítése:

A program egy egyszerű ablakból fog állni, felül egy menüvel, alul egy játéktérrel. A menüben van lehetőségünk a játékból való kilépésre. A játék irányítása billentyűzettel történik, a jobbra-balra nyíl lesznek használva. A nyílbillentyűkkel lehet az ütőt mozgatni. A játékos addig játszhat, amíg minden téglát ki nem ütött, vagy le nem esett a labdája.

A játéktér egy Canvas lesz, amit mi valósítunk meg, erre rajzoljuk majd ki a megfelelő objektumokat. Meg kell oldani majd az időzítést, illetve az események kezelését is. Az eredményjelző szövegként fog megjelenni a canvason.

A játékot egy időzítővel működtetjük (Timer osztály), ami a megadott intervallumban frissíti a játékteret. Az billentyűzet események kezelésére a beépített eseménykezelő funkciókat fogjuk használni (KeyListener interfész). A játékos pontjait egy JSON fájlban fogjuk tárolni, amit így könnyen hordozhatunk.

A megvalósítandó funkciók:

- A játék indítása a megfelelő gombbal (init)
- A játék vezérlése a billentyűkkel (game)
- Kilépés a játékból az ablak bezárásával, vagy a megfelelő menüpontban (exit)

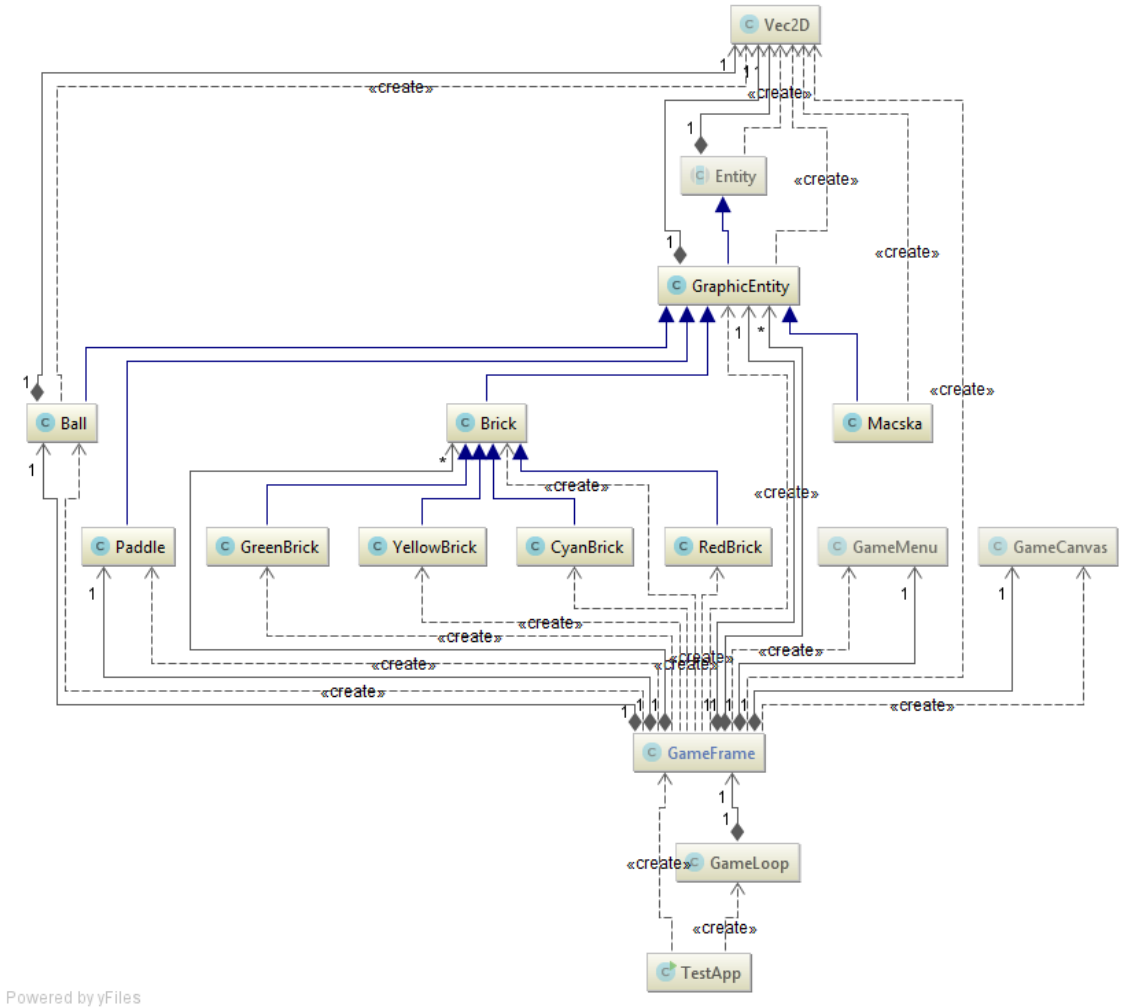
2. rész: Megvalósítás

Az osztályok

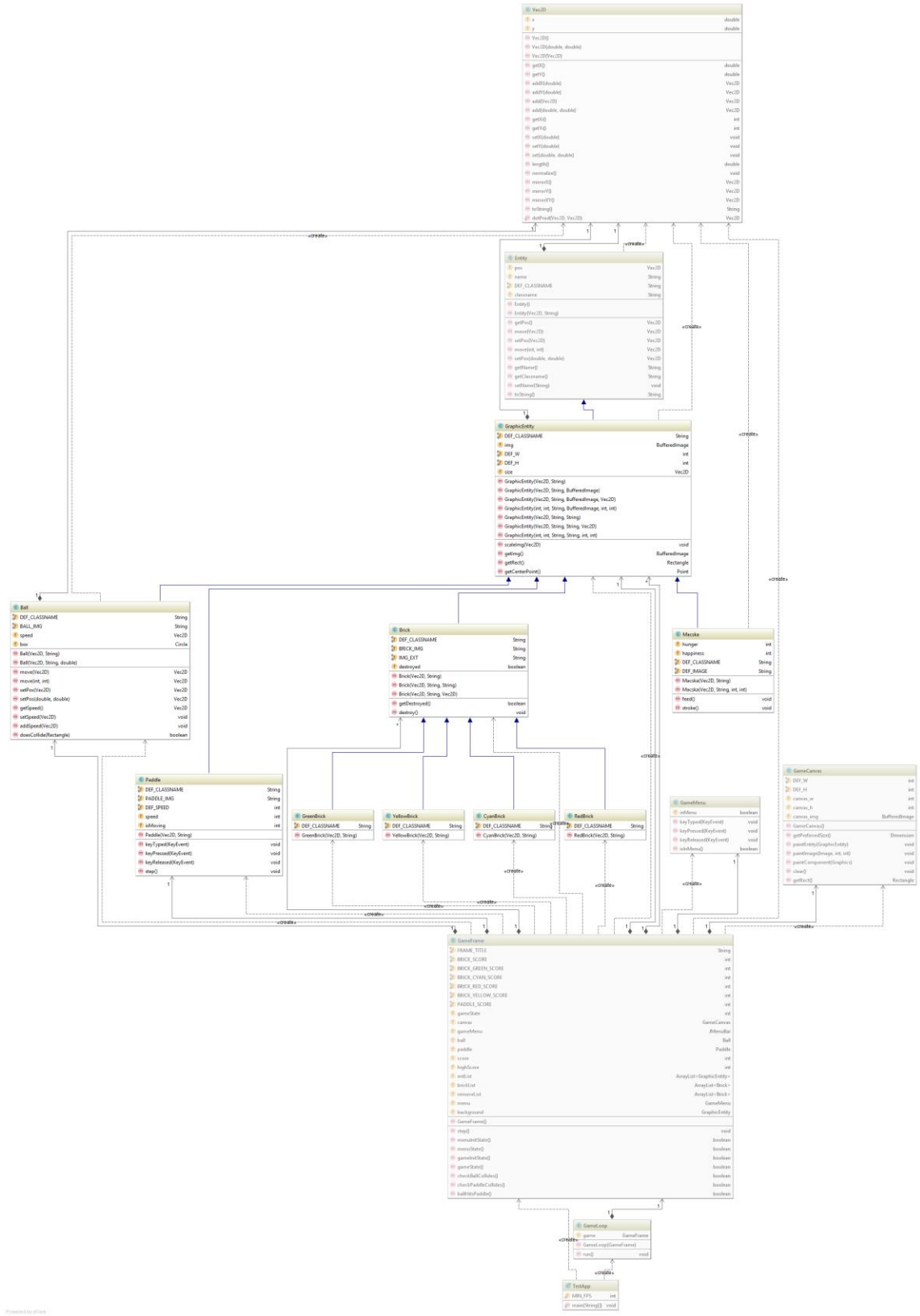
A legalapvetőbb osztály, mivel 2-dimenziós grafikai játékról van szó, egy kétdimenziós vektor osztály. Ennek a funkcionalitását nagyon sok osztály használja ki, ammint az osztálydiagramon látszik.

A játékunkban a tárgyakat (téglák, ütő, labda) egy közös absztrakt osztályból, az Entity osztályból származtatjuk. Ennek a funkcionalitását kibővítve létrehoztam egy GraphicEntity osztályt, ami képes egy textúrát és annak méreteit tárolni. Ez már önmagában is megjeleníthető a canvason. A Brick, Ball illetve Paddle osztályokat ebből származtattam, a speciális funkciókkal kibővítve. A Brick osztály gyermekei a különböző színű téglák, amiknek elbontásáért különböző pontok járnak.

A GameFrame osztály feladata a canvas elhelyezése az ablakban és a különböző entityket inicializálja, valamint a játék logikáját kezeli. A GameLoop ennek a step() metódusát meghívva lépteti a játék állapotát. A MainApp feladata létrehozni egy GameLoop típusú timert, az előre beállított intervallummal, majd elindítani azt.



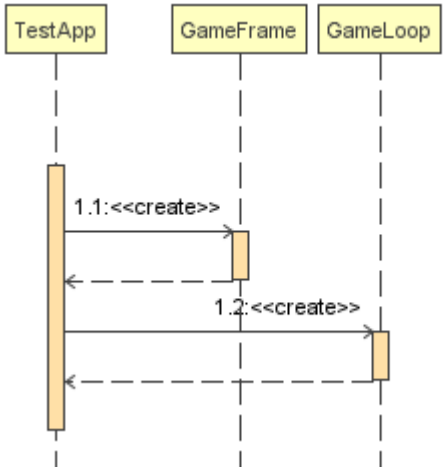
0-1 Az osztálydiagram metódusok nélkül



0-2 A teljes osztálydiagram

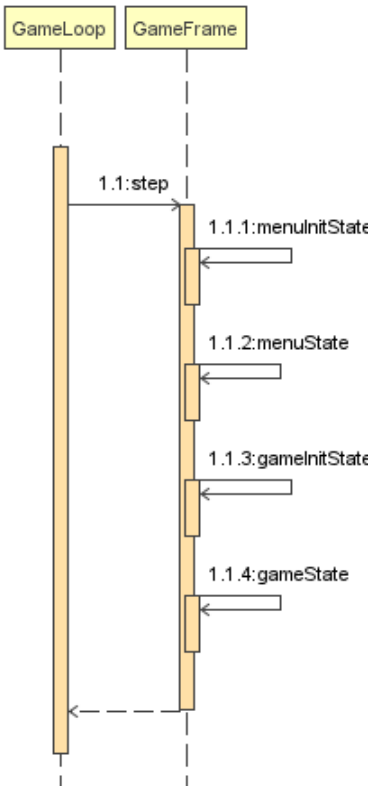
A funkciók leírása

A MainApp main függvénye létrehozza a GameFrame és GameLoop objektumokat, amivel elindul a játék inícializálása. Fontos, hogy először a GameFrame objektumot hozzuk létre, hiszen a GameLoop ezt a konstruktorában paraméterként veszi át.



0-3 A MainApp main metódusának szekvenciadiagramja

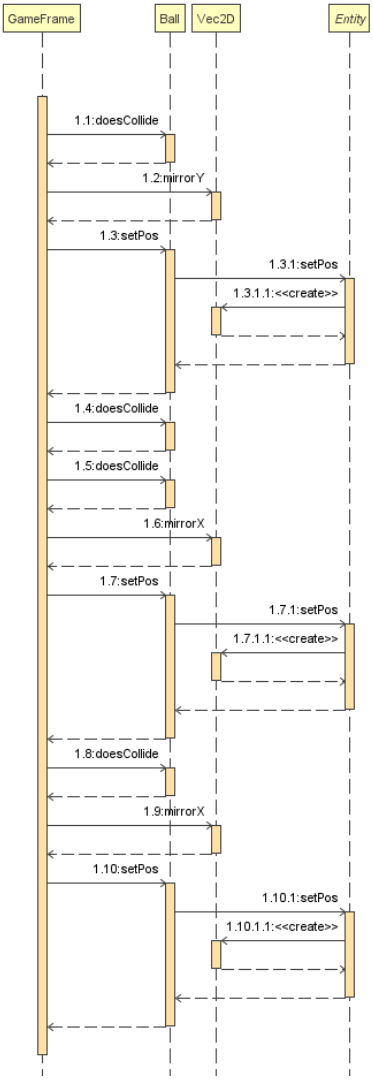
A GameLoop ezután minden ütemben meghívja a GameFrame step() metódusát, amiben a játék állapotát vezéreljük.



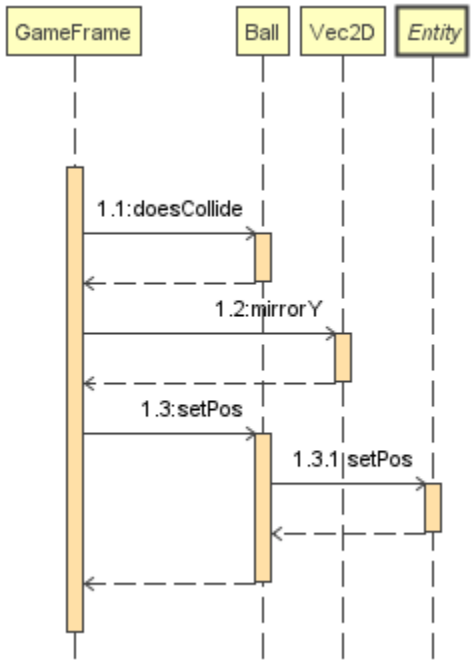
0-4 A GameLoop run() függvényének szekvenciája

A step() függvény egy állapotgép, ami a játék állapotait kezeli. A kiinduló állapot a menuInitState, ez inícializálja a menüt, a következő állapot a menuState, ebben kiírjuk a képernyőre az üdvözlő szöveget, valamint a játékos maximális pontszámát. A következő állapot a gameInitState, itt hozzuk létre a különböző entityket, majd ezután a gameState állapotba lépünk, ahol addig maradunk, amíg a játékos nem nyer vagy nem veszít. Ezután újra a menuInitState-be kerülünk, ahol az egész folytatódik elölről, addig, amíg ki nem lépünk a programból.

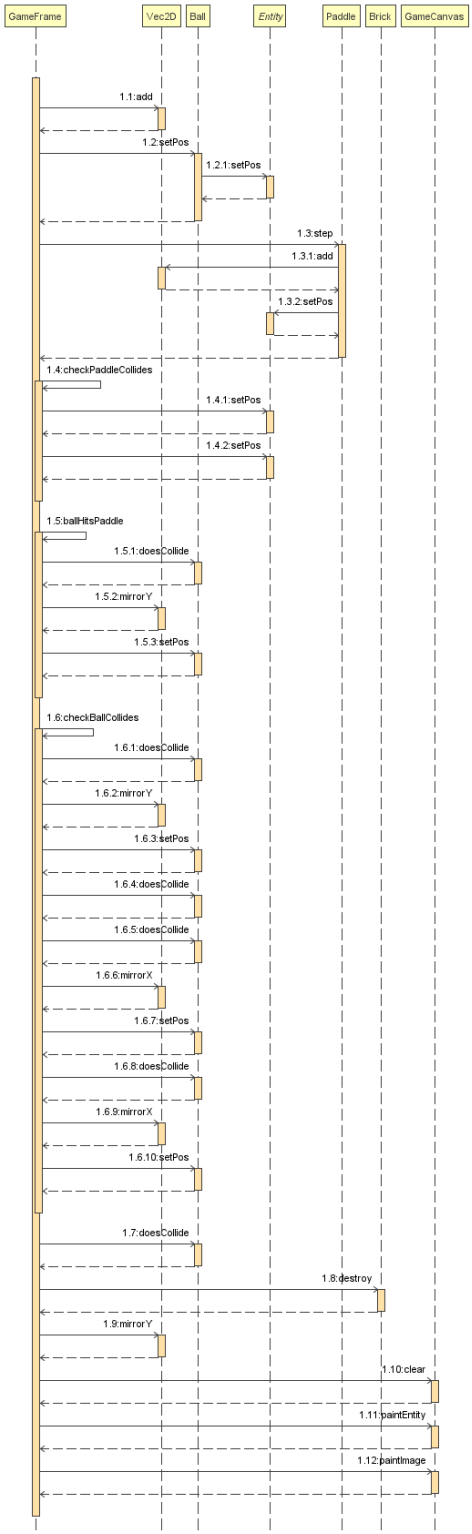
A gameState ezek közül a számunkra érdekes állapot, itt történik a játék vezérlése és a logika. Mint látható, ennek bonyolult a szekvenciája, nagyjából annyi történik, hogy mozgatjuk a labdát és az ütőt, majd ellenőrizzük, hogy nem-e akarnának kimenni a játéktérből (checkBallCollides). Ezután azt ellenőrizzük, hogy az ütőn pattant-e a labda (ballHitsPaddle), majd minden téglán ellenőrizzük, hogy nem-e azokon pattant a labda. Ha elfogytak a téglák, vagy a labda leesett, akkor vége a játéknak.



0-6 A checkBallCollides szekvenciája



0-5 A ballHitsPaddle szekvenciája



0-7 A gameState teljes szekvenciája

3. Összegzés

A feladatot azért választottam, mert régi kedvencem a Breakout játék, és szerettem volna egy egyszerű klónját megvalósítani. A játék még számos funkcióval kibővíthető, ez a felépítése miatt viszonylag könnyen megvalósítható, csak idő kérdése lenne. A feladat megoldása során törekedtem az átláthatóságra, valamint arra, hogy minél egyszerűbben lehessen bővíteni. Az eredeti tervek szerint a játékban hangeffektek is lettek volna, sajnos a Java hanglejátszás támogatása túl bonyolulttá tette ezt, ezért nem implementáltam.