EVA 1.beadandó

Hartyányi Kevin
hartyanyi.kevin@gmail.com
Neptun kód: C0S0RJ

2018.10.10

Contents

1 Feladat

Készítsünk programot, amellyel a következő kétszemélyes játékot lehet játszani. Adott egy *n x n* mezőből álló tábla, ahol a két játékos bábúi egymással szemben helyezkednek el, két sorban (pont, mint egy sakktáblán, így mindkét játékos *2n* bábuval rendelkezik, ám mindegyik bábu ugyanolyan típusú). A játékos bábúival csak előre léphet egyenesen, vagy átlósan egy mezőt (azaz oldalra, és hátra felé nem léphet), és hasonlóan ütheti a másik játékos bábúját előre átlósan (egyenesen nem támadhat). Az a játékos győz, aki először átér a játéktábla másik végére egy bábuval.
A program biztosítson lehetőséget új játék kezdésére a táblaméret megadásával *6 x 6, 8 x 8, 10 x 10*, és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, melyik játékos győzött, majd automatikusan kezdjen új játékot.

2 Tervezés

2.1 Elemek

Elössző egy grafikus felületen bekérjük a felhasználótól a tábla méreteit. A feladat *n x n*-es táblát említ, azonban ezt kiterjesztjük *n x m*-esre, hogy érdekesebb pályákat is létrehozhassunk. A méret bekérésénél figyelünk rá, hogy csak megfelelő értékeket fogadjunk el. Ezután egy új felületen (fő felület) létrehozzuk a kért *n x m*-es nyomógombokból álló táblát. A nyomogómb alatt itt most egy egyedi osztályt értünk, ami a nyomógombon kívűl magában foglalja, hogy az melyik játékoshoz tartozik ez alapján a nyomógombot megfelelő színűre szinezzük, továbbá tárolja, hogy melyik pozícióban helyezkedik el a táblán. A fő felületen helyet kap egy menü sor, ahonnan új játékot indíthatunk, vagy kiléphetünk az alkalmazásból.

2.2 A játék menete

A játék körökre osztott, ahol minden lépésben az adott játékos választ egyet az elérhető bábúi közül és lép velük. A játékos csak arra a nyomógombra tud rákatintani, amivel tud is lépni az adott körben. Miután rákatintott a lehetséges lépéseket kiszínezük figyelve rá, hogy a lépés üres mezőt vagy ellenséges bábut tartalmaz, hogy könnyeben láthatóak legyenek a lehetőségek.

Ha valamely játékos a bábujával átért a másik oldalra vagy elfogytak a saját bábúi, akkor a játék véget ér. Ekkor egy új ablakot jelenítünk meg, ahol értesítjük a játékosokat, hogy véget ért a játék és statisztikai adatokat jelenítünk meg nekik az adott játékról, továbbá lehetőséget biztosítunk, hogy új játékot indíthassanak vagy kilépjenek a játékból.

3 Megvalósítás

3.1 Fontosabb osztályok és változók

Az alkalmazás indításakor egy *Create* osztályt hozunk létre melyet a QWidget-ből származtatunk, mely a játék bezárásáig aktív, csupán láthatóságát változtatjuk. Itt a bekért adatok után létrehozzuk a játék fő magját képező *Game* osztályt. Ennek legfontosabb eleme a táblát reprezentáló *QVector < QVector < Warrior* >>* vector. A *Warrior* egy speciális osztály melyet a QPushbutton-ból származtatunk. Mivel a játék körökre osztott, ezért minden körben leellenőrizzük, hogy a játék végetért-e a GameOver fügvényel, ha ez hamisat ad, akkor elérhetővé tesszük az adott játékos bábút. Egy kör akkor ér véget mikor az adott játékos befejezte a lépését.

3.2 Eseményvezérlés

Osztályok

1. *Create*
   - Slots
     (a) SetTable
         – **Signal** a **Battle** gombra kattintás
         – **Action** *Game* osztály létrehozása és *Create* hide-olása

Table 1: Használati esetek

Felhasználói eset	Leírás
Alkalmazás indítása	GIVEN: az alkalmazás telepítve van WHEN: alkalmazás indítása THEN: *Create* osztály létrehozása és a felületének megjelenítése
Kilépés	GIVEN: *Create* felület WHEN: a felület ablakának lezáró ikonjára kattint THEN: alkalmazás befejezése
Battle	GIVEN: *Create* felület WHEN: a **Battle** gombra kattint THEN: a *Create* eltűnik és megjelenik a *Game* osztály felülete a kért méretű táblával
Új játék	GIVEN: *Game* felület WHEN: az új játék-ra kattint a menüsorban THEN: *Game* eltűntetése és *Create* megjelenítése
Kilépés	GIVEN: *Game* felület WHEN: a felület ablakának lezáró ikonjára kattint THEN: alkalmazás befejezése
Lépés	GIVEN: *Game* felület WHEN: az adott játékos végrehajt egy lépést valamelyik bábújával THEN: a tábla a lépésnek megfelelően átalakul
GameOver	GIVEN: *Game* felület WHEN: valamelyik játékos teljesíti a játék végéhez szükséges feltételek egyikét THEN: a *GameOver* felület megjelenítése
Kilépés	GIVEN: *Game* felület WHEN: a felület ablakának lezáró ikonjára kattint THEN: alkalmazás befejezése
Új játék	GIVEN: *Game* felület WHEN: az **új játék** gombra kattint THEN: a *Game* és *GameOver* eltűntetése és a *Create* megjelenítése

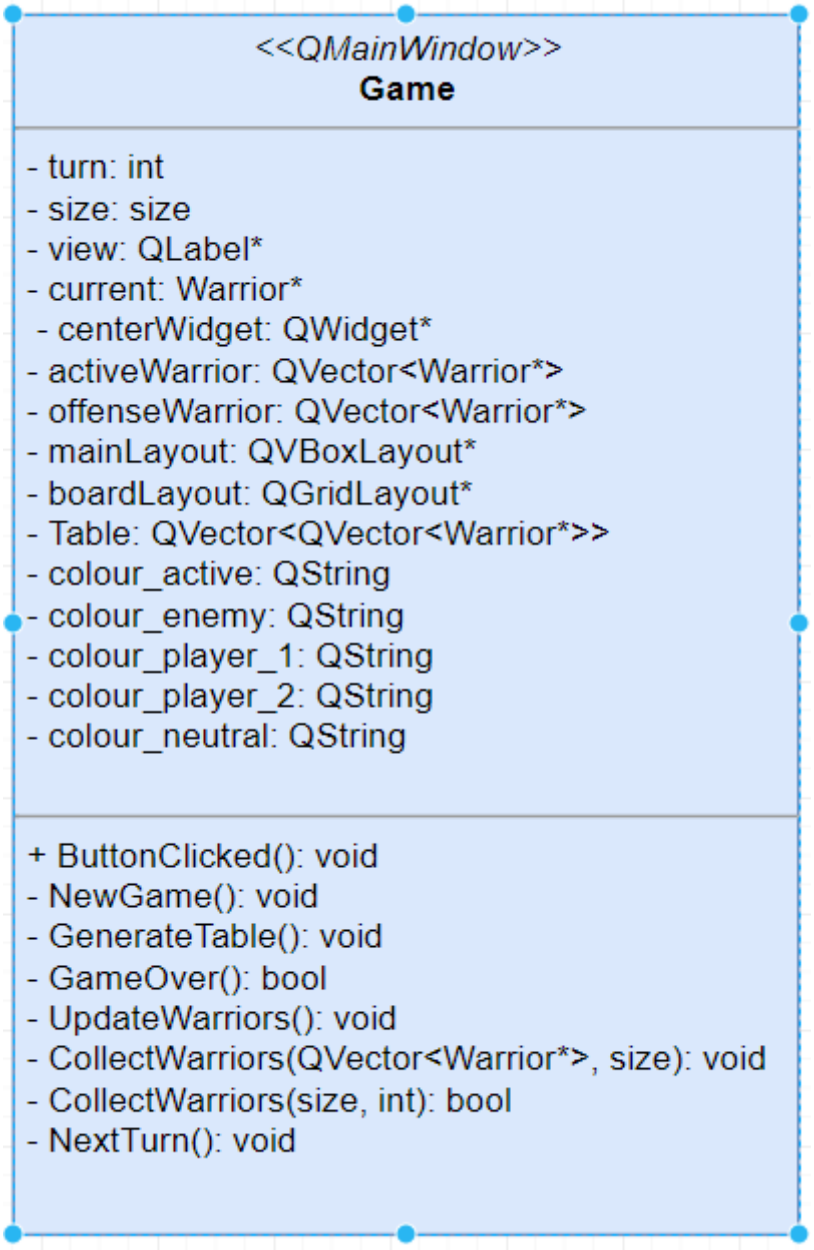

Figure 1: *Game* osztályszerkezete

(b) NewGame
- **Signal** *Game* StartNewGame-e
- **Action** *Game* törlése és *Create* unhide-olása

(c) EndGame
- **Signal** *Game* EndGame-e
- **Action** *Game* törlése és *Create* bezárása

- Signals **nincs**

2. *Warrior*
   - Slots **nincs**
   - Signals **nincs**
3. *Game*
   - Slots

(a) ButtonClicked
- **Signal** egy elérhető gombra kattintás
- **Action** a gomb kiszínezése és a lehetséges lépéseké is, a lehetséges lépéseken kívűl az összes gomb elérhetetlenné tétele

(b) on actionNew Game triggered
- **Signal** Az új játék-ra kattintás a menü sorban
- **Action** StartNewGame signal küldése

(c) on actionExit triggered
- **Signal** Az exit-re kattintás a menü sorban
- **Action** EndGame signal küldése

- Signals
  - StartNewGame
  - EndGame

4. *GameOver*

- Slots

(a) on actionNew Game triggered
- **Signal** Az új játék-ra kattintás a menü sorban
- **Action** NewGame signal küldése

(b) on actionExit triggered
- **Signal** Az exit-re kattintás a menü sorban
- **Action** EndGame signal küldése

- Signals
  - on NewGameButton clicked
  - on ExitGameButton clicked

4 Fejlesztési lehetőségek

Ebben a részben azt írjuk le, hogyan lehetne jobbá tenni a programot, itt nem térünk ki olyan lehetőségekre, mint a kodólás jobban átláthatósága, esetleg osztályok átalakítási. Pl.: Miért aktív egész végig a *Create* osztály? Nem lehetne csak mindig létrehozni mikor új játékot akarunk indítani? Ezek helyet ötleteket fogalmazunk meg és értékeljük a megvalósításuk nehézségét.

4.1 Játékosok nevet választhatnak és a kezdés véletlenszerű

Ennek megvalósítása igazán **egyszerű**. A *Create* osztályba felveszünk két QLabel-t megadva a lehetőséget névválasztásra. Ezeket a tábla alján és tetején fogjuk kijelezni a *Game* osztályban, esetleg az adott játékost aki lép mindig kiszínezük. A véletlenszerű kezdés annyiból áll, hogy egy véletlen szám generálás után vagy hozáadunk egyet a *turn* változóhoz a *Game* osztályban vagy nem. Ekkor azonban figyelni kell, hogy ennek megfelelően írjuk ki a turn-t a képernyőre.

4.2 Gépi ellenfél

Minimax A minimax algoritmussal könnyedén létrehozhatunk egy gépi játékost aki (ha jól írtuk meg) mindig a tökéletes lépést választja. Ehhez valószínűleg kicsit át kell alakítani a meglévő fügvényeket, vagy esetleg újat írni, emiatt ez inkább egy **közepes** nehézségű inplementáció. Persze felmerülhet a kérdés, hogy nem fog-e túl sok időt igénybe venni a lépés kiszámítása? Kis táblánál biztosan nem fog látszodni, azonban a maximum méretűnél *99 x 99*-es talán már látható lesz, hogy a játék fa igazán nagyra nőtt. Persze lehet tévedek. Ennek megoldására lehetne egy alpha-beta pruning-al felfejleszteni a minimaxot. Ha ez nem elég akkor átlehetne alakítani monte carlo tree keresés-re.

Monte Carlo Tree Search Ez már valószínűleg egy **nehéz** feladat, de csak azért mondom ezt, mert még sosem csináltam a gykorlatban ilyet. (Mikor egy féléve próbáltam nem sikerült megírnom egyet c++-ban, azonban mostmár elég magabiztos vagyok, hogy sikerülne.) Ez már biztosan működne bármilyen nagyságú táblára is, hiszen csak az változik, mennyire lesz pontos. Itt nem írnánk le az algoritmus müködését, de megjegyezük, hogy ez a lépés olyan szempontból is hasznos lehet, hogy több nehézségi szintű ellenfelet is könnyedén el tudunk készíteni, hiszen csak a keresési időt kell változtatni, míg a minimaxnál ez nem ilyen egyszerű.