3. Beadandó feladat dokumentáció

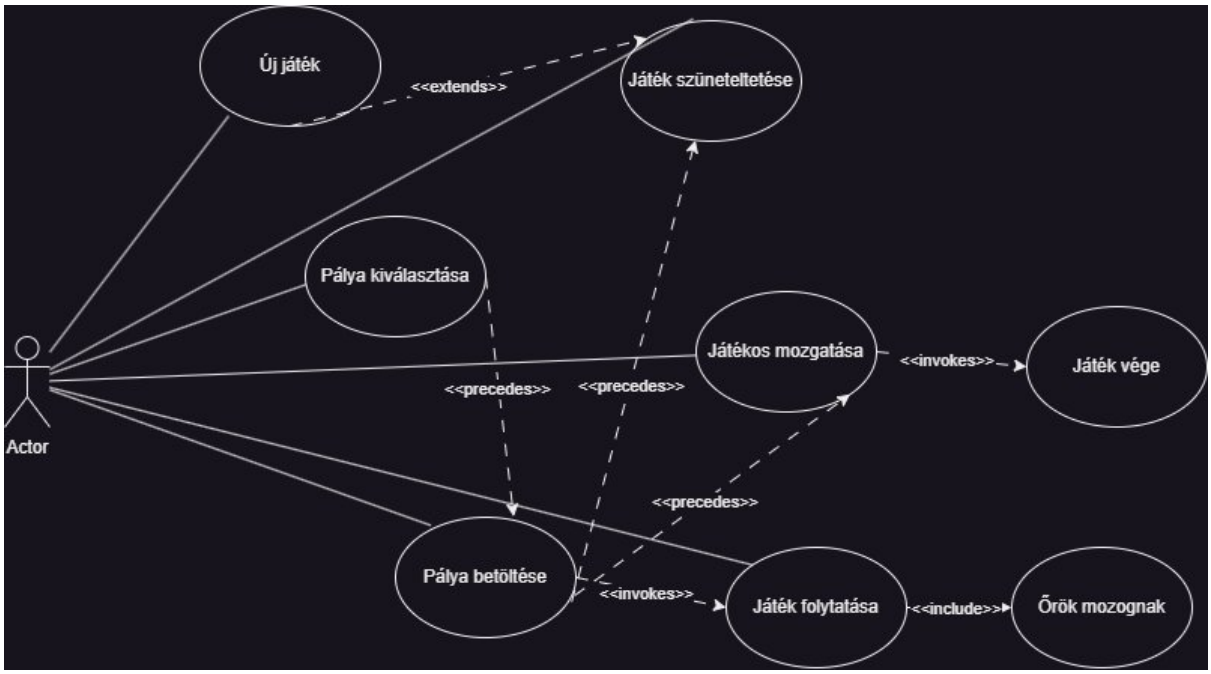
Készítette:
Bekovics Dániel
E-mail: g9e74r@inf.elte.hu

Feladat: Lopakodó

Készítsünk programot, amellyel a következő játékot játszhatjuk.
Adott egy $n \times n$ elemből álló játékpálya, amely falakból és padlóból áll, valamint őrök járőröznek rajta. A játékos feladata, hogy a kiindulási pontból eljusson a kijáratig úgy, hogy közben az őrök nem látják meg. Természetesen a játékos, illetve az őrök csak a padlón tudnak járni.
Az őrök adott időközönként lépnek egy mezőt (vízszintesen, vagy függőlegesen) úgy, hogy folyamatosan előre haladnak egészen addig, amíg falba nem ütköznek. Ekkor véletlenszerűen választanak egy új irányt, és arra haladnak tovább. Az őr járőrözés közben egy 2 sugarú körben lát (azaz egy 5 × 5-ös négyzetet), ám a falon nem képes átlátni.
A játékos a pálya előre megadott pontján kezd, és vízszintesen, illetve függőlegesen mozoghat (egyesével) a pályán.
A pályák méretét, illetve felépítését (falak és kijárat helyzete, játékos és őrök kezdőpozíciója) tároljuk fájlban. A program legalább 3 különböző méretű pályát tartalmazzon.
A program biztosítson lehetőséget új játék kezdésére a pálya kiválasztásával, valamint játék szüneteltetésére (ekkor nem telik az idő, és nem léphet a játékos). Továbbá ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, hogy győzött, vagy veszített-e a játékos.

Elemzés:

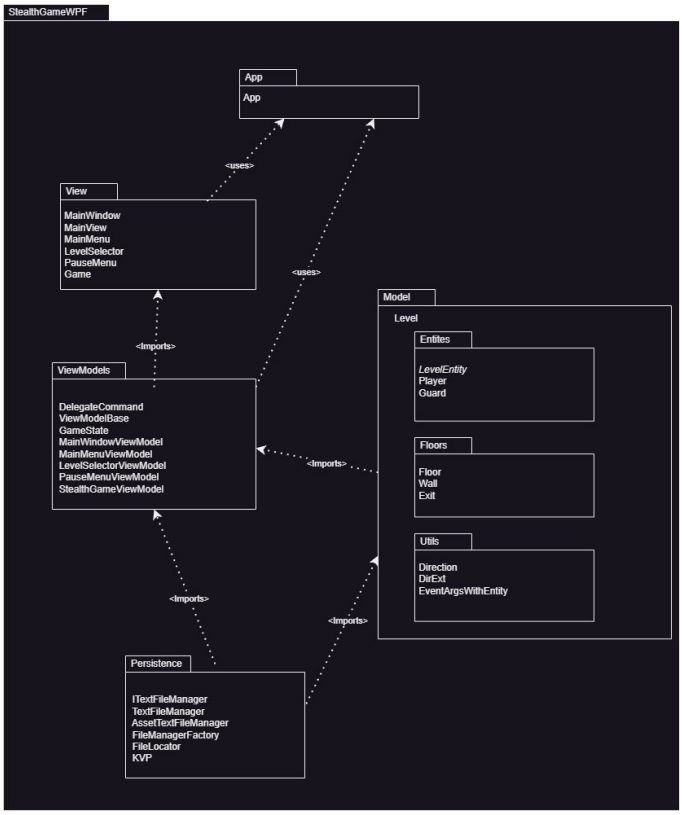
- A játékos kezdhet új játékot, ahol minimum 3 pálya közül választhat
- A pálya egy n x n-es mátrixként reprezentálható, ahol a cellák lehetnek padlók, falak vagy a kijárat(ok). Minden cella egy listában tárolja a rajta lévő entitásokat(őr, játékos).
- Az őrök változtassanak 2 másodpercenként poziciót
- A játésost a WASD billentyűkkel irányítjuk és minden mozdulata után meg kell néznünk, hogy hova lépett és az a kijárat-e, vagy benne van-e az őr látóterében.
- Az Escape billentyű lenyomásakor megállíthatjuk a játékot és akár új játékot is kezdhetünk a feljövő “Pause menu”-ben
- Ha a játékost meglátja az őr, vagy eljut a kijárathoz legyen vége a játéknak.
- A megjelenítésért AvaloniaUI kezelőfelület és egy UniformGrid, illetve képek felelnek.



Ábra 1: Felhasználói eset diagram

Tervezés:

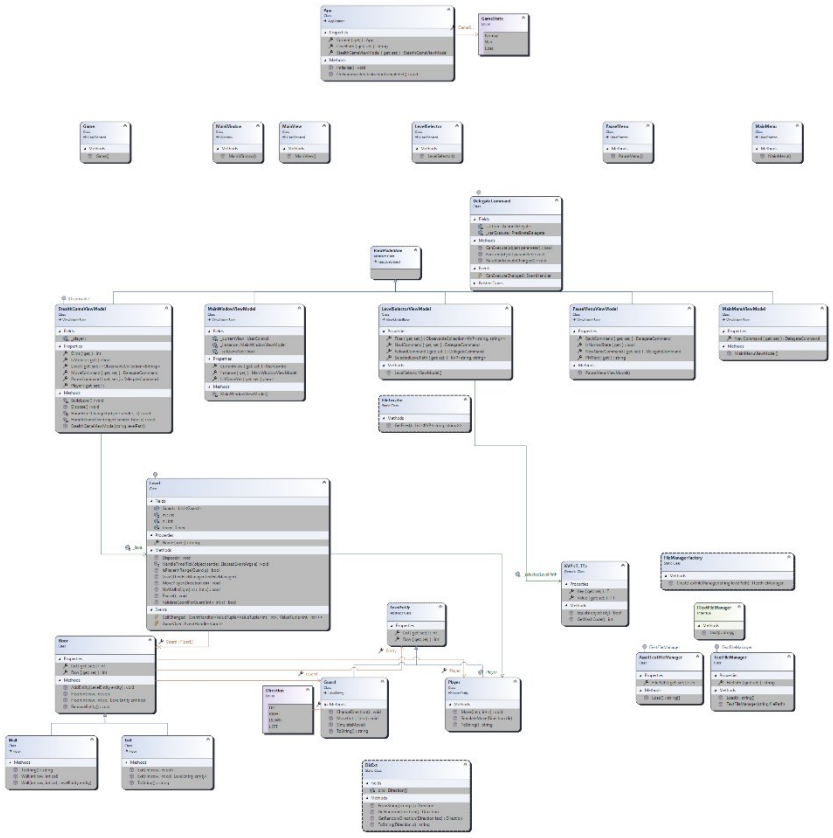
- Programszerkezet:
  - A programot háromrétegű architektúrában valósíjuk meg. A megjeleníté a View, a modell a Model, míg a perzisztencia a Persistence névtérben helyezkedik el. A program csomagszerkezete a 2. ábrán látható.
- Perzisztencia
  - Az adatkezelés feladata a szövegfájlban tárolt adatok betöltése és a szövegfájlok megkeresése
  - Az alap pályafájlok a Resources mappában vannak, viszont a program képes a %userprofile%\Documents\My Games\StealthGame\ mappában lévő pályafájlokat is használni. Első futáskor a program létrehozza a mappát.
  - Amennyiben a betölteni kívánt fájl nem létezik egy MessageBox tudatja ezt a felhasználóval.
- Modell
  - A pályát a Level osztály reprezentálja a benne lévő Board mátrixban
  - A Board cellái Floor típusuak, amelynek altípusa a Wall és az Exit
  - Bármilyen Floornak van egy adattagja amiben a rajta lévő LevelEntityt tárolja (Guard vagy Player), illetve lekérdezhető a rajta lévő játékos(ha van ilyen), illetve a rajta lévő őr, ha van.
  - Az időzítő Elapsed eventje után minden őr lép és megnézi látja-e a



Ábra 2: Csomagdiagram

játékost.
  - A MovePlayer metódus felel a játékos mozgatásáért és minden hívás végén megnézi, hogy a játékos Exit mezőre lépett, vagy látja-e őr.
- Nézet

- A játék indításakor egy „New Game” felítatú gombbal indíthatjuk el a pályaválasztót
- A pályaválasztó lista, ami dinamikusan van feltöltve a perzisztencia réteg által lekérdezett pályafájlok neveivel.
- A pályát kiválasztva egy rács jelenik meg, ahol a kék a játékos, piros az őr, zőld a kijárat, fekete a fal és a padlók világosszürkék. Ezr a renderLevel() metódus jeleníti meg.
- A „Pause menu”-ben gombok segítségével kezdhetünk új játékot vagy térhetünk vissza az elkezdett pályához.



Ábra 3: Osztálydiagram

Tesztelés

- A modell a StealthGameTest névtérben lévő osztályokban lett tesztelve
- A LevelTest osztály a pálya funkcióit teszteli
    - TestCtor: A konstruktort teszteli, hogy a Level osztály megfelelően inicializálja a játékteret. Ellenőrzi, hogy a játékos, az őrök, a falak és a kijárat a megfelelő pozíciókon találhatók, és hogy azok típusai helyesek.
    - TestValidateCoords: Ez a teszt azt vizsgálja, hogy a koordináták érvényességét ellenőrző metódusok helyesen kezelik a pálya határait, a falakat, az őröket és a kijáratot. Ellenőrzi, hogy csak megfelelő pozíciók elfogadhatók őrök számára.
    - TestPlayerInRange: A teszt azt ellenőrzi, hogy az őrök érzékelési tartományában lévő játékosok helyesen felismerhetők. Teszteli, hogy falak akadályozzák-e az érzékelést, és hogy az érzékelés dinamikusan frissül-e a játékos és az őrök mozgása során.
- A FloorTests osztály a Floor osztály metódusait teszteli.
    - TestRemE, TestAddE: triviális tesztek, az Entities listához adást, elvételt tesztelik, illetve a Unique property-t.
    - TestGetGuard, TestGetPlayer: A Guards és Player property get-jét tesztelik.
- A PlayerTests osztály a Player osztály metódusait teszteli
    - TestMove: A játékos mozgását teszteli.
    - TestSimMove: A játékos következő lépését teszteli.
- A GuardTests osztály a Guard osztály metódusait teszteli
    - TestMove: Az őr mozgását teszteli.
    - TestSimMove: Az őr következő lépését teszteli.