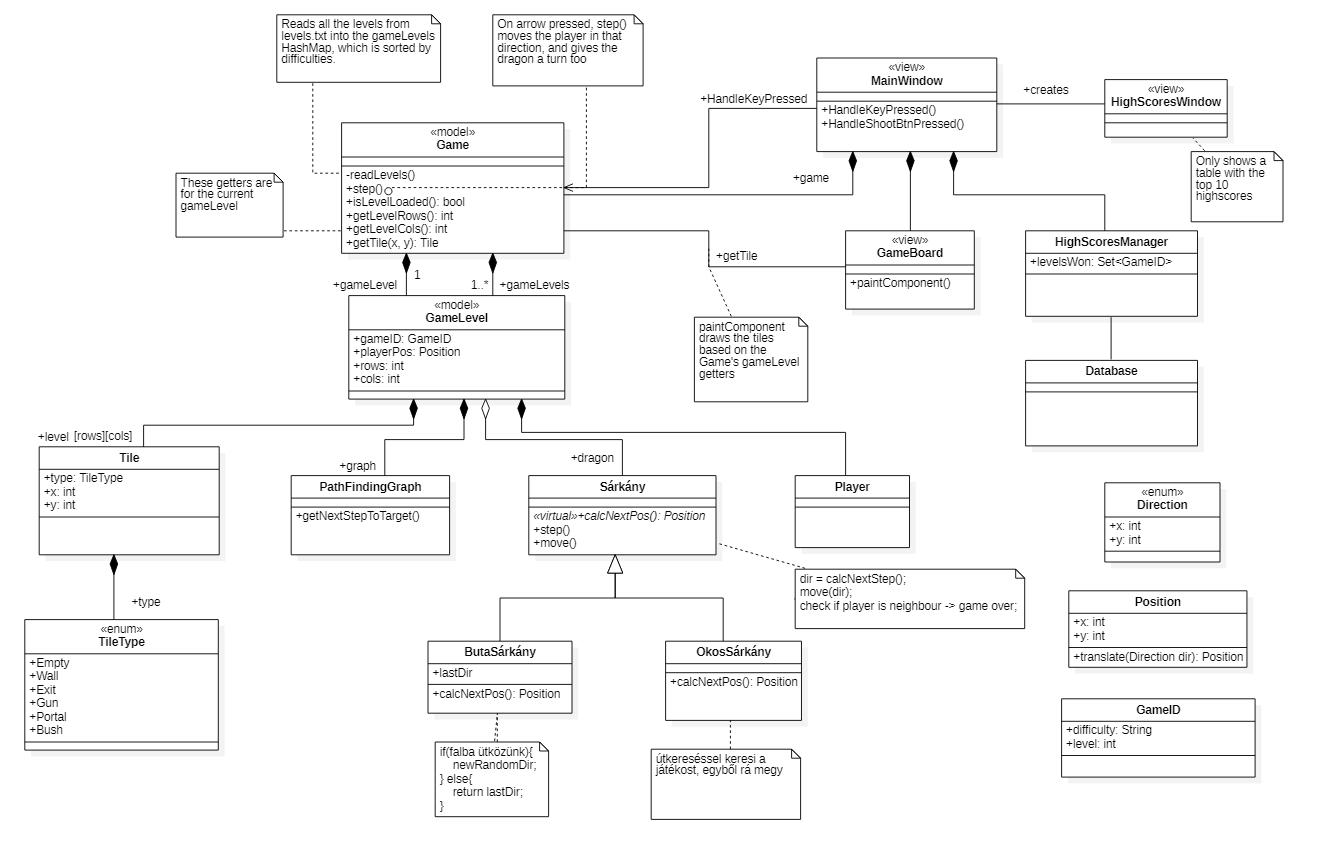
|  |  |  |
| --- | --- | --- |
| **Tasi Zoltán György** | **Progtech 3. beadandó/3. feladat** | 2021.december 08. |
| HM37UQ |
| [tasi.zoltan07@gmail.com](mailto:tasi.zoltan07@gmail.com) |
| 8. csoport |

# Feladat: Labirintus játék

Készítsünk programot, amellyel egy labirintusból való kijutást játszhatunk. A játékos a   
labirintus bal alsó sarkában kezd, és a feladata, hogy minél előbb eljusson a jobb felső   
sarokba úgy, hogy négy irányba (balra, jobbra, fel, vagy le) mozoghat, és elkerüli a   
labirintus sárkányát.   
  
Minden labirintusban van több kijutási útvonal. A sárkány egy véletlenszerű   
kezdőpozícióból indulva folyamatosan bolyong a pályán úgy, hogy elindul valamilyen   
irányba, és ha falnak ütközik, akkor elfordul egy véletlenszerűen kiválasztott másik   
irányba. Ha a sárkány a játékossal szomszédos területre jut, akkor a játékos meghal.   
Mivel azonban a labirintusban sötét van, a játékos mindig csak 3 sugarú körben látja a   
labirintus felépítését, távolabb nem. Tartsuk számon, hogy a játékos mennyi labirintuson   
keresztül jutott túl és amennyiben elveszti az életét, mentsük el az adatbázisba az   
eredményét. Egy menüpontban legyen lehetőségünk a 10 legjobb eredménnyel rendelkező   
játékost megtekinteni, az elért pontszámukkal, továbbá lehessen bármikor új játékot   
indítani egy másik menüből. Ügyeljünk arra, hogy a játékos, vagy a sárkány ne falon   
kezdjenek.

Extra követelmények: Legyen tobbfele sarkany tobbfele algoritmussal, az egyik utkeresessel talalja meg a jatekost. Legyenek benne teleport kapuk, meg olyan mezok, ahol el lehet bujni (lathatatlan lesz a sarkannyal szemben). Illetve lehessen egy fegyvert is talalni, amivel egy lovedeket ki lehet loni (ertelemszeruen ha a sarkanyt igy eltalajuk az elpusztul).

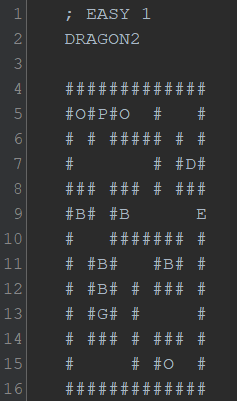
# Megoldás UML diagramja:



# Megoldási terv:

A program egy MainWindow-ból fog indulni. Ez létrehozza magának a Game és a HighScoresmanager osztályt, amik az üzleti logikával foglalkoznak. Miután ezek létrejöttek, a MainWindow először létrehozza és beállítja a pálya kiválasztásához szükséges menüpontokat, majd készít és a saját ablakához appendál egy Board osztályt, és betölti az első pályát. Illetve ez a MainWindow kezeli majd a felhasználótól jövő inputok eseményeit is, ez később lesz részletezve.

A Board osztály fogja magát a játékmezőt megjeleníteni. A Game-nek éppen betöltött GameLevel minden mezőján átiterál, és a helyes helyen megjeleníti a Tile-okat. Ez a Java Swing Graphics interfészén keresztül csinálja, 32x32 méretű képek rajzolásával.

A Game osztály konstruálása alatt file-ból betölti a pályákat. Egy file-ban tárolt pálya formátuma így néz ki:

* Első sor egy ’;’ jellel jelzi hogy új pálya kerül betöltésre
* Utána jön a GameId, ami egy nehézségi fokozat + sorszám
* Ezután jön hogy milyen fajta sárkány lesz a pályán
  + DRAGON1: Ez az a sárkány, ami véletlenszerűen választja az utát
  + DRAGON2: Ez az a sárkány, ami útkereséssel egyből a játékoshoz megy
* Ezután soronként jön a pálya felépítése, ahol a jelek ezeket a mezőket jelentik:
  + #: Fal
  + P: Játékos kezdőpontja
  + E: Kiút a labirintusból
  + D: Sárkány kezdőhelye
  + G: Fegyver
  + B: Bokor, amiben el lehet bújni
  + O: Portál, amivel el lehet teleportálni egy másik véletlenszerű portálhoz

Miután betöltődött a pálya, a játékos és a sárkány felváltva lépegetnek, először a játékos kezd. Ez addig történik, ameddig a játékos vagy el nem jut a céljához, vagy pedig a sárkány megöli (tehát egy szomszédos cellába lép a játékoshoz). Ekkor a játékos tovább léphet egy következő pályára, vagy ha meghalt, feltöltheti az elért pontszámát a Highscores-ba egy névvel együtt.

A játékos többféle speciális mezőre is léphet. Ezek:

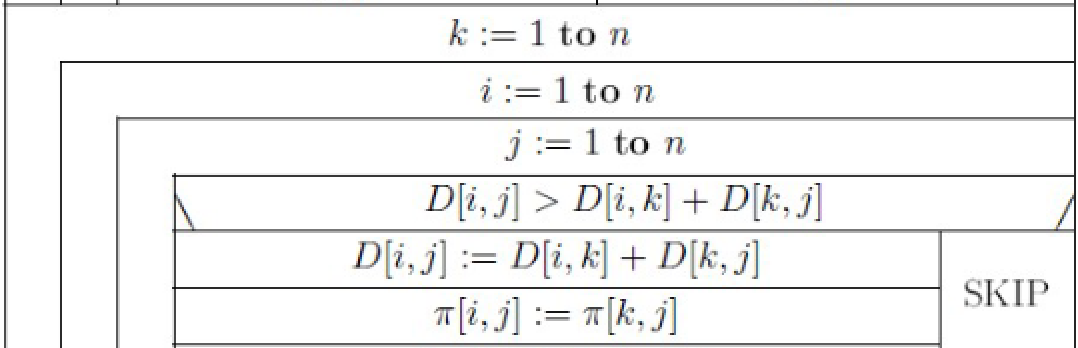
* Bokor:amíg egy ilyen mezőn van a játékos, a sárkány nem is léphet a mezőjére, nem is sebesítheti meg, illetve az okosabb sárkány nem is próbál felé menni.
* Fegyver: Ha erre lép a játékos, akkor utána tud egyet lőni. Ekkor létrejön egy Bullet objektum, ami a megadott irányba addig megy, amíg valaminek nem ütközik, utána eltűnik. Ha sárkányt talál el, akkor az meghal, az adott játékra eltűnik.
* Portál: Ha erre lép a játékos, egy másik véletlenszerűen kiválasztott portálra teleportálódik a játékos. Természetesen ha csak egy másik portál van, akkor biztosan oda.

# Útkereső algoritmus elemzése:

Ha az okos sárkány (DRAGON2) van a pályán, akkor az egy előre készített gráf alapján fog a játékos felé haladni. Ezt a gráfot (PathFindingGraph) mindig a pálya betöltésekor hozzuk létre, és csúcsai a játékos által elérhető mezők lesznek. Ez először feltöltődik a következő képpen: ha két csúcs szomszédos, akkor 1 lesz a távolságuk, ha nem szomszédosak, végtelen. Ezután lefuttatunk rajta egy Floyd-Warshall algoritmust, amivel megkapjuk minden csúcspárra a legoptimálisabb utat.

Az algoritmusban a D mátrix-ban lesznek tárolva a távolságok, D[i, j] lesz az i és j csúcs közötti távolság. A Pi mátrixban meg a közvetlen megelőző csúcsokat tároljuk.

A legoptimálisabb utat minden egyes csúcspárra egy Floyd-Warshall algoritmussal számoljuk ki, ami így néz ki alapból:



Itt a D mátrix-ban lesznek tárolva a távolságok, D[i, j] lesz az i és j csúcs közötti távolság. A Pi mátrixban meg a közvetlen megelőző csúcsokat tároljuk, tehát Pi[i, j] az i és j csúcs között vezető optimális úton j csúcs közvetlen megelőzőjét tartalmazza.

Az okos sárkány ezzel a gráffal a getNextStepToTarget() metóduson keresztül kommunikál, ami egészen addig lépeget Pi[kezdőcsúcs, célcsúcs] közvetlen megelőzőin, ameddig az út legelső lépését vissza nem kapja, és ide fog lépni a sárkány. Ha nem tud utat találni oda, akkor a kezdőcsúcs pozícióját adja vissza.

Ennek az algoritmusnak a műveletigény Theta(n^3), azt pályánként egyszer kell lefuttatni, utána a Pi mátrixban való visszafejtés már Ordó(n) idejű.

# Esemény-eseménykezelők:

MainWindow-on belül:

* Ha valamelyik pályaválasztó menübe lép, akkor a Game-en belül meghívja a loadGame metódust a megfelelő gameID-val
* Ha az Escape gombot megnyomjuk, Game-en belül meghívja a loadGame metódust a mostani gameID-val (restart)
* Ha valamelyik nyíl billentyűt megnyomjuk, akkor a Game-en keresztül abba az irányba megpróbálja elmozgatni a játékost. Ha ez sikeres (nem falba megyünk), ellenőrizzük hogy speciális Tile-ra léptünk-e, ha van Bullet a pályán léptetjük, utána a sárkány is lép egyet. Ellenőrizzük, hogy nyertünk vagy vesztettünk-e.
* Ha vettünk már fel egy felhasználatlan fegyvert, akkor vagy valamelyik az ablak tetején található gombra, vagy AWSD lenyomásával lőhetünk a megadott irányba, ezzel létrejön egy Bullet objektum. Ellenőrizzük hogy eltalált-e valamit, majd a sárkány is léphet. Ellenőrizzük, hogy nyertünk vagy vesztettünk-e.

Bullet.step()

* A lövés irányába lép egyet, és ellenőrizzük, hogy eltalált valamit.

Dragon.step()

* Ha a sárkány még életben van, akkor a típustól függő módon lépünk egyet, majd ellenőrizzük, hogy a játékost meg tudja e ölni.

Nyerés/vesztés:

* Ilyenkor a járék megáll, játékosunkat már nem tudjuk tovább léptetni.
* Ha meghalunk, akkor felugrik egy párbeszédablak, amibe nevünket beleírva feltudjuk tölteni a HighScoresba az eredményünk.
* Ha nyertünk, tovább léphetünk egy következő pályára.