7. gyakorlat

A hetedik gyakorlaton egy mintazárthelyi dolgoazatot oldunk meg.
Egy egyszerű játékot fogunk megvalósítani kétrétegű (modell-nézet architektúrában), mellyel átismételjük a félévben tanultakat.

Feladat: Színező

Készítsünk Qt alkalmazást a következő játékra modell/nézet architektúra segítségével. A játéktáblán mező foglal helyet, amelyek kezdetben 4 különböző színben (piros, sárga, zöld, kék) jelennek meg véletlenszerűen. A színek sorban vannak kezelve, amikor egy mezőt kijelölünk, akkor a következő színre lép (pirosról sárgára, kékről pirosra, stb.), azonban nem csak annak a mezőnek a színe, hanem annak valamennyi szomszédja változik (3x3-as területen, összesen tehát 9 mező, kivéve, ha a tábla szélén jelölünk ki mezőt). A feladat, hogy a teljes játéktáblát egyszínűvé alakítsuk, minél rövidebb idő alatt.
A játékban van egy nehezítés, hogy meghatározott lépésközönként történik egy színléptetés egy 3x3-as (véletlenszerűen kiválasztott) területen.

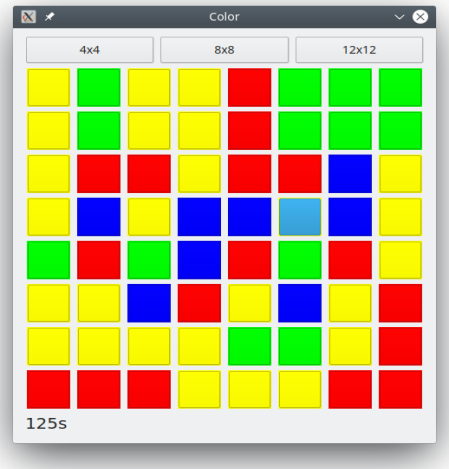
Részfeladatok:

1. **(2 pont)** A program jelenítse meg a játéktáblát, amelynek méretét (4x4, 8x8, vagy 12x12) a felhasználó adja meg. Egy mező kijelölésével az adott mező, illetve szomszédságának összes mező színét léptesse eggyel. A játék kezdetekor a táblaállás legyen olyan, amely egy teljesen piros színű táblából az előbbi lépés (véletlenszerűen kiválasztott mezőkön történő) n-szeri alkalmazásával alakult ki.

2. **(1 pont)** A program számolja a játék idejét másodpercben, és minden kijelölésre növelje ezt az értéket 10 másodperccel.

3. **(1 pont)** A program ismerje fel, ha vége a játéknak (egy színű lett a tábla), és jelenítse meg a játékos játékidejét, majd kezdjen automatikusan új játékot. Egyébként is lehessen bármikor új játékot kezdeni.

4. **(1 pont)** Legyen lehetőség a nehézség állítására három fokozatban. A könnyű fokozat megegyezik az eddigiekkel. Közepes fokozaton minden 10. kijelölés után történik egy véletlenszerű állítás, míg nehéz fokozaton minden 4. kijelölés után.



1/

Nézet

A nézeten szereplő elemeket hozzuk létre dinamikusan, a változó pályaméret miatt.
Érdemes az alkalmazás felületének beállítását függvényekbe kirendezni a jobb átláthatóság elérése érdekében. Készítsünk egy ***setupWindow()*** eljárátást, melyet majd a nézet konstruktora fog meghívni.
A ***setupWindow()*** eljáráson belül hozzuk létre a megfelelő layout-okat:

- egy horizontális layout, ahova a pályaméretet változtató gombokat helyezzük
- egy grid layout a gombrácsnak, melyet majd később helyezünk fel
- ezeket mind hozzáadjuk egy vertikális layout-hoz, hogy egymás alatt helyezkedjenek el

Induljon új játék a legfelső, horizontális layout-ra felhelyezett gombokra való kattintáskor, a gombon szereplő táblamérettel! Ezekhez hozzunk létre három függvényt, minden gombhoz egyet-egyet (Ne felejtsük el hozzárendelni a megfelelő gombokat a megfelelő eseményhez a ***setupWindow()***-ban!). Ezek a függvények jelezzenek a modell felé, annak megfelelő függvényének meghívásával, hogy induljon a játék, valamint generálják le a táblát. Ezt a két mozzanatot érdemes kettéválasztanunk egy ***newGame()***, valamint egy ***setupTable()*** eljárásokra. Ezek megírásakor figyeljünk a következőkre:

- az előző játék gombjait, ha vannak, levegyük a pályáról
- hozzunk létre egy ***buttonClicked()*** eseményt, melyet összekapcsolunk a tábla gombjaival

Érdemes egy ***refreshTable()*** függvényt is megírnunk, mely lekéri a modelltől a pálya állapotát.

Modell

A játék aktuális állapotának tárolására több lehetséges módszer is van. A tábla egy mezőjét reprezentálhatjuk többféleképpen is, akár egy char-ral, int-tel, vagy egy saját enum-mal is. Mindenesetre a tábla állapotát QVector<QVector<_tetszoleges_tipus_>> típusú objektumban tároljuk.

Írjuk meg a megfelelő gettereket a nézet számára, majd azokat az egyéb függvényeket, eljárásokat, amelyeket a nézet meghívhat, pl.: oneStep(...), newGame(...). Ezek működését a feladat szövege alapján gondoljuk meg!

2/

Modell

A játék idejét számolnunk kell. Ehhez vegyünk fel adattagként egy QTimer-t, mely inicializálását a megfelelő helyen végezzük el (érdemes az eltelt időt egy külön int-ben tárolnunk). Indítsuk új játék kezdetekor, és növeljük 10-zel a feladat szövegének megfelelő esetben!
Ahhoz, hogy a timer aktuális állapotát lássa a felhasználó a nézetben létrehozott label-en vagy kijelzőn, a modellnek egy signal-t kell küldenie a nézet felé. Ezt a signal-t a megfelelő időben ki kell váltani, azaz amikor változik az eltelt idő értéke.

Nézet

Az idő múlását jelenítsük is meg! Ehhez egészítsük ki a ***setupWindow()*** függvényünket, és a vertikális layout-hoz adjunk hozzá egy label-t, vagy lcd-kijelzőt amely jelzi a felhasználó felé az idő múlását.
Készítsünk egy eljárást (mint private slot, ***refreshTime(int)***), amely a modell signal-jára vár, és az imént felhelyezett label/kijelző állapotát frissíti.

3/

Modell

A játék végének megállapítását oldjuk meg a következő függvény megírásával: ***isGameOver() : bool***.

Nézet

Ha az ***isGameOver()*** függvény visszatérési értéke igaz, jelenítsünk meg egy MessageBox-ot.

4/

Modell

Oldjuk meg, hogy különböző nehézségi szintek legyenek a játékban! Ehhez érdemes valamilyen privát adattagban tárolni a játék nehézségi szintjét, majd egy lépés végrehajtásakor ezt figyelembe venni, és ennek függvényében változtatni a lépés implementációján.

Nézet

Új játék kezdetekor legyen lehetőség a nehézségi szint megadására! Ennek megoldására több mód van (újabb gombok felvétele, dialógusablak, stb...)