Dokumentáció

**Feladat**

Bombázó

Készítsünk programot, amellyel a következő játékot játszhatjuk.

Adott egy 𝑛×𝑛 mezőből álló játékpálya, amelyeken falak, illetve járható mezők helyezkednek el, valamint ellenfelek járőröznek. A játékos célja, hogy ellenfeleit bombák segítségével minél gyorsabban legyőzze.

Az ellenfelek 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.

A játékos figurája kezdetben a bal felső sarokban helyezkedik el, és vízszintesen, illetve függőlegesen mozoghat (egyesével) a pályán, de ha találkozik (egy pozíciót foglal el) valamely ellenféllel, akkor meghal.

A játékos bombát rakhat le az aktuális pozíciójára, amely rövid időn belül robban megsemmisítve a 3 sugáron belül (azaz egy 7×7-es négyzetben) található ellenfeleket (falon át is), illetve magát a játékost is, ha nem menekül onnan. A pályák méretét, illetve felépítését (falak, ellenfelek 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). Ismerje fel, ha vége a játéknak, és jelezze, győzött, vagy veszített a játékos. A program játék közben folyamatosan jelezze ki a játékidőt, valamint a felrobbantott ellenfelek számát.

**Elemzés:**

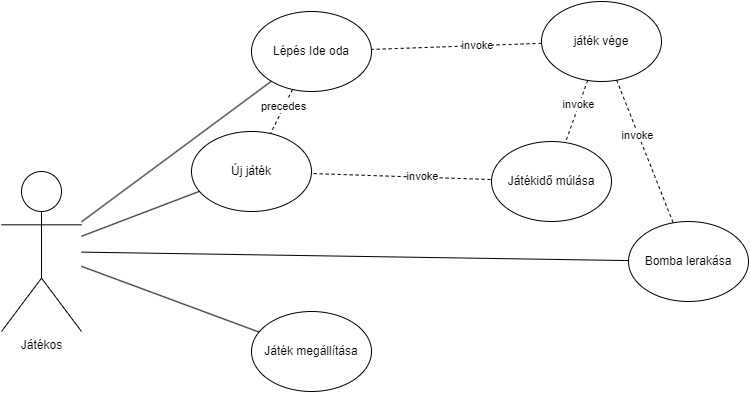
A feladatot egyablakos asztali alkalmazásként Windows Forms grafikus felülettel valósítjuk meg.

Az ablak megnyitásakor egy rövid ismertetőt kapunk magáról a játékról. A Játék fülben be tudunk tölteni egy új pályát és ki tudunk lépni. A súgó fülben további információkat olvashatunk a játékszabályról.

A pálya betöltése után tudunk mozogni és bombát lerakni. Amint a számláló elindult, az ellenfelek folyamatosan mozognak egy random irányba. Ha olyan mezővel találkoznak amire nem léphetnek, akkor egy random másik irányba kezdenek el mozogni. Ha a játékos az ellenség pozíciójára próbál lépni, elkapták, ugyan ez fordítva is, ha az ellenség a játékos pozíciójára lép, akkor a játékost elkapták.

A bombák segítségével fel tudja robbantani az ellenségeket. Miután lerakta a bombát létrejön egy bomba és játékos mező. Amint ellép onnan a Bomba marad, és a játékos mező újra megjelenik. Miután lerakta a bombát a játékos a Bomba osztály időzítője elindul, és a bomba 3 másodperc után felrobban egy 7x7es mezőben. A robbanás 1 másodpercig tart, utána az adott mezők vissza lesznek állítva.

A játékos nyer, ha megölte az összes ellenséget, veszít, ha elkapták, veszít, ha felrobbantja magát. Ezután a játék automatikusan feldob egy dialógusablakot, melyben kiírja az eredményt, majd egy másikat, hogy szeretnénk-e új játékot játszani, ha igen akkor választhatunk új pályát, de ha nem, akkor pedig kilépünk az alkalmazásból. Lehetőség van a játék megállítására is ezt megtehetjük a játék menüben, vagy a P gombbal, de ezt csak akkor tudjuk, ha nincs robbanó tárgy a mapon, azaz bomba vagy robbanás. A program folyamatosan jelzi az információkat az alján az alkalmazásnak.



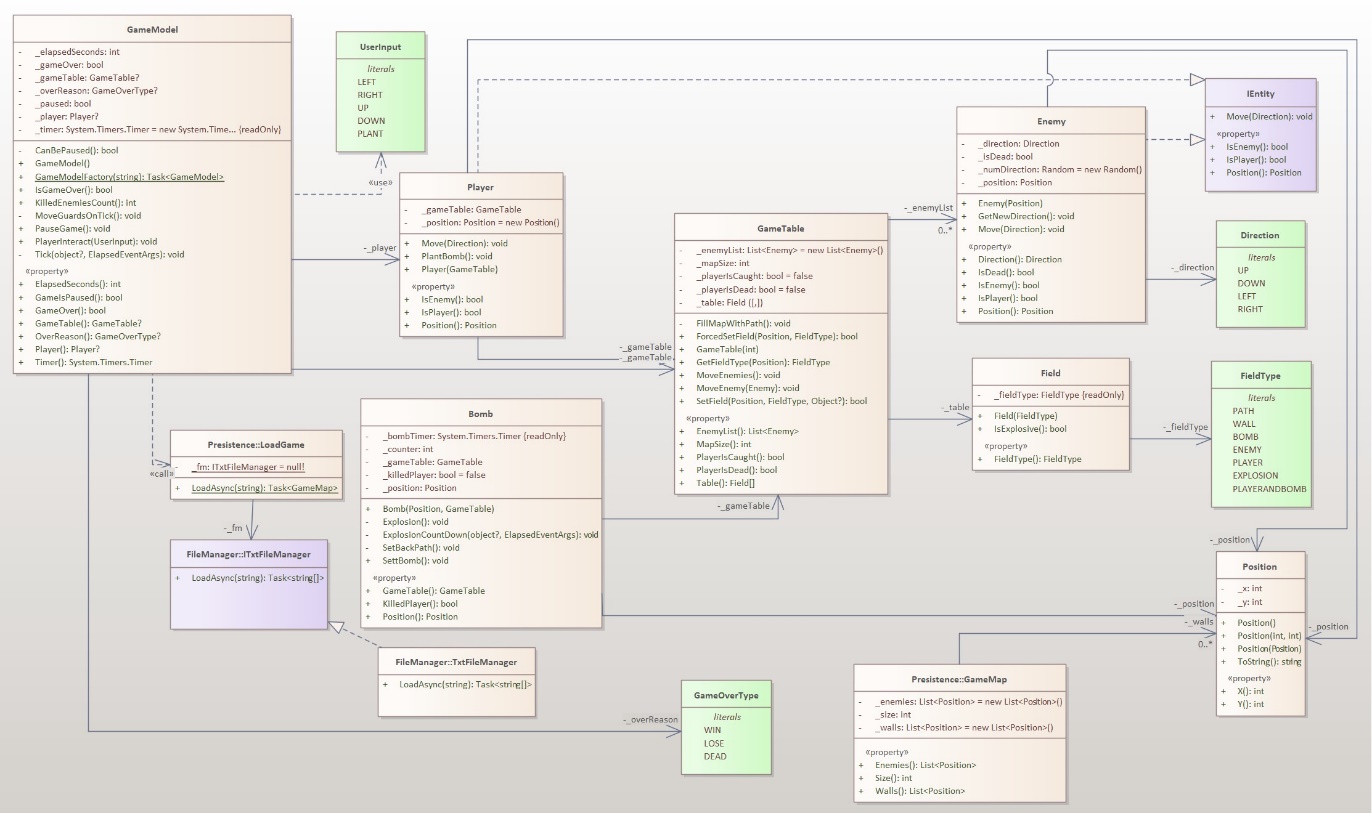
**Programszerkezet:**

* A programot háromrétegű architektúrában valósítjuk meg MVC. A megjelenítés a BombazoForm, a modell a Model, míg a perzisztencia a Persistence névtérben helyezkedik el.
* A program szerkezetét két projektre osztjuk implementációs megfontolásból: a Persistence és Model csomagokra a program felületfüggetlen projektjében, míg a View csomag a Windows Formstól függő projektjében kap helyet.
* A program alapja az, hogy egy n\*n es mátrix mezőkkel van feltöltve és a formsban egy flowlayoutpanel pedig panelekkel. A panelek különböző színűek és ezt a formsban egy metódus frissíti.

**Tervezés:**

* A Játékos és az ellenség nem láthatják egymás. A Táblán irányítja őket, a táblát pedig a model irányítja.
* A Field és a Position osztály, csak a könnyebb érthetőség olvashatóság és átláthatóság szempontjából készült.
* A Bomba osztálynak van egy időzítője, ami segítségével fel tud robbanni és tud arról, hogy ölt-e meg játékost.
* A Player osztály tud mozogni és bombát lerakni
* Az Enemy osztály szintént tud mozogni, itt egy random irányt generál a program és azt adja át a Move függvénynek.
* A GameTable osztálynak a SetField metódusa kap egy Objectet, egy FieldTypeot és egy Positiont. Ezzel a metódussal lehet állítani a táblán lévő elemeket. Ez nézi meg hogy a Player és a Enemy nem léphet falmezőre vagy játéktáblán kívülre. Itt állítjuk be, hogy a bomba az nem robbanthat fel falat. A játéktábla tudja, hogy a játékos meghalt e vagy sem, és van egy listája az ellenségekről is.
* A GameModel osztály irányítja a játékot. Van a GameModelnek egy GameModelFactory ami új játékot hoz létre. Meghívódik a LoadGame osztály LoadAsync() metódusa, ami feldolgozza a TxtFileReader osztály által beolvasott adatot. Megtörténnek a kezdeti értékadások. Van egy PlayerInteract és egy MoveGuardsOnTick metódusa és van egy IsGameOver metódusa, ami logikai értékkel tér vissza, és megnézi, hogy vége e már a játéknak.
* A Formnak is van egy időzítője. A fő oka annak, hogy ez itt van és nem a modellben az, hogy van egy RefreshMap metódus, ami folyamatosan frissíti a játéktáblán lévő elemek elhelyezkedését. Miután megnyitjuk a mapot, az ablakon különböző beállítások hajtódnak végre, és elindul az időzítő. A formsnak van egy Tick metódusa, ami 1 másodpercenként lefut. Ebben az 1 másodperc alatt lefut a CheckGameOver metódus, frissül a map és az információjelző sáv, mozog az enemy. A játékot, csak akkor lehet megállítani ha nincsen explosive mező a pályán.

**UML Diagramm:**



**Tesztelés:**

A Játék működése egységtesztekkel lett ellenőrizve a BombazoUnitTest osztályban. A tesztprojekt típusa: MS Test Project.

Tesztelésre kerültek a következő szempontok

* Létrejönnek a különböző entitások
* Létrejön a bomba és fel is tud robbanni
* Ellenség meghal a bomba robbanásban
* Játékos meghal a bomba robbanásban
* Játékost elkapják
* Játékos nyer