1. Beadandó feladat dokumentáció

Készítette:

Amamou Martin

W3Q74H

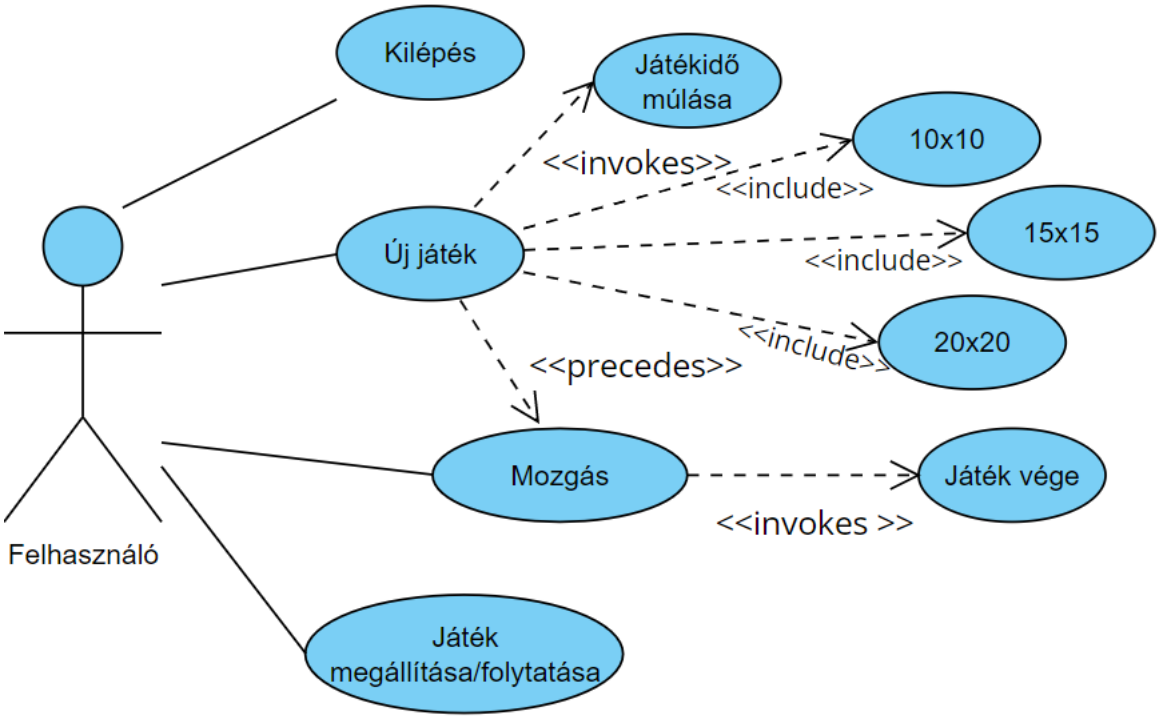
E-mail: amamoumartin@gmail.com

Feladat:

Készítsünk programot, amellyel a klasszikus kígyó játékot játszhatjuk. Adott egy $n \times n$ elemből álló játékpálya, amelyben akadályok (falak) találhatóak. A játékos egy kezdetben 5 hosszú kígyóval indul a képernyő közepén, amely vízszintesen, illetve függőlegesen halad rögzített időközönként a legutoljára beállított irányba. A kígyóval elfordulhatunk balra, illetve jobbra. A pályán véletlenszerű pozícióban mindig megjelenik egy tojás, amelyet a kígyóval meg kell etetni. Minden etetéssel eggyel nagyobb lesz a kígyó. A játék célja, hogy a kígyó minél tovább elkerülje az ütközést az akadályokkal, a pálya szélével, illetve saját magával. A pályák méretét, illetve felépítését (falak helyzete) 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 mozog a kígyó). Továbbá ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, hány tojást sikerült elfogyasztania a játékosnak.

Elemzés:

- A játékot három fajta pályamérettel játszhatjuk: 10x10(6 fal), 15x15(12 fal), 20x20(18 fal)-as táblán. A program indításkor a 15x15-ös pálya töltődik be.
- A feladatot egyablakos asztali alkalmazásként Windows Forms grafikus felülettel valósítjuk meg.
- Az ablakban elhelyezünk egy menüt a következő menüpontokkal: File (New Game(10x10,15x15,20x20), Load Game, Kilépés), továbbá egy, az aktuális pontot mutató menüponttal, illetve egy súgó gombbal.
- A program indításakor a játékmentet áll, a játékot az ENTER billentyűvel lehet szüneteltetni/újra elindítani. A játékban a kígyót a nyíl billentyűkkel, vagy a W A S D billentyűkkel lehet irányítani, ahogy az a felhasználónak kényelmesebb. A fel nyíl/W lenyomásakor a kígyó felfele, a bal nyíl/A lenyomásakor balra, a le nyíl/S lenyomásakor lefele, végül a jobb nyíl/D lenyomásakor a kígyó jobbra fog menni. A kígyó kezdetben 0.4 másodpercenként mozog egy egységnyit előre a táblán, azonban minden 10. elfogyasztott tojás után egy tizedmásodperccel gyorsul.

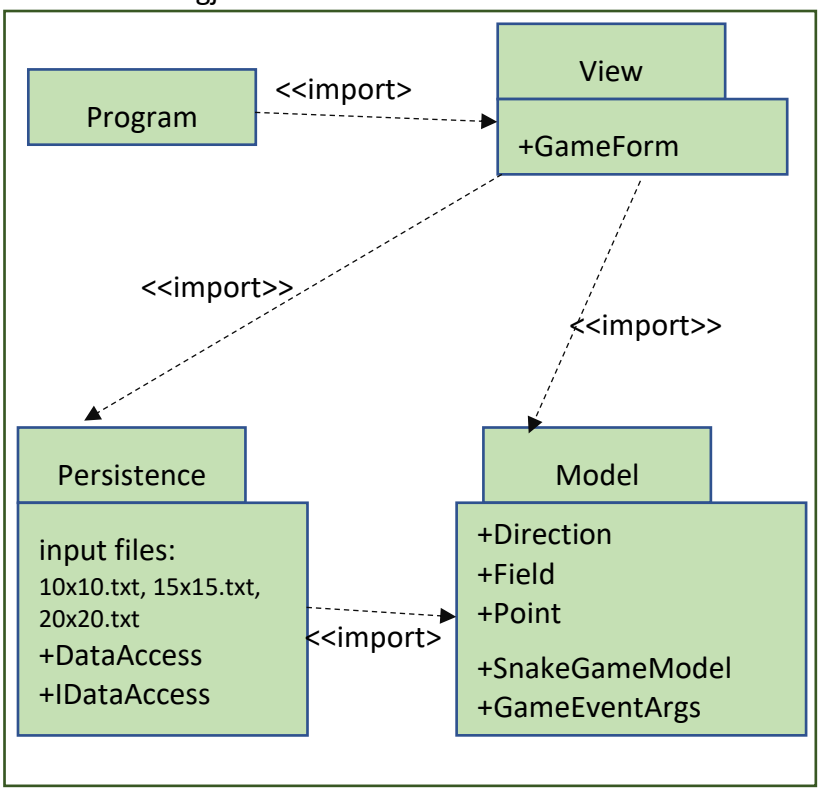


1. ábra: Felhasználói eset diagram

Tervezés

Programszerkezet:

A programot háromrétegű architektúrában valósítjuk meg. A modell a `Game.SnakeGame.SnakeGameModel`, a perzisztencia a `Game.SnakeGame.SnakeGamePersistence`, továbbá a megjelenítés a `Game.SnakeGame.View` névtérben helyezkedik el.



Modell:

A modell fő részét a SnakeGameModel valósítja meg, a tábla jelentősebb tevékenységeit kezeli, szabályozza a paramétereket: táblaméret(**TableSize**), pontszám(**Score**), irány(**Direction**),sebesség(**Speed**),stb.

Lehetőséget ad új játék kezdésére (**StartNewGame**)

A modell példányosításkor az adatkezelést megkapja, ennek segítségével átveszi az inputfájl tartalmát, ez alapján inicializálja a táblát(**ParseDataAccess**)

Tartalmazza a játéktáblát, mely egy játékmező(**Field**) felsorolási típusból álló mátrix.

Létrehozza az elemeket(tábla, kígyó, tojás, falak) (**GenerateTable**)

Továbbá a kígyó reprezentációját(pontokból álló lista), és folyamatait: mozgását(**MoveSnake**), étkezését(**Eat**), elpusztulását(**Die**) implementálja, emellett az ezeket biztosító ellenőrző- és segédfüggvényeket tárolja.

Eseménykezelés:

- A játékállapot változásáról a ScoreWriter (aktuális pont kiírása) és SpeedChange (sebesség változása) események tájékoztatnak.
- A tábla kirajzolásához a model a DataToDraw eseményt használja.
- A játék végét az OnGameOver esemény jelzi.
- Az ezekhez tartozó adatokat az események argumentuma (**GameEventArgs**) tárolja.

Persistence:

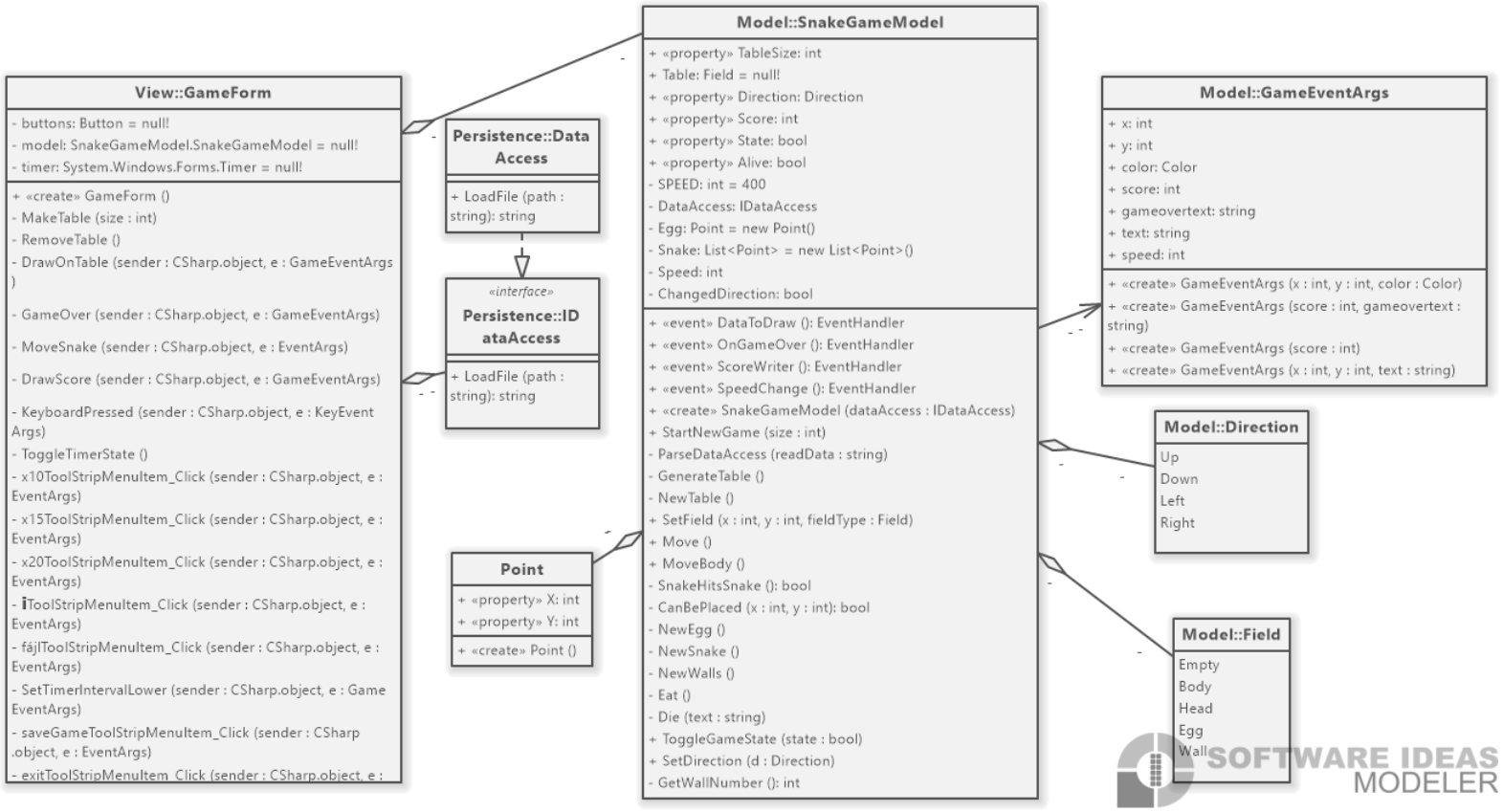
A játék ezen része a megadott pályák beolvasásával foglalkozik, melyet a modell megkap példányosításkor, hogy beállítsa a tábla értékeit. Az interfészt szöveges fájlkezelésre a DataAccess osztály valósítja meg, ez az IDataAccess-ből szárzmazik. Lehetőséget ad egy paraméteren keresztül megadott útvonalon egy fájlt beolvasni + adatait eltárolni.

A programhoz három darab előre elkészített, nem változtatható szöveges fájl kapcsolódik, melyek a 3 különböző pálya adatait tartalmazzák.

A fájl n sorból, n oszlopból áll, ahol n a tábla mérete. Csupán 0,1,2,3,4-eket tartalmaz, a játékmező típusától függően:

0- üres mező
1- a kígyó teste
2- a kígyó feje
3- fal
4- tojás

View:
A nézetet a GameForm osztály biztosítja, amely tárolja a modell egy példányát(**model**).
A játéktáblát egy dinamikusan létrehozott gombokból álló mátrix(**buttons**) reprezentálja. A felületen létrehozzuk a megfelelő menüpontokat, dialógusablakokat, és a kapcsolódó eseménykezelőket. A játéktábla készítését(**MakeTable**), eltávolítását(**RemoveTable**) külön metódusok végzik.
A játék időbeli kezelését egy időzítő végzi(**timer**), amelyet mindig aktiválunk/deaktiválunk a játék során, a funkcióktól függően.



2. ábra: A program teljes statikus szerkezete

Tesztelés

A modell funkcionalitása egységtesztek segítségével lett leellenőrizve a *SnakeGameTest* osztályban, az alábbi tesztesetekkel:

- `TenBoardTestMethod, FifteenBoardTestMethod, TwentyBoardTestMethod` ellenőrzik a 10x10-es, 15x15ös, 20x20-as pályákon a tábla megfelelő mezőinek típusát.
- `EatTest`:
  azt vizsgálja, hogy a kígyó elfogyassza-e a tojást → ezáltal növekszik-e a pontszáma, és a testhossza
- `WallCollisionTest`:
  azt teszteli, hogy valóban vége lesz-e a játéknak, ha a kígyót hagyjuk falnak menni
- `BorderCollisionTest`:
  azt teszteli, hogy valóban vége lesz-e a játéknak, ha a kígyót hagyjuk a pálya szélének menni
- `SnakeCollisionTest`:
  azt teszteli, hogy valóban vége lesz-e a játéknak, ha a kígyót hagyjuk önmagának nekimenni