2. Beadandó feladat dokumentáció

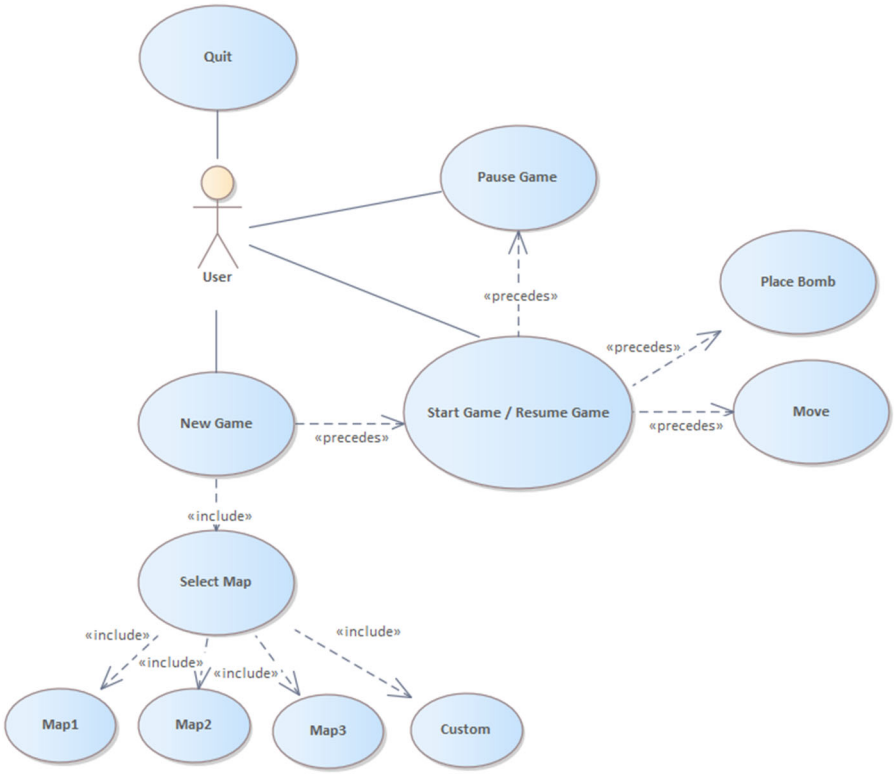
Készítette:
Raczkó Dávid Attila
Email: ycfciy@inf.elte.hu

Feladat:

Készítsünk programot, amellyel a következő játékot játszhatjuk. Adott egy $n \times n$ 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:

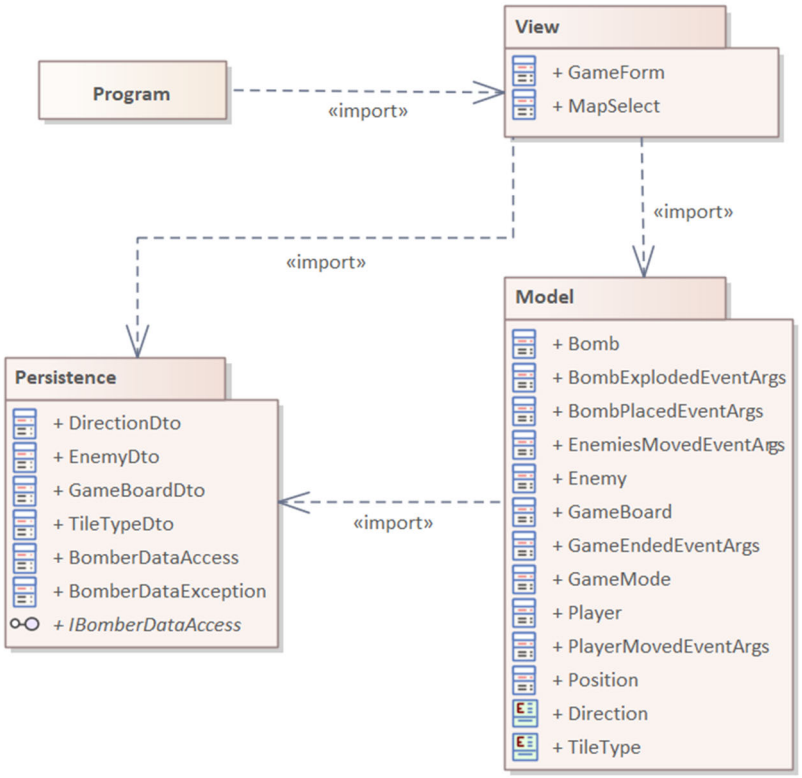
- A program indításakor egy pályaválasztó Form jelenik meg, majd, ha a pályaválasztás sikeres, megjelenik a játék ablaka.
- A feladatot kétablakos (pályaválasztó, játék) asztali alkalmazásként Windows Forms grafikus felülettel valósítjuk meg.
- A pályaválasztó ablakon lehetőség van 3 előre definiált pálya közül választani, vagy egyedi pályát betölteni fileból.
- A Játék ablakon megjelenítjük az eltelt időt, a játékos pontszámát, valamit 3 gombot, amelyek lehetőséget adnak a játékosnak.
    - új játék kezdésére: Ilyenkor megjelenik a pályaválasztó és eltűnik a játékablak
    - játék elindítására/újraindítására: Elkezd telni az idő, elkezdenek mozogni az ellenségek és a felhasználónak is lehetősége van a játékost mozgatni
    - játék megállítására
- A játéktáblát egy nxn-es (pályamérettől függő) PictureBox mátrix reprezentálja. Minden PictureBox háttérszíne a játék állapotától függően változik. A játékos kék, az ellenségek piros, az üres mezők zöld, a falak fekete, a bombák pedig szürke színnel jelennek meg a képernyőn. Egy bomba felrobbanásakor a robbanás közepétől mért 3 sugarú négyzetben a PictureBoxok színe narancssárga színű lesz, jelezve a bomba robbanását.
- Amikor egy ellenség meghal, a játékos pontot szerez, ami meg is jelenik a képernyőn. Amikor a játéknak vége (meghal az összes ellenség / az ellenségek elkapják a játékost), egy dialógusablak jelenik meg, ami tájékoztat a játék kimeneteléről.
- A felhasználói esetek az 1. ábrán láthatóak.



1. ábra: Felhasználói esetek diagramja

Tervezés:

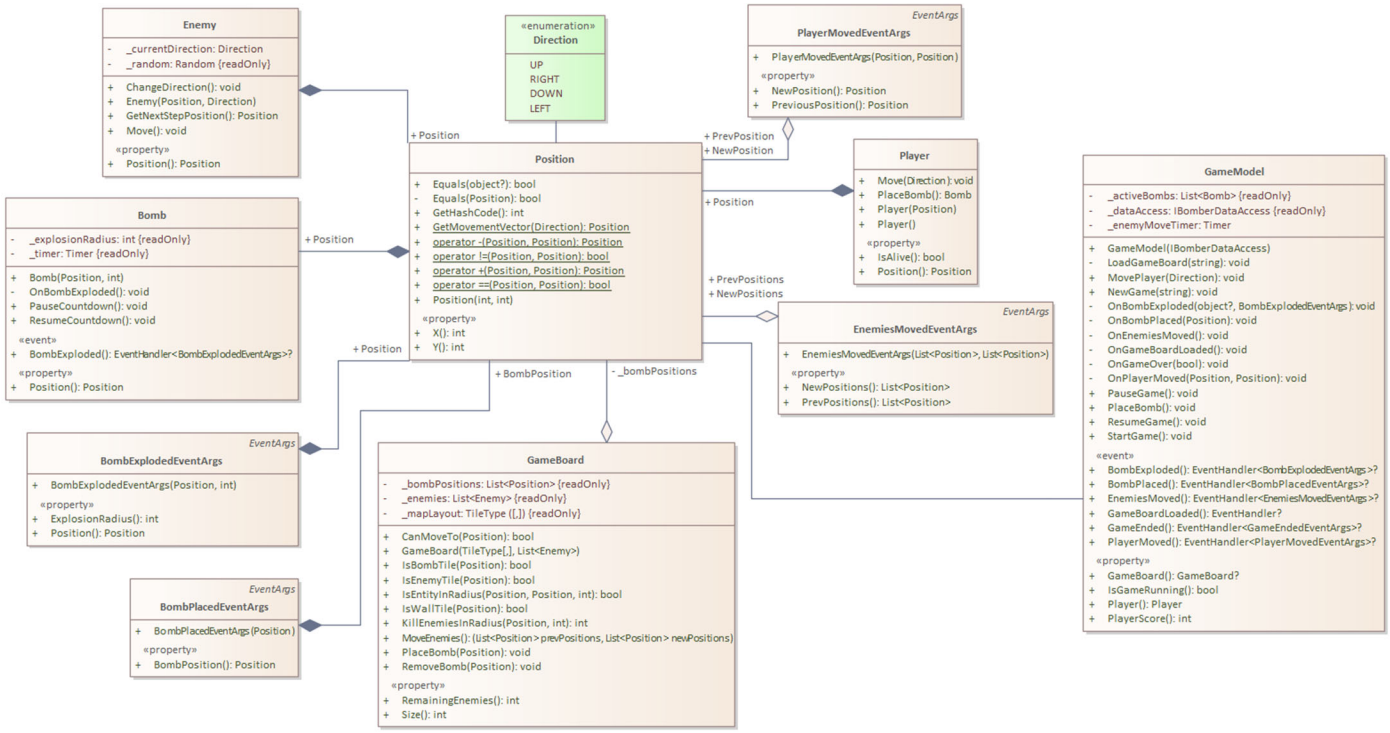
- **Programszerkezet:**
  - A programot háromrétegű architektúrában valósíjuk meg. A megjelenítés a BomberUI, a modell a BomberModel, míg a perzisztencia a BomberPersistence névtérben helyezkedik el. A program csomagszerkezete a 2. ábrán látható.
  - A program szerkezetét 3 projektre osztjuk implementációs megfontolásból: a Persistence és Model csomagok a program felületfüggetlen projektjeiben, míg a View csomag a Windows Formstól függő projektjében kap helyet.
- **Perzisztencia**
  - Az adatkezelés feladata a Játéktáblával kapcsolatos információk betöltése fileból (előre definiált / egyedi pályák).
  - A perzisztencia rétegben úgynevezett data transfer objectek (DTO-k) találhatók, amik biztosítják az adatáramlást a két réteg között anélkül, hogy a perzisztencia réteg hozzáférne a modellben található implementációs részletekhez.
  - A hosszú távú adattárolás lehetőségeit az IBomberDataAccess interfész adja meg, amely lehetőséget ad a tábla betöltésére (LoadGameBoard).
  - Az interfészt szöveges fájl alapú adatkezelésre a BomberDataAccess osztály valósítja meg. A fájlkezelés során fellépő hibákat a BomberDataException kivétel jelzi.
  - A fájl első sora megadja a tábla méretét (n) és az ellenségek számát (e). Az ezek után következő n sorban a pálya elrendezése látható, ahol „.”-tal vannak jelölve az üres mezők, a falak pedig „#” jellel. Az ezt követő e-sorban az ellenfelek pozícióinak és kezdő irányainak megadása található.

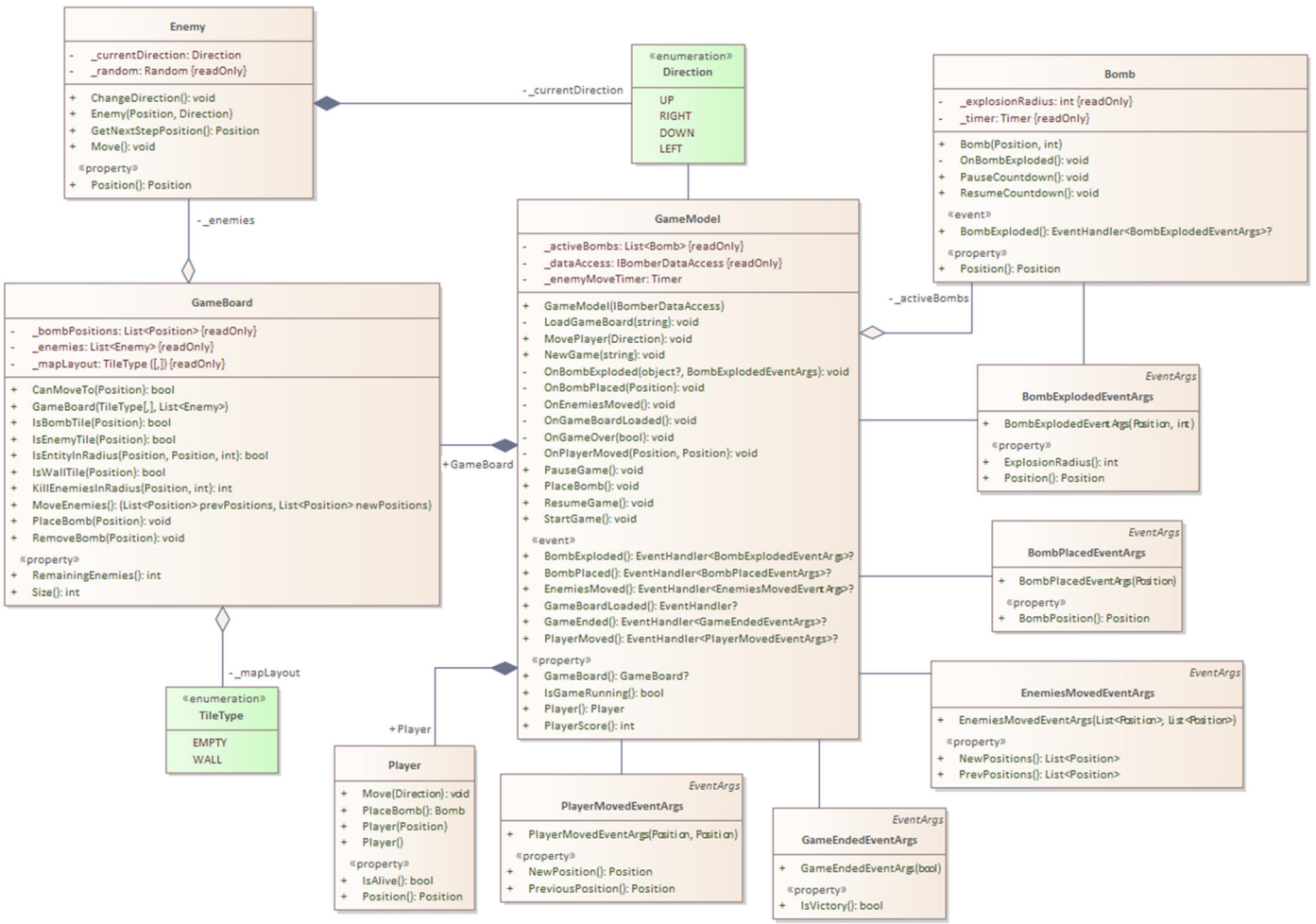


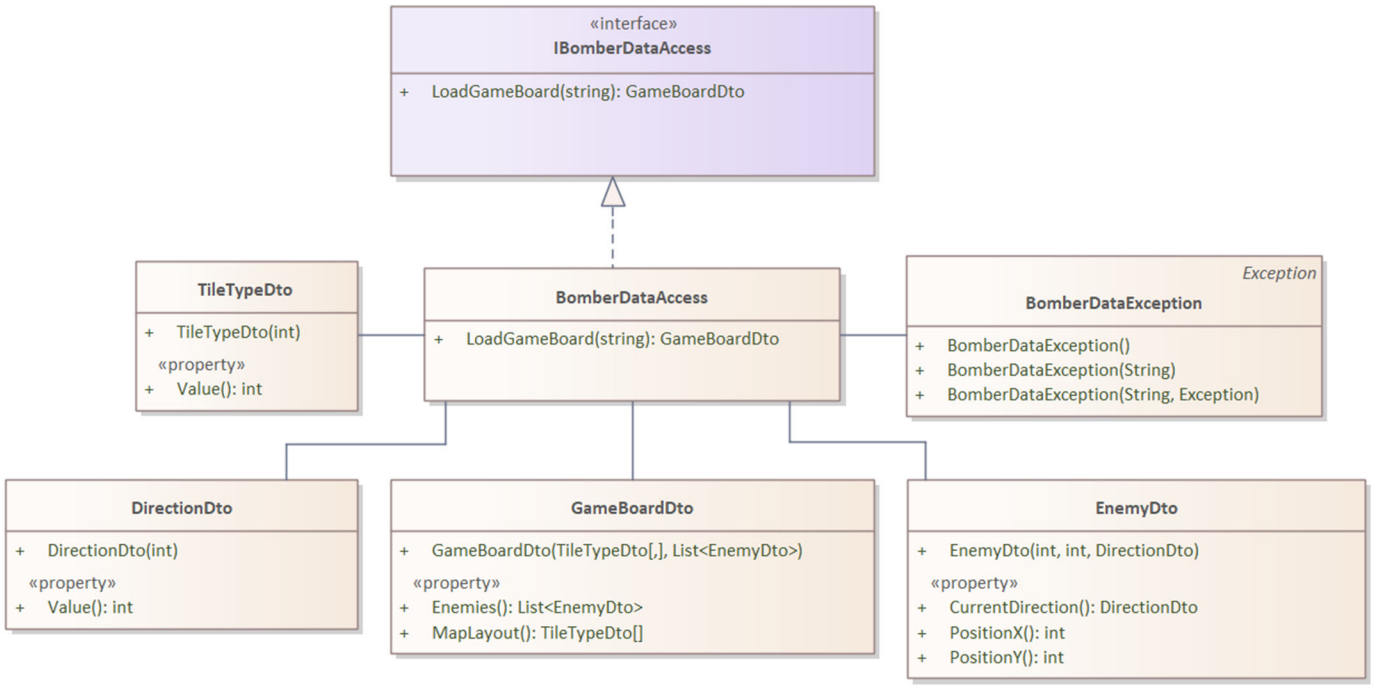
2.ábra: Az alkalmazás csomagdiagramja

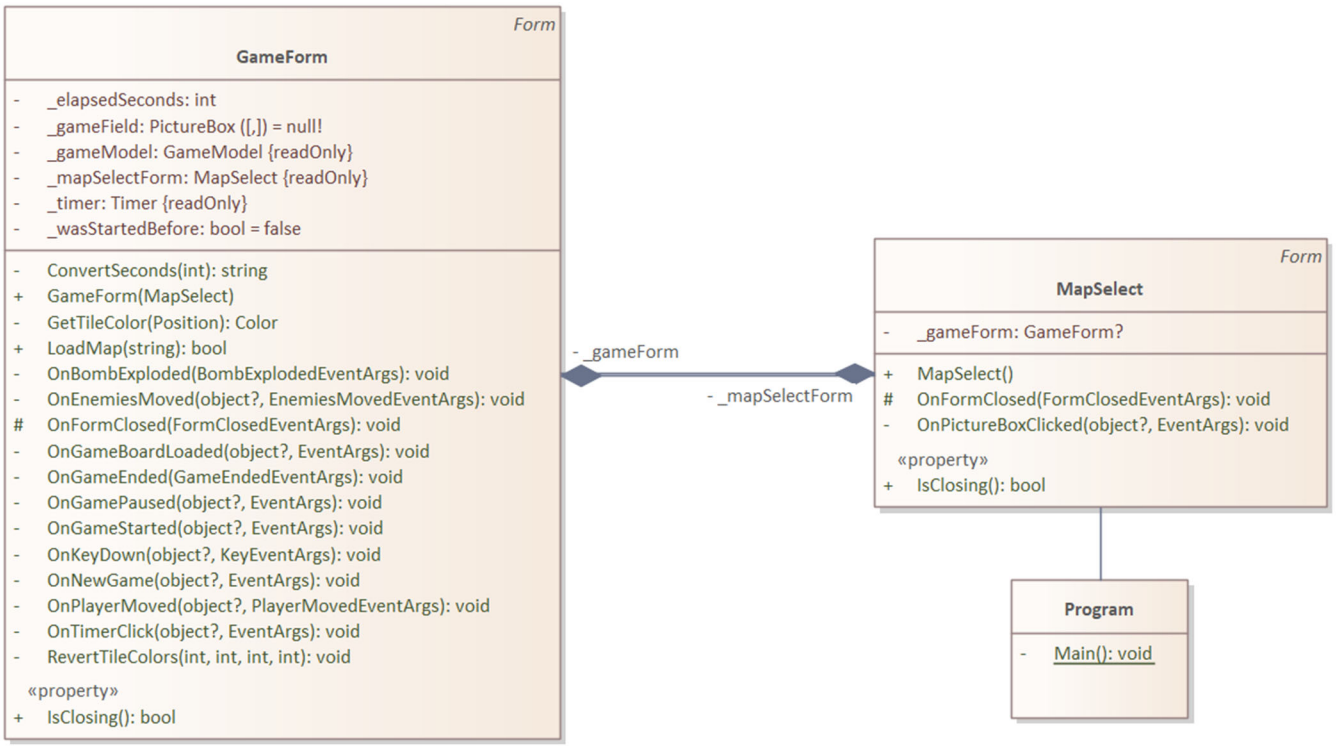
- **Modell**
  - A játékmodell lényegi részét BomberModel és a GameBoard osztályok alkotják. A GameBoard osztály tárolja a pálya adatait (elrendezés, bombák pozíciói, ellenségek) és lekérdező műveleteket biztosít. A GameModel osztály valósítja meg a játék logikáját (játékos mozgatása, bomba lerakása) és felelős az események kiváltásáért, kezeléséért.
  - Események:
    - PlayerMoved: a játékos lépéséről értesíti a nézetet, átadva a játékos előző és aktuális pozícióját.
    - EnemiesMoved: az ellenségek mozgásának hatására váltódik ki, az ellenségek korábbi és jelenlegi pozícióját adja át.
    - A BombPlaced: Kiváltódik, ha a játékos lerak egy bombát, a lerakott bomba pozíciója továbbítódik a nézet felé.
    - BombExploded: Kiváltódik, amikor egy bomba felrobban, és továbbítja a bomba helyzetét, valamint a robbanás sugarát.
    - GameEnded: Jelzi a játék végét, és átadja a kimenetelt (győzelem vagy vereség).
    - GameBoardLoaded: Akkor váltódik ki, amikor a játéktábla sikeresen betöltődött, jelezve, hogy megjeleníthető.
    - Az események száma hatékonysági okokból ilyen sok, hiszen lehetne egyesíteni őket egy UpdateUI eseménnyel is, aminek hatására a nézet újrarajzolná az egész játéktáblát. De ez helyett inkább csak azok a mezők kerülnek újrarajzolásra, amik megváltoztak, ezeket pedig az események továbbítják a nézet felé.
- **Nézet**
  - A GameForm osztály biztosítja a játék nézetét, tárolva a játékmodell (_gameModel) és a pályaválasztás formját (_mapSelectForm). A játéktábla megjelenítése egy PictureBox mátrix segítségével történik, amelyet dinamikusan hoz létre a betöltött pálya alapján.
  - A felületen található gombok lehetőséget adnak a játék indítására, szüneteltetésére és új játék indítására, miközben címkék mutatják az eltelt időt és a legyőzött ellenségek számát.
  - Az időzítést egy Timer biztosítja, amely minden másodpercben frissíti az eltelt időt a játék folyamán, míg a játékos mozgását és a bombák elhelyezését billentyűleütések irányítják (W, A, S, D a mozgáshoz, B a bombákhoz).

- A program szerkezetét a következő diagrammok szemléltetik:






«interface»
IBomberDataAccess
+ LoadGameBoard(string): GameBoardDto
TileTypeDto
+ TileTypeDto(int)
«property»
+ Value(): int
BomberDataAccess
+ LoadGameBoard(string): GameBoardDto
Exception
BomberDataException
+ BomberDataException()
+ BomberDataException(String)
+ BomberDataException(String, Exception)
DirectionDto
+ DirectionDto(int)
«property»
+ Value(): int
GameBoardDto
+ GameBoardDto(TileTypeDto[,], List<EnemyDto>)
«property»
+ Enemies(): List<EnemyDto>
+ MapLayout(): TileTypeDto[]
EnemyDto
+ EnemyDto(int, int, DirectionDto)
«property»
+ CurrentDirection(): DirectionDto
+ PositionX(): int
+ PositionY(): int


Form
GameForm
- _elapsedSeconds: int
- _gameField: PictureBox ([,]) = null!
- _gameModel: GameModel {readOnly}
- _mapSelectForm: MapSelect {readOnly}
- _timer: Timer {readOnly}
- _wasStartedBefore: bool = false
- ConvertSeconds(int): string
+ GameForm(MapSelect)
- GetTileColor(Position): Color
+ LoadMap(string): bool
- OnBombExploded(BombExplodedEventArgs): void
- OnEnemiesMoved(object?, EnemiesMovedEventArgs): void
OnFormClosed(FormClosedEventArgs): void
- OnGameBoardLoaded(object?, EventArgs): void
- OnGameEnded(GameEndedEventArgs): void
- OnGamePaused(object?, EventArgs): void
- OnGameStarted(object?, EventArgs): void
- OnKeyDown(object?, KeyEventArgs): void
- OnNewGame(object?, EventArgs): void
- OnPlayerMoved(object?, PlayerMovedEventArgs): void
- OnTimerClick(object?, EventArgs): void
- RevertTileColors(int, int, int, int): void
«property»
+ IsClosing(): bool
- _gameForm
- _mapSelectForm
Form
MapSelect
- _gameForm: GameForm?
+ MapSelect()
# OnFormClosed(FormClosedEventArgs): void
- OnPictureBoxClicked(object?, EventArgs): void
«property»
+ IsClosing(): bool
Program
- Main(): void

Tesztelés:

- A teszteket a GameModelTest osztályban hajtjuk végre, mockolt adatkezelővel (Moq), hogy egy előre beállított játéktáblán futtathassuk a teszteket. Az alábbiakban részletezzük a tesztelési folyamatokat:
- TestNewGame:
  - Ebben a tesztben ellenőrizzük, hogy egy új játék indításakor helyesen inicializálódik-e a játéktábla.
    - Megfelelő a pálya mérete?
    - Az ellenségek helyes pozíción lettek inicializálva?
    - Megfelelő a pálya elrendezése?
    - Megfelelő számú ellenség került le a pályára?
  - Emellett ellenőrizzük a játékos kezdőpozícióját (0, 0), azt, hogy él-e, és a játék futási státuszát (a játék eleinte nem fut).
- TestPauseResumeStart:
  - Ebben a tesztben azt ellenőrizzük, hogyan viselkedik a játék szüneteltetése és újraindítása.
  - A játék elindítása előtt a szünet és folytatás funkcióknak nincs hatása, a játék futása nem indul meg.
  - Miután elindítjuk a játékot, a szüneteltetés leállítja azt, és a folytatás újraindítja.
- TestPlayerMovement:
  - A játékos mozgását teszteljük, ellenőrizve, hogy a játék indulása előtt a játékos nem tud mozogni.
  - Miután elindítjuk a játékot, megvizsgáljuk, hogy a játékos mozgásai helyesek-e: nem léphet ki a pálya szélén, és falakon sem tud áthaladni. Pl. a fal előtt a játékos megáll, és nem tud továbbmenni.
- TestBombPlacement:
  - Ebben a tesztben a bomba elhelyezését vizsgáljuk. Kezdetben, mivel a játék nem fut, nem tudunk bombát lerakni.
  - A játék elindítása után a bomba sikeresen lehelyezhető, és eseményt vált ki, amelyet a tesztben ellenőrzünk. A bombát a játéktábla megfelelő pozíciójára kerül lehelyezésre.
- TestBombExplosion:
  - A bomba robbanását teszteljük. Miután a játékos lerak egy bombát, a robbanás időzítése aktiválódik, és néhány másodpercen belül felrobban.
  - A robbanás eseményt vált ki, amelyet szintén ellenőrzünk. A robbanás után a bombának el kell tűnnie a játéktábláról, és a robbanásnak el kell pusztítania a közelben lévő ellenségeket.
- TestEnemyMovement:
  - Az ellenségek mozgását vizsgáljuk. Elindítjuk a játékot, és várunk, hogy az ellenségek mozgásba lendüljenek.
  - Ellenőrizzük, hogy az ellenségek valóban elmozdulnak-e az előre meghatározott irányukba, és eseményt váltanak-e ki, amely a mozgásukra vonatkozik.