**1.Beadandó feladat dokumentáció**

**Készítette:**

Buchsbaum Miklós (T5K17G)

E-mail: [buchsbaum.miki2004@gmail.com](mailto:buchsbaum.miki2004@gmail.com)

**Feladat:**

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 játékot három pályával tudjuk játszani, amik egyben a nehézségi szintet is jelölik, mivel minél nagyobb pályán játszunk, annál több ellenséggel kell megbirkóznunk. A pálya méretek: 7x7 (könnyű), 10x10 (közepes), 14x14 (nehéz).
* A feladatot egyablakos asztali alkalmazásként Windows Forms grafikus felülettel valósítjuk meg.
* Az ablakban elhelyezünk egy menüt a következő menü pontokkal: File (Új játék, Kilépés), Beállítások (7x7, 10x10, 14x14). Az ablak alján megjelenítünk egy státusz sort, amely az ellenfelek számát, illetve a hátralevő időt jelzi.
* A játéktáblát a menüben kijelölt méret fogja reprezentálni. A fekete kockák lesznek a falak, amin sem játékos, sem az ellenfelek nem tudnak áthaladni. A kék pont reprezentálja a játékost, amivel a fehér területeken tudunk mozogni össze vissza. A piros háromszögek reprezentálják az ellenfeleket. Mindig arra néz a háromszög csúcsa, amerre halad az ellenfél. A lila pont reprezentálja a bombát, amit a játékos tud lerakni, hogy felrobbantsa az ellenfeleit.
* Az ellenségek megadott időközönként mozognak, a játékos a billentyűzet segítségével mozog és bombákat tesz le a pályára, amivel felrobbanthatja az ellenfeleit és saját magát is.
* A játék automatikusan feldob egy dialógus ablakot, ha a játéknak vége van (összes ellenfél fel lett robbantva, játékos felrobbantotta magát, ellenfél elkapta a játékost).
* A felhasználói esetek az 1. ábrán láthatók.

A képen szöveg, diagram, kör, képernyőkép látható

Automatikusan generált leírás

**1.ábra: Felhasználói esetek diagrammja**

**Tervezés:**

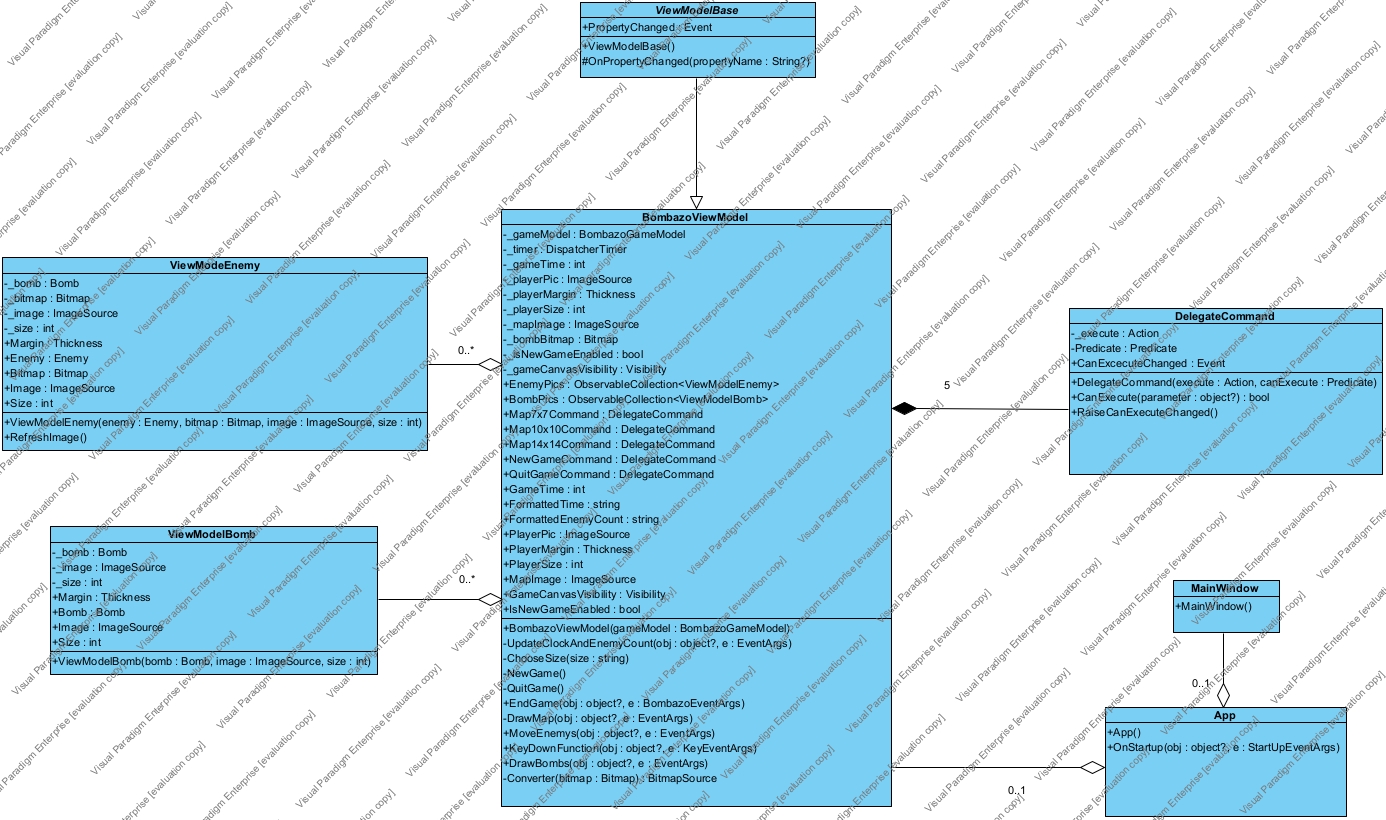
* **Programszerkezet**
  + A programot MVVM architektúrában valósítjuk meg. Tehát a View, ViewModel, Model és Persistencia névterek kerülnek létrehozásra az alkalmazáson belül.
  + A program szerkezete két projektre van osztva: a Persistence és Model csomagok a program felületfüggetlen projektjében (BombazoSajat), míg a View és ViewModel csomag a WPF függő projektjében (BombazoSajatWPF) van.
* **Perzisztencia**
  + Az adatkezelés feladat a Bombázó pályával/táblával kapcsolatos információk tárolása, valamint a betöltés biztosítása.
  + A **BombazoMap** osztály lehetővé teszi a pályaelemek tárolását és kezelését a játék során. A TileSize tulajdonság a térkép csempéinek méretét tárolja, és módosítható a SetTileSize metódus segítségével. A Map tárolja a pályaelemeket, Enemys listában vannak az ellenségek, \_bombs listában pedig a bombák. A Player változó tartalmazza a játékos objektumot. Az osztályban szerepel egy konstruktor, amely beállítja az alapértelmezett értékeket, és a Clear metódus, ami ezeket az értékeket alaphelyzetbe állítja. A LoadMap függvény a térkép betöltéséért felel, megadott elérési útról tölt be egy pályát, és a pálya elemeket beállítja a Map-en, illetve az ellenségeket az Enemys listába. Az adatok feldolgozása a betöltött pályából egy switch szerkezet segítségével történik, és az egyes karakterek alapján különböző típusú pályaelemeket hoz létre a térképen a fentiekben leírtak alapján.
  + A hosszú távú adattárolás lehetőségeit az **IBombazoDataAccess** interfész adja meg, amely lehetőséget ad a tábla betöltésére (Load)
  + Az interfészt szöveges fájl alapú adatkezelésre a **BombazoFileAccess** osztály valósítja meg.
  + A program az adatokat szöveges fájlként tudja eltárolni, amik a .txt kiterjesztést kapták. Ezeket az adatokat a programban bármikor be lehet tölteni.
  + A beállítások menüben kiválasztott pálya fogja megadni a pálya méretét. Ez egyben megadja a sorok és oszlopok számát. Ezután minden sor számokkal van feltöltve. A számok tabulátorral vannak elkülönítve egymástól. 0-2-ig fordulható elő szám a fájlban. 0 jelöli a járható utat a játékos és az ellenfelek számára, az 1 a fal, amin nem tud egyik sem áthaladni és a 2 pedig az ellenfeleket jelöli a pályán
* **Modell**
  + A modell lényegi részét a **BombazoGameModel** osztály valósítja meg, amely szabályozza a tábla tevékenységeit, valamint a játék egyéb paramétereit, úgymint az ellenfelek számát (**EnemyCount,InitialEnemyCount**) és nyomon tudjuk követni, hogy folyamatban van-e a játék vagy szüneteltetve van-e (**InGame, Paused**). **\_bombazoMap** segítségével tároljuk és kezeljük a játék térképét és elemeit. Események jelzik a játék végét és a térkép betöltését (**GameOverEvent, MapLoadedEvent**). Az események argumentuma (**BombazoEventArgs**) tárolja a győzelem állapotát. A típus lehetőséget ad pálya kiválasztására (**SelectMap**), ahol megadhatja az álltalunk kiválasztott pályát. Az idő léptetését az időbeli lépések végzésével tudjuk megtenni (**TimerTick**). Az idő léptetése által bekövetkező változásokat nyomon tudjuk követni és ellenőrizni tudjuk a játék aktuális állását, illetve kezelni a vesztés eseményt. (**CheckGameStatus, Lose**). Tudjuk kezelni a játék indítását, szüneteltetését, folytatását és befejezését (**StartGame, PauseGame, ResumeGame, EndGame**)
* **A program teljes statikus szerkezeti a 2. ábrán látható**

A képen szöveg, diagram, sor, Párhuzamos látható

Automatikusan generált leírás

**2. ábra: Az alkalmazás osztály diagrammja**

* **Nézet**
  + A nézetet a **MainWindow** osztály kezeli, amely tárolja a modell egy példányát (**\_gameModel**).
  + A játék kezdésekor a fenti menü sorból a Beállítások menü pont alatt, ki kell választani a kívánt pálya méretet és utána a File menü pont alatt pedig ki kell választani az Új Játék opciót és a játék kezdetét veszi. Bármikor ki tudunk lépni a programból, ha a File menü pont alatt a Kilépés opcióra kattintunk. Az ESC gombbal a játék szüneteltethető, illetve újbóli megnyomásával a játék folytatható. Ha játék közben új játékot szeretnék kezdeni, ugyan azt kell tenni, mint a játék kezdéskor, azonban ilyenkor az aktuális játékot automatikusan elveszíti a játékos.
* **Nézetmodell**
  + A nézetmodell megvalósításához felhasználunk egy általános utasítás (DelegateCommand), valamint egy ős változásjelző (ViewModelBase) ősosztályt.
  + A nézetmodell feladatait a **BombazoViewModel** osztály látja el, amely parancsokat biztosít az új játék kezdéséhez, játék betöltéséhez, valamint a kilépéshez. A parancsokhoz eseményeket kötünk, amelyek a parancs lefutását jelzik a vezérlőnek. A nézetmodell tárolja a modell egy hivatkozását (**\_gameModel**), de csupán információkat kér le tőle, illetve a játéknehézséget szabályozza. Direkt nem avatkozik a játék futtatásába.
  + A játék időbeli kezelését egy időzítő végzi (**\_timer**), amelyet mindig aktiválunk játék során, illetve inaktiválunk, amennyiben bizonyos menüfunkciók futnak.
  + Az ellenfelek és a bombák számára egy-egy külön mezőt biztosítunk (ViewModelEnemy, ViewModelBomb), amely eltárolja pozícióját, méretét, képét, valamint a karakterre való hivatkozást. Ezeket a mezőket egy-egy felügyelt gyűjteménybe helyezzük a nézetmodellben.
* **Környezet**
  + Az App osztály feladata az egyes rétegek példányosítása és összekötése.

****

**2. ábra: A nézet és nézet modell osztály diagrammja**

**Tesztelés:**

* A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a **BombazoTest** osztályban.
* Az alábbi tesztesetek kerültek megvalósításra:
  + **ConstructorTest:** Létrehozása után paraméterek ellenőrzése.
  + **SelectMap\_WithCorrectMap\_LoadsMap:** Pálya betöltése megfelelő fájlokkal.
  + **SelectMap\_WithNoMap\_GeneratesErrorTest:** Pálya betöltése paraméter nélkül.
  + **SelectMap\_WithIncorrectMap\_GeneratesErrorTest:** Pálya betöltése nem létező pálya esetén.
  + **TimerTickTest:** A játékbeli idő múlásának tesztelése az ellenségek mozgatásával.
  + **CheckGameStatus:** Játék végének felismerése, ha a játékos és az ellenség egy pozíciót foglal el.
  + **StartGameTest:** Futás után paramér helyes beállításának tesztelése.
  + **PauseGameTest:** Futás után paramér helyes beállításának tesztelése.
  + **ResumeGameTest:** Futás után paramér helyes beállításának tesztelése.
  + **GameEndTest:** Futás után paramér helyes beállításának tesztelése.
  + **EventsGetInvokedTest:** Események meghívásra kerülésének tesztelése