Eseményvezérelt Alkalmazásfejlesztés 2. Beadandó

Bittner Barnabás TC8TT8

2016. Április

Contents

1	Feladat	2
2	Programterv 2.1 Osztálydiagram	2
3	További dokumentáció	4

1 Feladat

Készítsünk programot, amellyel a következő játékot játszhatjuk. Adott egy $n\times n$ mezőből álló erdő, amelyben Maci Lacival kell piknikkosarakra vadásznunk, amelyek a játékpályán helyezkednek el. A játék célja, hogy a piknikkosarakat minél gyorsabban begyűjtsük. A játékpályán a piknikkosarak mellett akadályok (pl. fa) is elhelyezkedhetnek, amelyekre nem léphetünk. A pályán emellett vadőrök is járőröznek, akik adott időközönként lépnek egy mezőt (vízszintesen, vagy függőlegesen). A járőrözés során egy megadott irányba haladnak egészen addig, amíg akadályba (vagy a pálya szélébe) nem ütköznek, ekkor megfordulnak, és visszafelé haladnak (tehát folyamatosan egy vonalban járőröznek). A vadőr járőrözés közben a vele szomszédos mezőket látja (átlósan is, azaz egy 3×3 -as négyzetet). A játékos kezdetben a bal felső sarokban helyezkedik el, és vízszintesen, illetve függőlegesen mozoghat (egyesével) a pályán, a piknikkosárra való rálépéssel pedig felveheti azt. Ha Maci Lacit meglátja valamelyik vadőr, akkor a játékos veszít.

A pályák méretét, illetve felépítését (piknikkosarak, akadályok, vadőrök kezdőpozíciója) tárolhatjuk fájlban, vagy létrehozhatjuk véletlenszerűen (előre rögzített paraméterek mellett). 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 megszerzett piknikkosarak számát.

2 Programtery

A feladat megoldásához a Qt keretrendszert fogjuk használni. A játék terve, hogy készítünk egy $n \times n$ -es táblát, amelyen reprezentáljuk az objektumokat, majd a megadott időközönként frissítjük a vadőröket illetve, ha a játékos lép, akkor őt is, hogy a játék szabályainak eleget tegyen.

2.1 Osztálydiagram

class EVA2_Class QObject EVA_2_Beadando MagicGameImpl boardCont: BoardContainer_t elapsedT: int BoardContainer_t: using = std::vector<std... game: std::unique_ptr<BoardRepresentation> impl: std::unique_ptr<Implementation::MagicGameImpl> hunterRefreshTimer: QTimer ui: Ui::EVA_Bead_2Class isInPausedState: bool secondsTimer: QTimer createNewGame(): void drawObjectsToGrid(): void elapsedTime(int): void EVA_2_Beadando(QWidget*) endGame(int): void ~EVA 2 Beadando() getBoardRepr(): BoardRepresentation& {query} gameEnded(int): void MagicGameImpl(QObject*) initGameBoard(): void ~MagicGameImpl() # keyPressEvent(QKeyEvent*): void movePlayer(Implementation::direction_t): void newGameInit(): void newGame(QString): void openNewMap(): void newGameStarted(): void playerMoveRequest(Implementation::direction_t): void pauseGame(): void startNewGame(QString): void sendElapsedTime(): void updateElapsedTimeLabel(int): void sendFoodToEat(int): void updateFoodToEat(int): void updatedHunters(): void updateGameBoard(): void updatedPlayer(): void updateHunters(): void «enumeration» gridLabel «enume... «struct» player_t direction_t BoardRepresentation gridBorderStyle: QString = "border: 1px so... {readOnly} NoPlayer UP boardSize: int $gridLabel(QWidget^*, Implementation::player_t)$ MaciLaci **DOWN** foodToEat: int setIcon(Implementation::player_t): void Tree LEFT gameBoard: gameBoard_t Hunter **RIGHT** hunterNumber: int InHunterRange hunterPos: QVector<hunterIndex_t> Food playerPos: playerIndex_t

Figure 1: A feladat osztálydiagramja

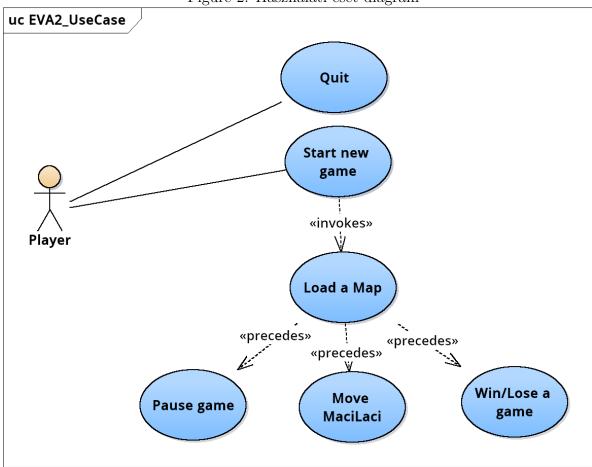


Figure 2: Használati eset diagram

3 További dokumentáció

A programot mélységében jobban leíró, automatikusan generált dokumentáció megtalálható itt: Doxygen dokumentáció