Tartalomjegyzék

Mindenek elõtt

Hozzatok létre egy új projectet **Asteroids2** néven. Hozzatok benne létre egy **asteroids** package-et. Majd töltsétek le innen a hiányos java file-okat. (Ezek az elõzõ gyakorlat megoldásait tartalmazzák, plusz még egy-két módosítást, így mindenkinek le kell töltenie.)

Pontszámok

Az **AsteroidsGame** osztálynak van egy **statusbar** nevû változója, az **initUI**-ban van inicializálva (azaz itt jön létre egy objektum amit tárol). Láthatjátok, hogy egy 0-t tartalmazó Stringet kap, és ha elindítjátok a játékot azt is láthatjátok, hogy ezáltal a bal alsó sarokban egy 0 van a játékban. Ez lesz a pontszám, de még nem mûködik.

Most nézzünk bele a **MainLoop asteroidBulletCollision** metódusába. Ez fut le akkor, ha ütközik egy lövedék egy aszteroidával. Láthatjátok, hogy megkapja az aszteroidát ami ütközik és a lövedéket. Az aszteroidának meghívja a **destroy** metódusát majd a for ciklusban az újonnan létrejött aszteroidákat berakja az **asteroidContainer**-be (emlékezzetek vissza a **destroy** visszaadott egy tömböt az újonnan készítendõ aszteroidákról). A **destroy** hívása után kellene növelnünk a számlálót eggyel, egy aszteroida egy pontot ér.

- Hozzunk létre egy privát adattagot a **MainLoop**-ban, amiben tároljuk a pontokat (egész szám), én **score**-nak fogom nevezni.
- A következõ függvényhívás átállítja a **statusbar** értékét a megadottra (a **String.valueOf** átalakít pl egy **int**-et String-e):

```
statusbar.setText(String.valueOf( ide kell az érték ));
```

- Viszont van egy kis gond, hisz a **statusbar** az **AsteroidsGame** adattagja volt, azt itt nem érjük el. Hát csináljunk ide is egy referenciát hozzá:
  - Menjünk vissza az **AsteroidsGame** osztályba és hozzunk létre egy publikus függvényt **getStatusbar** névvel, ami visszaadja a **statusbar** adattagot. A visszatérési értékét ebbõl már tudnotok kell, hogy mi.
  - Térjünk vissza a **MainLoop**-ra, itt hozzunk létre egy megfelelõ típusú privát adattagot, amiben tároljuk a **statusbar**-t. Majd ezt vagy a konstruktorban vagy az **initBoard**-ban állítsuk be a következõ módon:

```
statusbar = parent.getStatusbar();
```

- - A parent, ha megnézitek az **AsteroidsGame** objektum, szóval elkérjük a **statusbar**-ját és így már a **MainLoop** is "http://wiki.math.bme.hutároljuk"http://wiki.math.bme.hu.
- Most, hogy már van **statusbar** itt is mûködni fog a korábbi **setText**-es kód, szóval próbáljátok ki.

- Ha még mindig 0-t mutat, ne felejtsétek el, hogy a **score** változót növelni kellene amikor egy aszteroida elpusztul, szóval tegyétek meg a növelést a **destroy** után.

Ûrhajó pusztulása

Ehhez sok dolgunk lesz. Át akarjuk írni a következõ osztályokat, hogy mindenki mindenkivel ütköztethetõ legyen:

- **AsteroidExtended**
- **SpaceShipExtended**
- **Bullet**

Elõbb az ötlet (aki magabiztos kezdje el a következõk megírni a programot és csak akkor nézzen bele a részletekbe, ha elakadna):

- Létrehozunk egy interface-t **Collidable** névvel, ami azt jelzi, hogy valami tud ütközni. Két metódussal, egyik ami elkéri az adott objektum ütközési "http://wiki.math.bme.hukörét"http://wiki.math.bme.hu (Circle objektum, már meg van írva). Másik amelyik az ütközésvizsgálatot végzi és igaz / hamisat ad vissza.
- Kibõvítjük az elõbb említett 3 osztályt, úgy hogy implementálja a **Collidable** interface-t. Szóval igazából csak azt a két függvényt kell megírni hozzájuk (mindegyikhez).
- Módosítjuk a **handleCollisions** függvényt a **MainLoop**-ban, hogy az újfajta ütközéseket kezelje.
- Végül vizsgáljuk minden aszteroidára a **handleCollisions**-ben, hogy ütközik-e a hajóval és ha igen, akkor meghívjuk a hajó egy olyan metódusát ami az **alive** adattagját **false**-ra állítja (az **alive** már létezik és jól mûködik).

És akkor most részletesebben:

- Interface-t úgy hoztok létre mint osztályt, csak ne a **Class**-ra kattintsatok, hanem az **Interface**-re. A **Collidable** interface két metódusa:
  - **getCollisionCircle**: publikus, **Circle**-t ad vissza, nem kap semmit
  - **collision**: publikus, **boolean**-t ad vissza, kap egy **Collidable**-t

Implementáljuk elõször a **Bullet**-hez:

- A **Bullet** feje nézzen ki így:

```
public class Bullet implements Collidable {
```

- Sírni fog, mert nincs implementálva a két **Collidable**-ben lévõ metódus, implementáljuk hát õket:
  - a **getCollisionCircle** csak adja vissza a **collisionCircle** adattagját a **Bullet**-nek.
  - a **collision** könnyen kitalálható, csak annyit mondok, hogy a **getCollisionCircle**-el mindkettõnek (this és bemenet) is el lehet kérni a körét, aminek van **overlap** nevû metódusa, ami megmondja, hogy két kör metsz-e.

Most implementáljátok a másik kettõt is, ugyanígy kell mindent csinálni. Ha megvan mind a három collideable akkor tudunk továbblépni.

Tisztogassunk elõször, töröljük ki az alábbi metódusokat:

- a **Bullet**-ben: **collideWithAsteroid**
- az **AsteroidExtended**-ben **collideWithBullet**

Ha ez megvan, akkor csak egy dolgot kell módosítanunk, jelez is hibát a **MainLoop**-ban, itt a **handleCollisions** metódus **if**-jében van a hiba, hisz pont azt töröltük ki ami ott áll. De semmi gáz, hisz pont ezért írtuk meg a **collision** metódusokat, hogy ez mûködjön, szóval cseréljük ki arra az ottani metódust (továbbra is **a**-n legyen meghívva és **b** legyen bemenet). Próbáljátok ki a játékot, simán mûködnie is kell.

Semmi új nem fog történni, de mostmár kényelmesen meg tudjuk írni az ûrhajó pusztulását:

- Írjunk egy új **for** ciklust a **handleCollisions**-be, ami az aszteroidákon megy végig (szabad lesni a fenti dupla ciklusból, a belsõ megy végig az aszteridákon).
- A cikluson belül vizsgáljuk, hogy az adott aszteroida ütközik-e az ûrhajóval (**player** a változó neve). Ha ütközik, akkor hívjuk meg a **player destroy** metódusát, még nem létezik, de nem baj.
- Írjuk meg a **SpaceShipExtended**-be a **destroy** metódust, csak annyit kell tennie, hogy az **alive** adattagját **false**-ra állítja (ezt már én létrehoztam és megoldottam, hogy eltûnjön az ûrhajó, ha hamis).

Egy szépséghiba lesz a dologgal, mostantól lehet úgy kezdeni játékot, hogy már rögtön veszítetek (aszteroida van a képrnyõ közepén). Találjatok ki valamit ezellen, például nézzetek bele az **AsteroidExtended** default konstruktorába ami random rakja le az aszteroidákat és érjétek el, hogy ne tehessen a kérnyõ közepsõ valahányszor valahányas négyzetébe. (**SPOILER**: pl ha olyan random számokat generált ami nem oké, akkor generáljuk õket újra addig amíg nem jók (**ULTRASPOILER**: while ciklus))

Reanimator

Tanulmányozzátok kicsit az **InputManager** osztályt és jöjjetek rá hogyan lehetne megoldani, hogy mondjuk az **ENTER** lenyomásával újraéledjen az ûrhajó (csak az **alive**-ot kell **true**-ra állítani.

További feladatok

Ha jól belenéztek látjátok, hogy az **alive** csak annyit szabályoz, hogy nem rajzolja ki az ûrhajót, ha hamis és nem tud lõni. A következõ gyakorlaton csinálunk normális Game Over képernyõt és restart gombot. Ezt akár elkezdhetitek azzal, hogy csináltok egy **pause** billentyût amivel ideiglenesen meg lehet állítani a teljes játékot. Elég hasonló módon meg lehet oldani, mint az **alive** volt, csak a **MainLoop**-ban, de lehetséges, hogy az ûrhajó osztályán is kell kicsit módosítani...

Gondolkozhattok inkább azon is, hogy hogyan kellene megoldani, hogy véletlen idõközönként teremjenek új aszteroidák. (Aztán azt is, hogy nehogy a játékos fejére teremjenek.)