

Java 4. házi feladat

Torpedó játék

A Feladat összefoglalása

Egy kétdimenziós torpedó játék kerül implementálásra, amellyel több játékos tud játszani egyszerre. Az első játékos beállítja a felhasználók számát, illetve a lerakható hajók méreteit és mennyiségét. Ezt automatikusan elhelyezi a játék a táblán, majd a felhasználók egymás után lőnek az ellenfél hajóira. A játékban két mód van, az egyikben az a felhasználó nyer, amelyik utoljára marad, a másikban az, aki a legtöbb találatot éri el.

A feladat részletes specifikálása

- **Hajók**

A játékokban 2, 3, 4 és 5 egység hosszúságú hajók közül lehet választani. A játék elején az első játékos állítja be, hogy melyik mérettípusból és mekkora mennyiséggel szeretnének játszani (itt történik a játékosok számának specifikálása is). Minden játékos ennyi hajóval rendelkezik, amit a gép fog elhelyezni a táblán úgy, hogy a hajók ne keresztezzék egymást (a hajók által lefedett területek legyenek diszjunktak).

- **Tábla**

A tábla egy $N \times N$ dimenziós rács. $N = ((\sum(\text{hajók hossza} - 1) \times (\text{játékosok száma} / 2) + 1) / 2$, majd ezt fölfelé kerekítjük, ahogy a játékosok számát is. Pl: 3 játékos, két 2, két 3, egy 4 és egy 5 hosszúságú hajóval szeretne játszani (per játékos), akkor $(2 \cdot (2 - 1) + 2 \cdot (3 - 1) + 1 \cdot (4 - 1) + 1 \cdot (5 - 1)) \cdot (3/2) = 13 \cdot 2 = 26$, $(26 + 1) / 2 = 13.5 \rightarrow 14$, tehát a tábla 14×14 méretű lesz. Minden játékos csak a saját hajóit látja kezdetben.

- **Harc**

Minden játékos a saját körében egy lövést adhat le az ellenfelek hajóira. A lövést leadni egy segéd táblán lehet, ahol látszik, hogy hova lőttek eddig a játék folyamán, ez egy GUI-s felület, ahol be lehet kattintani a lövés helyét. Ha az adott pozíción volt hajó, akkor egy kis lángocskát rajzol a program a megfelelő helyre, amit minden játékos lát. Ezt követően pontot ad a lövést indító játékosnak. Annak a felhasználónak, akit a találat ért, a hajója kigyullad a saját tábláján. Ha nem talált a lövés, akkor minden játékos segéd tábláján megjelenik egy X jel.

- **Automatikus lövés**

$\forall$ játékos lövési körében fut egy óra 30 másodperctől, és ha lejár, akkor a gép végzi el a lövést helyette. Gépi lövést úgy kell szimulálni, hogy meg kell vizsgálni az aktuális tábla állapotát és annak alapján egyenletes eloszlással kell kiszámítani a célzandó cellát. Tehát,

ha van a táblán 40 olyan cella amibe még nem lőttek és nem az adott játékos hajója foglalja el (teljesen mindegy, hogy áll-e rajta ellenséges hajó, vagy sem), akkor $\forall$ cellára $\frac{1}{40}$ lesz a valószínűsége, hogy őt célozzuk.

- **Ellenfél egyszerű implementálása**

Az előzőből könnyen megvalósítható egy egyszerűbb mesterséges intelligencia. Tetszőleges számú játékos ellen tetszőleges számú AI játszhat a fent leírt lövési struktúrával. Az AI darabszámát az első felhasználó állítja be, akárcsak a hajók paramétereit.

A **szerver** felelős a játék lebonyolításáért. Tárolja a hajók helyzetét, és a leadott lövéseket, továbbá figyeli, hogy valamelyik játékos nyert-e. Tetszőleges számú kliens kiszolgálására képes. Az első játékos specifikálja a szerver felé, hogy hány játékos és hány hajó lesz a következő játékban. A kliensek csatlakozása után elhelyezi a hajókat és elindítja a játékot.

A **kliens** a megjelenítésért és az interakciókért felelős GUI felületen keresztül. Meghatározhatja a játékos nevét, csatlakozik a szerverhez. A játékos lépéseit továbbítja a szerver felé, lövés esetén a lövés helyét megkapja a szervertől, tárolja és megjeleníti. A kliens jelenítse meg, hogy éppen melyik játékos lő (emelje ki a nevét), illetve kezelje, ha valamelyik játékos akkor is lőni akar, amikor nem a saját köre van.

Tehát a **kliens** GUI-n két tábla lesz látható. Az egyiken az adott játékos saját hajóit, illetve az azokat ért találatokat láthatja. A másikon az eddigi támadások lesznek megjelenítve ($\forall$ játékosé) - tűzzel a találatok, X -el az elvétett lövések. Továbbá láthatóak a pontszámok és a játékosok nevei.

Java követelmények

- A program kövesse az OOP alapelveit (tehát interface-ek, abstract osztályok stb.)!
- Használj megfelelő kivételkezelést a lehető legtöbb helyen! Tehát hibás felhasználói beavatkozás esetén ne omoljon össze a rendszer, illetve kliens oldalról ne lehessen összeomlasztani a szervert!
 - Ha lecsatlakozik egy kliens, ahelyett, hogy jönne ne álljon le a szerver.
 - Ha érvénytelen indexet küld a kliens azt is kezelje a szerver.
- Használj szálkezelést és figyelj oda, hogy a GUI ne akadjon!
- A GUI felület ne csak egymásra dobált komponensek halmaza legyen, alakíts ki minimális dizájnt és legyen könnyen átlátható!

Apró hint: Az interneten található torpedó megoldásoktól több ponton is eltér a feladat specifikációja. Véleményünk szerint nem érdemes az internetes megoldásokból kiindulni a feladat megoldása során, mert több időt vesz igénybe egy ilyen program átalakítása, mint az egészet megírni az általunk tanultak és a specifikáció szerint.

Leadási határidő: 2018. 05. 16. 8:59

Pluszpontok

A következő részfeladatok egészen izgalmasak, de a megoldásuk nem kötelező. Aki mégis rászánja magát, számos plusz pontra tehet szert.

- **Bonyolultabb mesterséges intelligencia**

Természetesen a mesterséges intelligenciát a végtelenségig lehet bonyolítani, pontosítani. Lehet adni neki egy matematikai heurisztikát, amelyben kiszámítja, hogy a tábla melyik részén lehet a legnagyobb valószínűséggel hajó. Például, ha már csak 4 darab 5 hosszú egység van a pályán, akkor nem érdemes a két kimaradásos cellákba lőni. Ha egy hajót megsebzett akkor érdemes azt körül lőnie és teljesen elpusztítani. Mint az egyszerű AI esetén, az első játékos specifikálja, hogy hány darab (és milyen erősségű) gépi ellenfél játszon. Csak a képzelet és az idő szabhat határt. :)

- **Egyszerű chat program**

A játékba könnyen implementálható egy egyszerű *broadcast* alapú chat program, vagyis ha egy játékos ír, akkor azt az összes többi felhasználó látja és reagálhat rá (természetesen ez alól az AI kivétel).