Szakdolgozat

Kiss Máté Árpád
Vámosi Béla

Debrecen
2011

Debreceni Egyetem

Informatikai Kar

Észjátékok

Témavezető:

Dr. Juhász István
egyetemi adjunktus

Készítették:

Kiss Máté Árpád
Vámosi Béla
PTI BSc

Debrecen
2011

Tartalomjegyzék

I. Bevezetés

Ez a dolgozat két programtervező informatikus hallgató munkája. A dolgozatban, egy a memória fejlesztésével foglalkozó, weboldal kerül részletes bemutatásra,valamint maguk a játék alkalmazások is.

Az emberi memória, a kor előrehaladtával romlik, és ez a romlás sajnos visszafordíthatatlan, viszont lassítható valamint a memória karbantartható. A jó memória segít abban, hogy sikeresek legyünk, az iskolában, a munkahelyen és az életben. Segít abban, hogy gyorsabban tanuljunk, és azok a személyek, akik hamar tudnak új ismereteket elsajátítani, azok hamarabb tudnak jó álláshoz jutni, illetve karrierüket is hatékonyabban tudják építeni. Ha változtatni szeretnénk életvitelünkön, akkor javítani, tökéletesíteni kell a memóriánkat. Ebben segítenek azok a játékok amiket készítettünk.

Dolgozatunk célja, hogy olyan memóriajátékokat hozzunk létre, melyek jó memóriagyakorlatoknak számítanak, és segítséget nyújtsanak a memória fitten tartásában, ezáltal csökkentve az időskori memóriaromlás előrehaladtát. A következő fejezetben részletesen tárgyaljuk az emberi memóriaszerepét, zavarait, és azt hogy ezek a rendellenességek, hogyan akadályozhatók meg bizonyos jó memóriagyakorlatok használatával és különböző modern módszerekkel.

Ezeket a játékokat megosztva fejlesztettük, a dolgozat további részeiben említett fejlesztőeszközökkel. A weboldal 4 darab játékot tartalmaz. Név szerint Képkirakó, Útvonal, 9 Kör valamint Alakzatok. Az előbbi két játékot Vámosi Béla készítette, az utóbbi kettőt pedig Kiss Máté Árpád. Továbbá a játékokat tartalmazó weboldal, közösen lett fejlesztve, beleértve a hozzá tartozó adatbázissémát (MSSQL 2008 Express), illetve az autentikációt, amit a játékokból a WCF RIA Services segítségével oldottunk meg. Valamint a játékokból, az adatbázis szerver elérése Domain Service Class segítségével történik meg.

II. Az emberi memória

A memória szerepe

Az ésszerű élet három alap képességének egyike a memória. A maradék két alapvető képesség az észlelés és a gondolkodás. Az emberi lény csaknem összes cselekedete a környezetünkből kiszűrt információk függvénye, és annak, hogy ezeket hogyan használja fel a környező világgal való kapcsolatainak rendezésére. Emberi nézőpontból a memória egyrészt a szervezetünk fenntartására és fejlesztésére alkalmas hasznos képességek, szokások, információ és tudás jelei, ugyanakkor ártalmas és káros tudást és tapasztalatokat is magában foglal. A civilizációnk és kultúránk fejlesztése és megőrzése szempontjából elengedhetetlen, hogy az ember tisztában legyen saját és leszármazottai létezésének időbeli folyamatosságával, nemcsak a múltját és jelenét, de a jövőjének terveit is észben tartva. A memória nem egy egyszerű mechanizmus hanem különböző mentális kapacitások megtestesülése. A memória egyik típusának megléte vagy hiánya nem szükségszerűen jelenti azt, hogy egy időben ezzel a többi típusú memória is létezik vagy éppen hiányzik [1].
A memóriáról szóló ezen gondolatokat Edel Tulving észt származású, kanadai neurológus fogalmazta meg [2].

A memória típusai

Az emberi memória többféle módon csoportosítható. Az egyik ilyen felosztás a következő [3]:

- **„folyékony", vagy tanulási memória**
  Ez az új dolgok megtanulásához kapcsolódik, és idős korban ez sérül először.
- **„kristályos", vagy tároló memória**
  A korábban megtanult, észlelt információk tárolódnak itt. Idős korban ez kevésbé sérül, gyengül.

Egy másik felosztás alapján a következő memóriatípusokról beszélhetünk:

- **rövidtávú memória**

  Jellemzője, hogy kicsi a kapacitása. Bizonyos források szerint 7 egységnyi információ tárolására tartják alkalmasnak. Helyileg ez a munka memória. A rövidtávú memóriában lévő információk később bekerülnek a hosszú távú memóriába.

- **hosszú távú memória**

  Itt a legrégibb élmények maradnak legtovább, ezért emlékszik az idős ember a legrégibb dolgokra, míg a most történtek hamar elfelejtődnek. A hosszú távú memóriát kémiai memóriának tekintik.

  A hosszú távú memória elemei:

  - epizodikus memória
  - szemantikus memória
  - metamemória
  - prospektív memória

A memória zavarai

Az emberi kor előrehaladtával a memória már egyre kevésbé teljesít a megfelelő szinten, így az időskori embereknél már a mindennapi élet során is problémák adódhatnak [4].

A legfőbb veszély a **demencia** jelensége, amely az időskori elbutulást jelenti. Legfőbb rizikófaktora az életkor, a 40 év feletti korosztályban indul, a 65 év felettiek 1%-át érinti, ez a kor előrehaladtával 5 évente megduplázódik, 90 évesek 40%-a demens.

Hét évig tartó kísérletsorozat bizonyította, hogy az ember agya 22 éves korában van a csúcson és 5 év múlva elindul a hanyatlás, 37 éves kortól beüt a *memória romlása* is (Neurobiology of Aging, University of Virginia kutatói, Timothy Salthouse). 22 éves csúcs után pár évig tartjuk a szintet, az első jelentősebb mentális visszaesés 27 évnél jelentkezik. Először a rövid távú memória sérül, ami nagyban megnehezíti a beteg önellátását. Felléphetnek a nyelvi kifejezőképesség és az információfeldolgozás sebességének zavarai.

Az összes demencia 50-60 %-át adja az Alzheimer-kór, 10-25%-a vascularis, 5-5%-kal jelennek meg az egyéb formák (neurodegeneratív kórképek, alkoholos, egyéb).

Jelenleg **világszerte** 115.4 millió Alzheimer-kóros ember él, évente 4.6 millió új esetet regisztrálnak. Ez azt jelenti, hogy 7 másodpercenként esik ebbe a betegségbe valaki a Földön. A fejlett országokban a 4. leggyakoribb halálozási ok [5].

Európában 1.4 millió új eset van évente, összesen 7.3 millió Alzheimer-kóros beteget tartanak nyilván.

Magyarországon 160 000 ember küzd ezzel a betegséggel. De ettől tulajdonképpen sokkal több embert érint ez a betegség és következményei, vele járó nehézségek, mert a családtagok, közvetlen hozzátartozók, barátok életét szintén nem hagyja érintetlenül.

A memória karbantartása

Epidemiológiailag igazolt, hogy a tanultabb, változatosabb, fokozottabb problémamegoldó képességet igénylő munkát végzők idősebb korban is szellemileg aktívak maradnak és csökken az esély a szellemi leépülésre, összehasonlítva az alacsonyabb iskolai végzettségűekkel, szellemileg monoton munkát végzőkkel.

Több kutatás is alátámasztja a rendszeres agytorna fontosságát a memória megőrzésében [6] [7]. Hatékony és olcsóbb mint a gondozás különböző formái.

Többféle módszer létezik a memória fitten tartására. Ilyen például **Alain S. Brown módszer**, melynek lényege, hogy a memorizálandó anyagra oda kell tudatosan figyelni, társítani szükséges valami meglévő tulajdonsághoz, gondolathoz, ezt többször meg kell ismételni, majd időről-időre fel kell frissíteni [3].

Egy másik módszer a **Neurobic**, amely egy modern módszer az agy működésének szinten tartására. A módszer az agy hálózatos szerkezetét, az impulzusok több oldalról (több érzékszerv útján) való megerősítését használja ki, építi az agykéreg különböző területei közötti kapcsolatot [8].

Ezeken a módszereken kívül számtalan memóriagyakorlat létezik. Ilyenek például a klasszikus memóriakártyák, az akasztófa-játék, a szómegjegyzés, keresztrejtvény, asszociációs játékok és még sok más.

Az időskori elbutulás a felére csökkenthető a rendszeres agytorna és megfelelő életmód mellett. A rövid távú memória fejlesztése fontos, ez az érintette a legkorábban az Alzheimer demenciában is. A gyakorlatok hatására az agy bizonyos területein belül a sejtek

közötti kapcsolatok száma növekszik. Tehát a szellemi frissesség megőrzésére irányulnak az időben elkezdett gyakorlatok. Ez pedig nem csupán az időskori elbutulás javításában játszik fontos szerepet, hanem a korábban részletezettek miatt, a korábbi életévek során is javítja a memória funkciókat, mely fontos a munkavégzésben, átképzésben stb.

III. Fejlesztői környezet

A fejlesztés során .NET 4.0 keretrendszerben dolgoztunk, Silverlight 4.0 technológiát alkalmazva [12]. A következő eszközöket vettük igénybe munkánk során:

- **Microsoft Visual Studio 2010 Expression Edition**
  Integrált fejlesztői eszköz, amely a .NET keretrendszer minden eszközét biztosítja. Segíti a tesztelési szemléletű fejlesztést, hibakereső eszközei pedig gondoskodnak az elkészült megoldások kiváló minőségéről.

- **Microsoft Expression Design 4**
  Professzionális illusztrációkészítő és grafikai tervezőeszköz, amellyel tetszetős alkotóelemeket lehet létrehozni webes és asztali alkalmazások felhasználói felületéhez.

- **Microsoft Expression Blend 4**
  Professzionális tervezőeszköz, amellyel lenyűgöző felhasználói élményt és alkalmazásokat lehet létrehozni Windows rendszerhez a .NET-keretrendszer új – a Windows megjelenítési alaprendszert is tartalmazó –, 3.0-ás verziójának használatával.

- **Deep Zoom Composer**
  Egy komplex képkezelő alkalmazás, melynek segítségével különböző felbontású képeket egy közös kollekcióba helyezhetünk, és ezt exportálva egy képként kezelhetjük a gyűjteményünket. A hálózaton való átvitel így sokkal gyorsabban megvalósul. Segítségével rendkívül látványos alkalmazásokat hozhatunk létre.

- **Microsoft SQL Server 2008 Express**
  Adatbázis kezelő rendszer, az MSSQL ingyenes verziója, ez tárolja, a játékokhoz használt adattáblákat.

IV. A memória játékok tervezési fázisa

A játékok leírása

9 kör játék leírása

A játék felületén 9 darab egyenként kattintható kör található, amelyet a játék kezdetekor az alkalmazás pirosra színez belőlük valamennyit véletlenszerűen. A játékos feladata, hogy a színezett köröket megjegyezze, erre biztosítva van 5 másodperc. Majd az idő lejárta után a program véletlenszerűen elkezdi színezni a köröket, ez természetesen nehézségi szinttől függ hogy hányszor.

Ezután a felhasználónak a legelsőnek látott pirosra színezett körökre kell kattintania, és annyi pontot kap, ahányat jól eltalált.

Alakzatok játék leírása

A játék indulásakor a játékos alakzatokat fog látni, melyek a fiókokban jelennek meg. Ezen alakzatok közül minden fiókban egy időben csak egy látszik. Miután minden fiókban látható volt egy alakzat, a nehézségi fokozatnak megfelelően, valahányszor újra megismétlődnek a képek, de mindig ugyanabban a sorrendben, ahogy először látható volt. A képek lejátszása után a felület jobb oldalán, az összes alakzat látható lesz, majd az alakzatok alatt lesz kiírva, hogy melyik az a kép, amely a játék által megjelölt fiókban látható volt. Ha a felhasználó jól emlékezett alakzatra, akkor sikeres volt a játék. Ellenkező esetben negatív visszajelzést kap.

Képkirakó játék leírása

A képkirakó játék leglényegesebb eleme egy kép, amelyet kicsi, négyzet alakú darabokra vágunk szét. Ezeket a képszeleteket összekeverjük, majd véletlen sorrendben egymás mellé rakjuk őket, ezzel egy táblát alakítunk ki. Egy képkocka üresen marad. A játékos feladata, hogy az összekevert darabokból előállítsa az eredeti képet, olyan módon, hogy a képdarabokat tologatja a táblán. A játékos nem helyezheti át bármelyik képkockát az üres területre. A tologatás művelete azt jelenti, hogy az üres hely mellett, fölött vagy alatt lévő részt átmozgatja az üresen lévő helyre. A játék akkor ér véget, ha minden apró darab az eredeti képnek megfelelő helyre kerül.

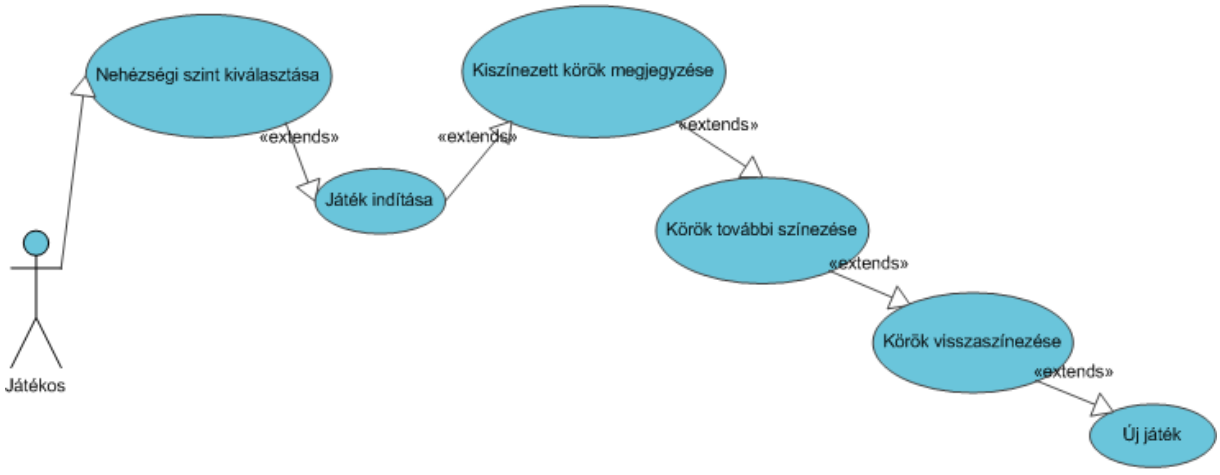
Útvonal játék leírása

Az útvonal játék a klasszikus memóriajátékok csoportjába sorolható. A link módszer jelenik meg benne, amely nagyon jó emlékezeterősítő hatással rendelkezik [7]. A módszer lényege, hogy különböző dolgok egy listáját kell megjegyezni, amely egyre bővül és végül egy „történetet” alkot, melyet a memória gyakorlatot végző személynek meg kell jegyezni. Az útvonal játék lényege, hogy egy folyamatosan bővülő ábrasorozatot (útirányokat) kell memóriánkban eltárolni, majd vissza kell tudnunk játszani az utat. A játékban a bal, a jobb és az egyenes irányok szerepelnek. A játék során szinteket kell teljesíteni a felhasználónak. Az első szinten egy ábra jelenik meg a felhasználónak. Minden egyes következő szintre lépésnél egy további iránnyal bővül a játékos által megjegyezendő képek sorozata. A számítógép véletlenszerűen kiválaszt útirányokat, – tehát előfordulhatnak azonosak is – majd visszaellenőrzi, hogy a felhasználó képes volt-e az adott sorrendben megjegyezni ezeket. Ha bármelyik lépést eltéveszti, akkor a játék véget ér, viszont az összes útirány hibátlan visszajátszása esetén a játékos nyer.

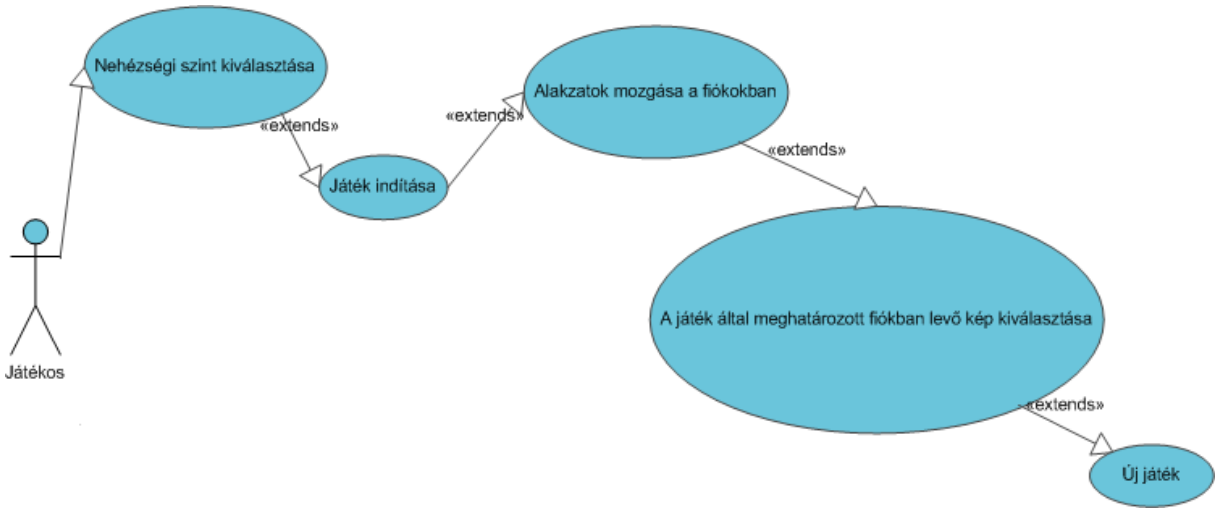
A játékok által megvalósított funkciók

A memória játék tervezési fázisában megfogalmazott funkcionális követelményeket, a játék funkcióit, működésének vázát a most következő, könnyen áttekinthető **Use Case** diagramok mutatják be.

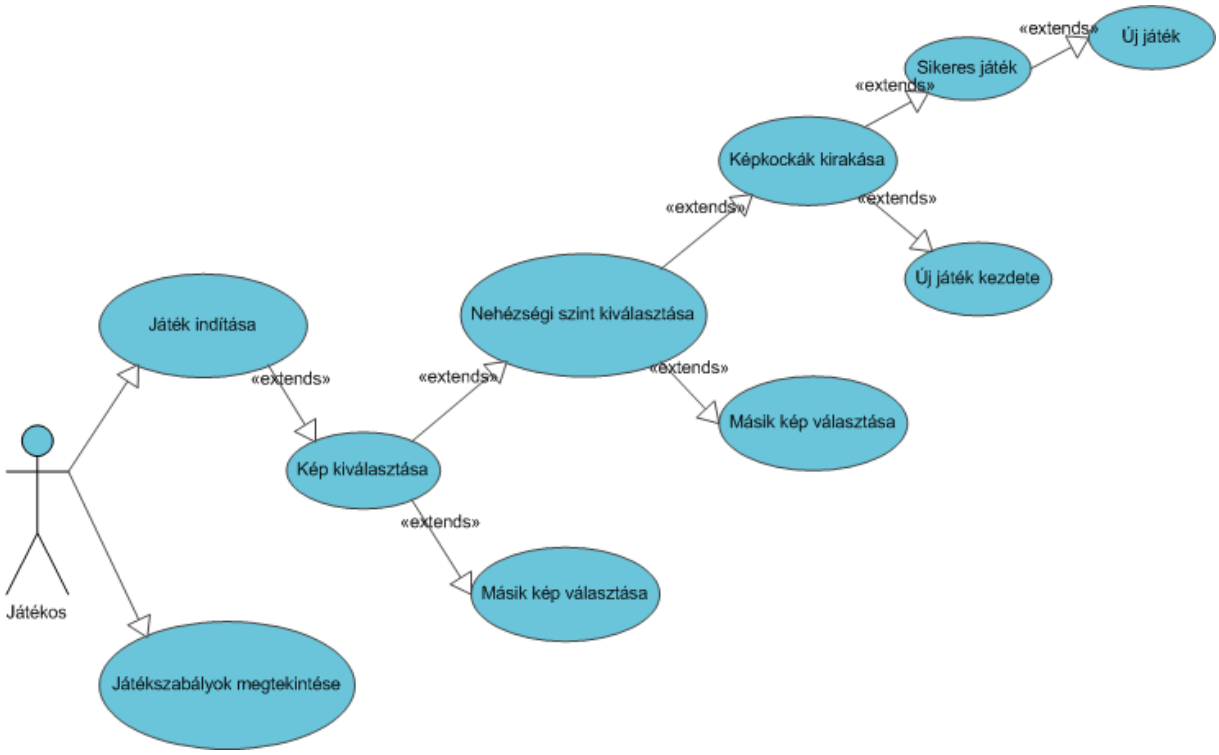
9 kör



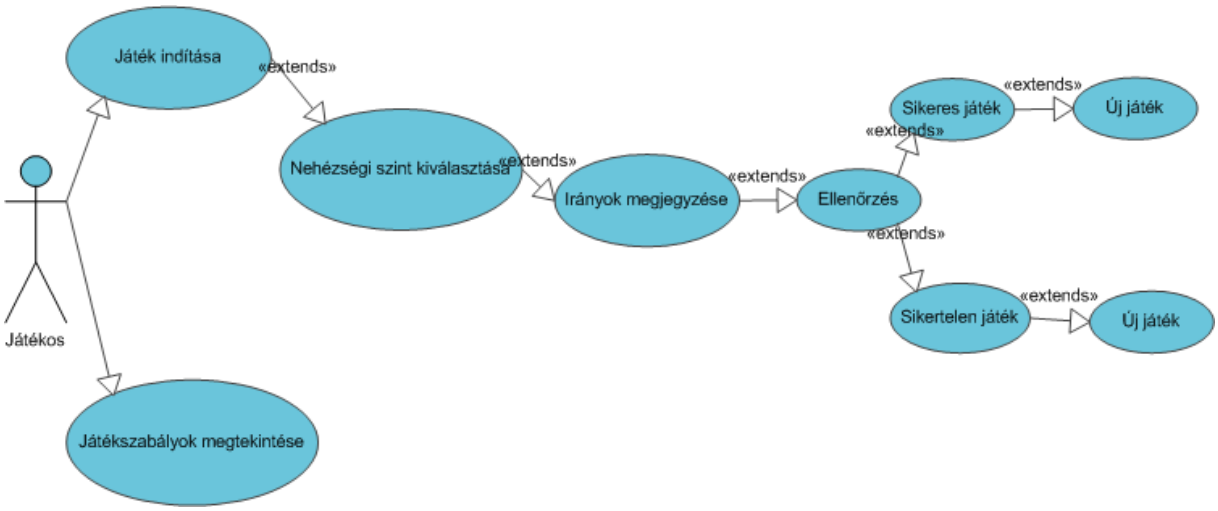
Alakzatok



Képkirakó



Útvonal


Játék indítása
«extends»
Nehézségi szint kiválasztása
«extends»
Irányok megjegyzése
«extends»
Ellenőrzés
«extends»
Sikeres játék
«extends»
Új játék
«extends»
Sikertelen játék
«extends»
Új játék
Játékos
Játékszabályok megtekintése

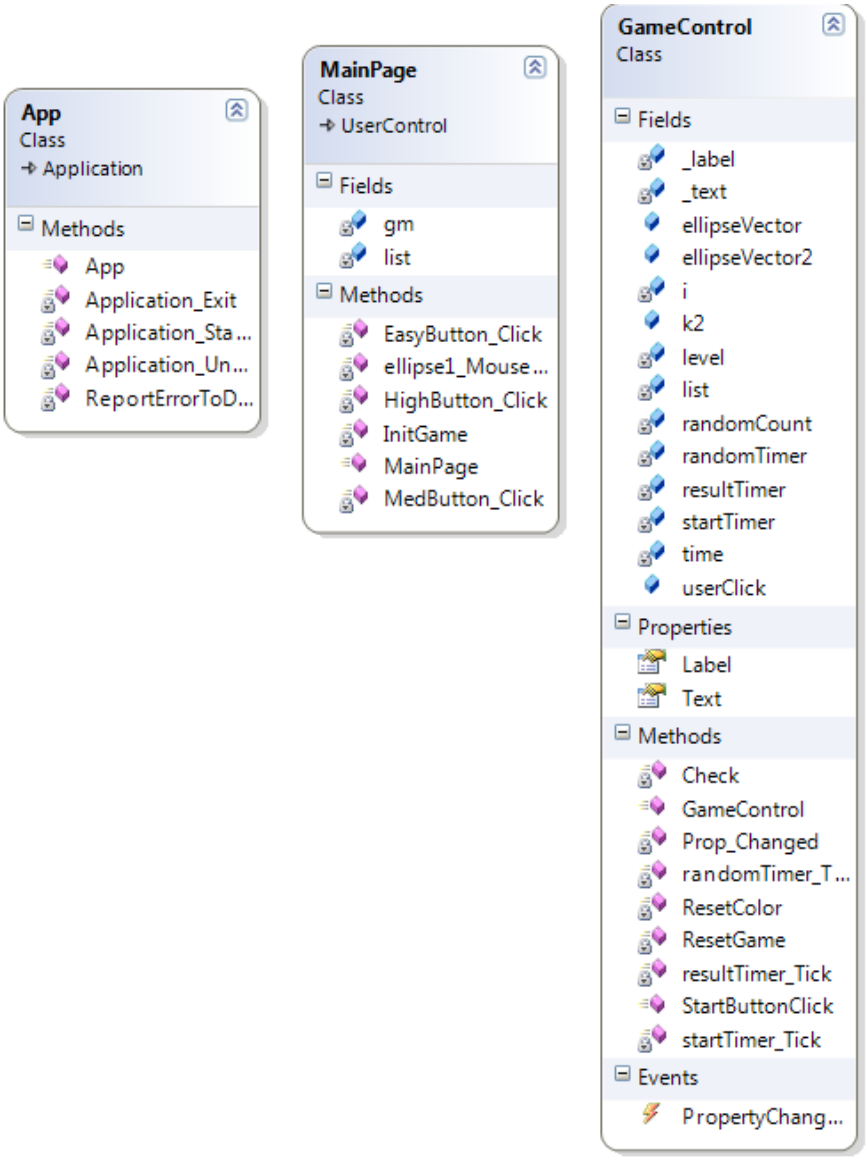
V. A memória játékok szerkezeti felépítése

A játékok megvalósítása

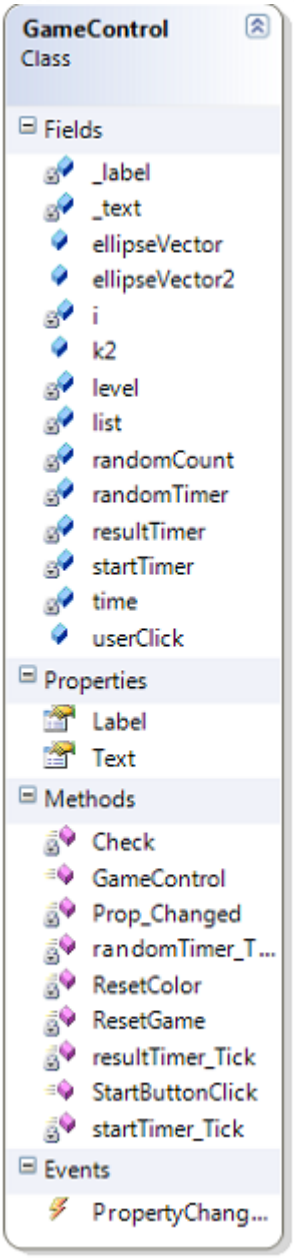
Ebben a fejezetben ismertetjük a játékok felépítését, az implementálás során létrejött osztályokat, és az azok által nyújtott funkciókat. Minden programnál látható az azt modellező UML-diagram, melyeket a Microsoft Visual Studio 2010 Professional integrált fejlesztői környezet generált le számunkra. Jól láthatóak az adott osztályok mezői és metódusai. A fejezet elkövetkező részeiben kitérünk minden játék szinte minden egyes osztályának főbb metódusaira és az ezek által megvalósított tevékenységek leírására.

9 kör játék szerkezete

UML osztálydiagram



Az egyes osztályok által megvalósított tevékenységek



A **MainPage** osztálynak két mezője van: a **gm** és a **list**. A **gm** a ***GameControl*** osztály egy példánya a **list** pedig egy generikus lista pontosabban ***List<Ellipse>***, amelynek típusa **Ellipse**, és amely tárolja a köröket.

Ezen osztály metódusai az ábrán láthatóak.

Az *EasyButton_Click, MedButton_Click, HighButton_Click* metódusok a megfelelő gombok eseménykezelői. Törzsükben az InitGame() metódus hívódik meg melynek paramétere integer típusú, amely a játék szintjét szimbolizálja.

Az *InitGame()* metódus törzsében a ***GameControl*** osztály van példányosítva ,illetve megtörténik a tényleges adatkötés.

Ebben az osztályban található továbbá az körök eseménykezelője, aminek érdekessége hogy mind a 9 körnek egyazon eseménykezelő metódusa van.

A ***GameControl*** osztály mezői a fent látható megfelelő osztálydiagramon látható. A _label és a _text string típusú változók, valójában segédváltozók a **Label** illetve a **Text** típusú propertyk-hez. Ezen osztály továbbá implementálja a ***INotifyPropertyChanged*** interfészt, amely a dinamikus adatkötéshez szükséges, és a property-k set metódusában válaszolnak a binding engine-nek [9].

Az adatkötés lényege, hogy nincs átadva az osztály konstruktorának a **TextBlock** mint referencia, hanem ennek megfelelő típusú (string) property van létrehozva, és ha azok változnak változnak a **TextBlock**-ok tartalma is.

A **level** változó a nevének megfelelően a játék nehézségi szintjét tárolja ez integer típusú.

Az **ellipseVector** és az **ellipseVector2** 9 elemmel rendelkező tömbök, integer típusúak. Minden körnek van egy sorszáma, és ennek megfelelően ha a játékos egy körre kattint, akkor a megfelelő tömbbeli elem illetve elemek 1-re állítódnak. A **k2** kettő változó pedig a pontszámításhoz szükséges, azt számolja meg hogy a játék hány kört színezett ki.

Az **i** ciklusokhoz használt segédváltozó.

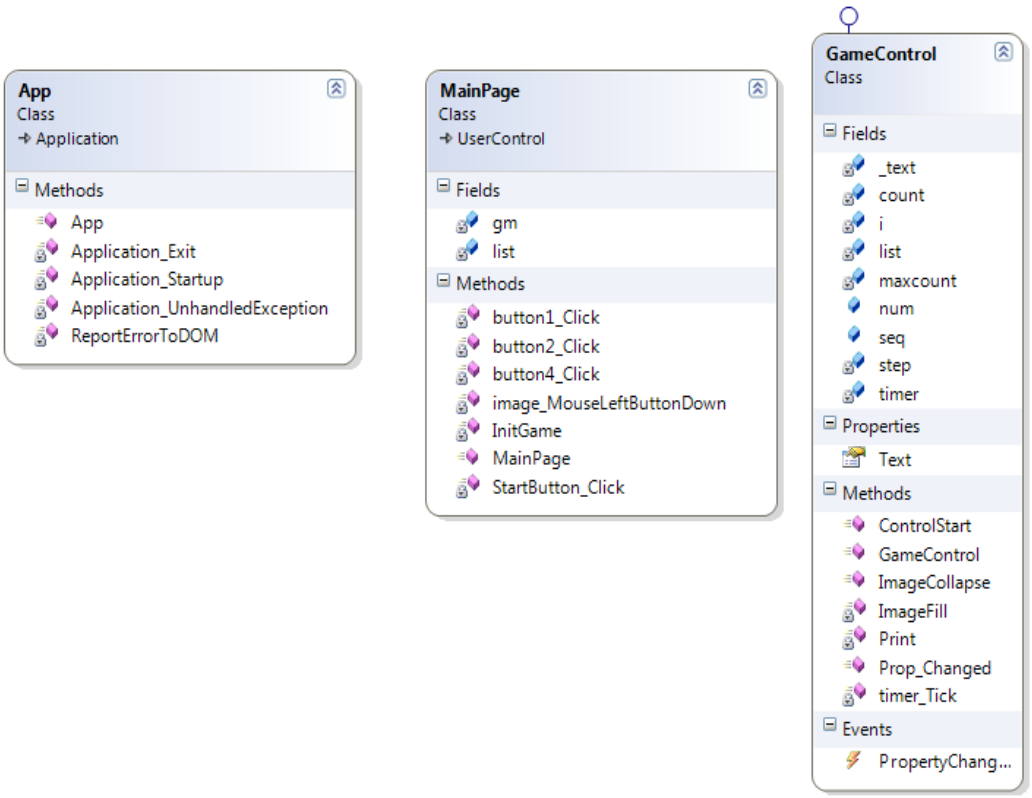
A **time** és **randomCount** pedig az időkorlátot és a véletlenszerű színezések gyakoriságát definiálja.

A **randomCount** a **level** értékének megfelelően változik. 1-es **level** érték mellett a gyakoriság 10, 2-esnél 20, 3-asnál pedig 40.

Továbbá a **startTimer, randomTimer, resultTimer**, a ***DispatcherTimer*** osztály példányai. Ténylegesen ezek valósítják meg a játék működését. Minden szál befejezésekor indul a következő szál így biztosítva a játék menetét.

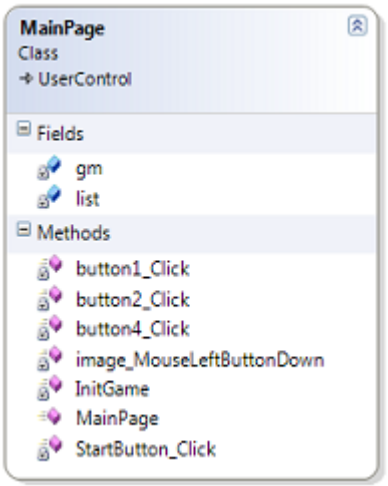
Alakzatok játék szerkezete

UML osztálydiagram



Az egyes osztályok által megvalósított tevékenységek

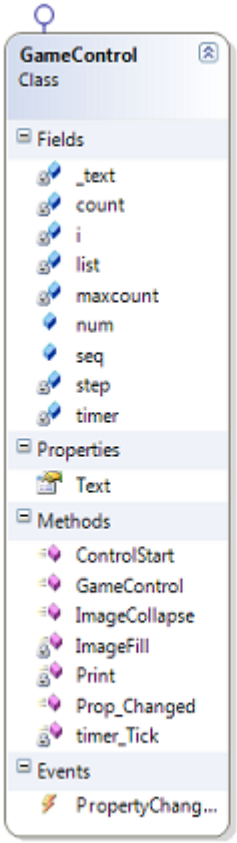
Az ábrán látható az **App** osztály, ami az előző játékban leírtakhoz hasonlóan, az alkalmazás inicializációját valósítja meg.



A **MainPage** osztály mezői a gm és a list. A *gm* a **GameControl** osztály egy példánya, a *list* pedig a játékfelületen található **Image** controlok tárolója (Image típusú generikus lista). A *button1_Click(), button2_Click()* valamint a *button4_Click()* a nehézségi szinteket reprezentáló interaktív gombok eseménykezelő metódusai. Ezen metódusok törzsében egyöntetűen az *InitGame()* metódus kerül meghívásra. Az *InitGame()*-nek egyetlen paramétere van amely egész típusú és a nehézségi szintet szimbolizálja. Értékei a következő számok lehetnek: 1,2,3. Természetesen a nehézségi szintnek megfelelően. Az *image_MouseLeftButtonDown()* eseménykezelő metódus akkor fut le, ha a játék felületén, mikor a felhasználóhoz kerül a vezérlés, ő a megjelenő alakzatok valamelyikére kattint. Ez a metódus van rendelve, minden Image controlhoz. Azonban azt, hogy mégis tudjuk, hogy melyik alakzat kattintás eseménye indította el az eseménykezelő metódust, a törzsben a következőképpen lett implementálva: Minden eseménykezelő metódus specifikációjában, 2 formális paraméter található, nevezetesen a sender mely **object** típusú (ez reprezentálja a küldőt), valamint az **EventArgs** osztály valamely megfelelő leszármazottja, melynek neve általában e. Ezen esetben a leszármazott osztály a **MouseButtonEventArgs.** A kérdéses eseménykezelő metódus törzsében, példányosítva van az **Image** osztály **image** néven, mely példányosítás jobb oldalán, a **sender** paraméter van típuskényszerítve **Image** objektummá. Továbbá ebben a metódusban, történik meg az ellenőrzése annak, hogy a játékos a megfelelő alakzatra kattintott e, avagy sem. Az *InitGame()* metódus, mint ahogy az előbbiekben említésre került, a *GameControl()* osztályt példányosítja, valamint deaktiválja az **Indul** interaktív gombot.

Az osztály konstruktor (*MainPage()*) törzsében, lefut az szóbanforgó *InitializeComponent()* metódus, majd feltöltődik az **Image** controlok tárolója (list, generikus lista).

A *StartButton_Click()* az **Indul** interaktív gomb eseménykezelő metódusa, meghívja a gm objektum *ControlStart()* metódusát, majd a hozzá rendel gombot deaktiválja.



Amint az ábrán látható az osztálynak meglehetősen sok mezője van. A **_text** string típusú változó, valjójában egy segédváltozó a **Text** property-hez, erre pedig azért van szükség, mert a **Text**_property adatkötésben áll a játék felületén elhelyezkedő **TextBlock**-hoz. Azonban a **Text** property értéke a játék menete során dinamikusan változik, ezért dinamikus adatkötésre van szükség. Ehhez az osztálynak implementálnia kell az **INotifyPropertyChanged** interfészt. A változásról a **binding engine** a **Text** property set metódusában értesül. A **count** és a **maxcount**_az alakzatok fiókokban való lefutását szabályozza. A **maxcount** értéke statikus, valójában a nehézségi szint határozza meg. A count értéke dinamikus, maximális értéke annyi amennyi a **maxcount**. Az *i* egész típusú változó a fikókokban mozgó **Image** control (image1 néven) függőleges pozíciójáért felelős, azaz ennek változása miatt mozog a control.

A **num** egész típusú változó, tárolja annak az alakzatnak a számát, amelyik alakzatra a játék kíváncsi, ennek értéke véletlenszerűen állítódik be, mely értéket az [1,5] intervallumból veheti fel.

A **seq** egész elemeket tartalmazó egydimenziós tömb. Az elemek értéküket szintén az [1,5] intervallumból vehetik fel. Ez a tömb arra szolgál, hogy az egyes fiókokban, mi legyen a mozgó komponens (**image1**) által felvett alakzat. A tömb elemszáma 5.

A **step** egész típusú változó, a mozgó **Image** control pillanatnyi helyzetét tárolja(azaz hányas számú fióknál jár a control). Értéke maximálisan 5 lehet.

A **timer** a **DispatcherTimer** egy példánya, az valósítja meg, a mozgó **Image** control mozgását a fiókokban.

ControlStart(): Ezen metódus indítja el a játékot, azaz meghívja a timer példány Start() metódusát.

GameControl(): Osztálykonstruktor, 2 paramétere van, egy List<Image> típusú lista(azaz az Image controlokat tárolja beleértve a mozgó controlt is), a másik pedig egész típusú és értéke a játék szintje azaz, 1,2 vagy 3 lehet. Törzsében továbbá az előbb említett mezők értékei kerülnek inicializálásra, illetve az Images mappából betöltődnek a képek.

ImageCollapse(): Ez a metódus, láthatatlanná teszi az alakzatokat a játékfelületen, a **Visibility.Collapse** property segítségével.

ImageFill(): Az alakzatokat láthatóvá teszi a játékfelületen, a Visibility.Visible property segítségével.

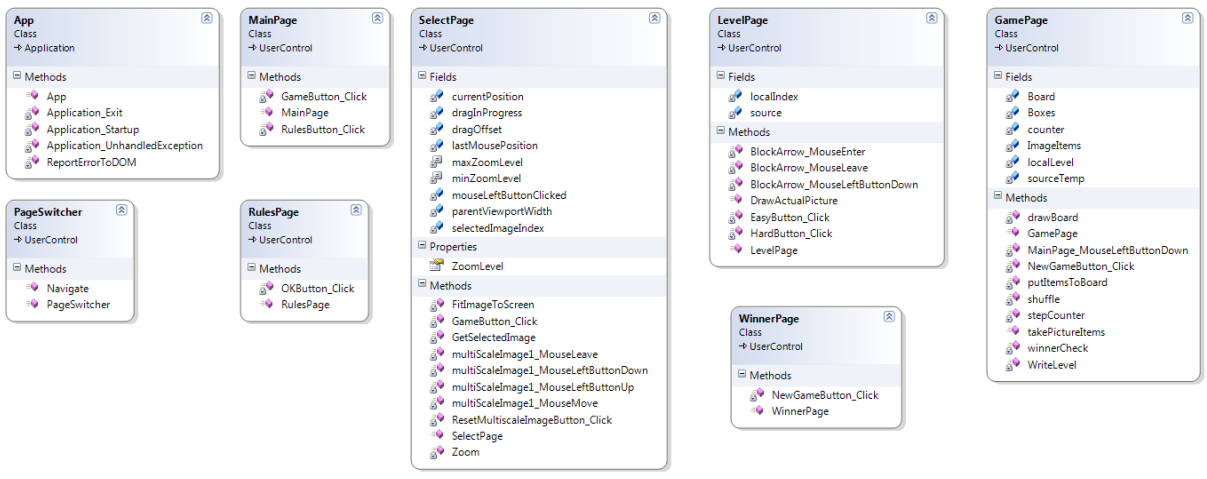
Print(): Átállítja a játékképernyőn lévő TextBlock szövegét, a fenti képernyőképen látható valamelyikére.

Prop_Changed(): A Text property megváltozásáért felelős metódus, valójában ez a binding engine eseménykezelője.

timer_Tick(): A timer objektum intervallum property-ben megadott időközönként lefut ez a metódus.

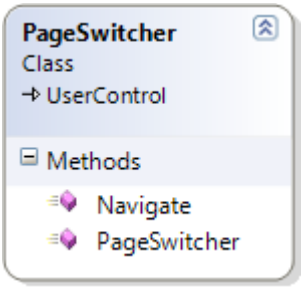
Képkirakó játék szerkezete

UML osztálydiagram



Az egyes osztályok által megvalósított tevékenységek

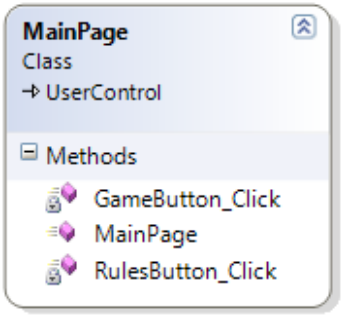
A nagy UML ábrán látható az ***App*** osztály, amely az alkalmazás indításának feltételeit definiálja. Ez az első osztály, amelyet a Silverlight indításkor betölt és végrehajt, illetve ezt zárja be legutoljára az alkalmazás leállításakor. A Silverlight az ***Application_Startup*** és ***Application_Exit*** eseményekkel nyitja meg és zárja be ezeket a fájlok. A MainControl alapértelmezés szerint nem jelenik meg, de ha beállítjuk az ***Application_Startup*** eseménykezelőben a **RootVisual** tulajdonságot a ***MainControl*** egy példányaként, akkor ez lesz a kezdő képernyőnk. Az ***App.xaml*** nem támogatja vizuális elemek hozzáadását, a XAML rész csak definíciós célra szolgál [9].



A Képkirakó játékban a vezérlés során a képernyők (osztályok) közötti váltást a ***PageSwitcher*** osztály végzi. A fentebb említett ***RootVisual*** tulajdonságnál ez a UserControl van példányosítva. Lényege, hogy egy olyan control-t hozunk létre, amely nem tartalmaz semmilyen vizuális elemet, és képes arra, hogy felvegye bármelyik control-nak a felületét.

A PageSwitcher osztály metódusai:

- *PageSwitcher():* az osztály konstruktora. Szerepe az, hogy ha a UserControl Content tulajdonsága null értékű, akkor betölti a ***MainPage*** osztályt.
- *Navigate (UserControl nextPage):* a paraméterként megadott UserControl-t beleteszi a Content-be, így felveszi annak kinézetét és betölti azt.



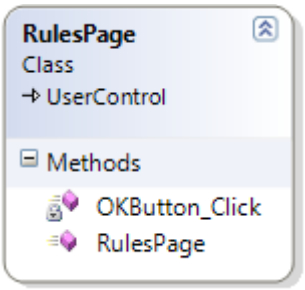
A ***MainPage*** osztály a játék kezdőfelületét definiálja, amelyből eljuthatunk a játékra vonatkozó leíráshoz és a szabályokhoz, vagy elindíthatjuk az alkalmazást.

A MainPage osztály metódusai:

- *MainPage()*: az osztály konstruktora, amely az inicializálásért felelős.

- *GameButton_Click (object sender, RoutedEventArgs e):* eseménykezelő metódus, amelyre a játék indítása gombra való kattintáskor kerül át a vezérlés. A ***PageSwitcher*** osztály *Navigate* függvénye segítségével betölti a játék következő képernyőjét.

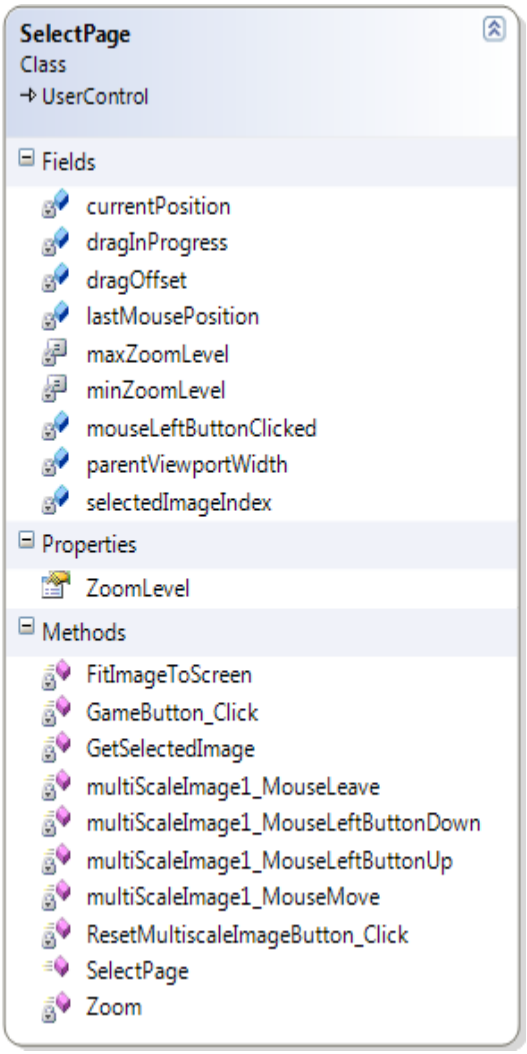
- *RulesButton_Click (object sender, RoutedEventArgs e):* eseménykezelő metódus. Feladata, hogy betöltse a játék tudnivalóit leíró képernyőt.



A ***RulesPage*** a játék leírását tartalmazza. Egy tájékoztató szöveget ír ki a képernyőre, amely elmagyarázza a felhasználónak a lépéseket és a játék funkcióit.

A RulesPage osztály metódusai:

- *RulesPage():*az osztály konstruktora
- *OKButton_Click (object sender, RoutedEventArgs e):* eseménykezelő függvény, amelynek lefutása után visszatöltődik a játék kezdeti felülete.



A ***SelectPage*** bizonyos számú kép megjelenítését, és ezek közül egy darab kiválasztását teszi lehetővé. A játékos a neki megfelelő képet használhatja ezáltal a játék folyamán.

A Silverlight tartalmaz egy különleges technológiát, amelynek segítségével egyedülálló módon kezelhetünk képeket alkalmazásunkban. Nagyon nagy felbontású képeket szolgáltathatunk a felhasználóknak anélkül, hogy a megjelenítés előtt le kellene tölteniük a teljes képet. Egy adott nagyítású ábrát beágyazhatunk egy másik, eltérő nagyítású képbe, ezzel számos különleges effektust biztosíthatunk. Ennek a technológiának a neve **Deep Zoom**, amely nagy virtuális teret biztosít a képek számára [10].

A XAML-ben ezt az osztály fő vezérlőeleme egy **MultiScaleImage** elem valósítja meg, amellyel méretezést és nagyítást vezérelhetünk. Lényege,

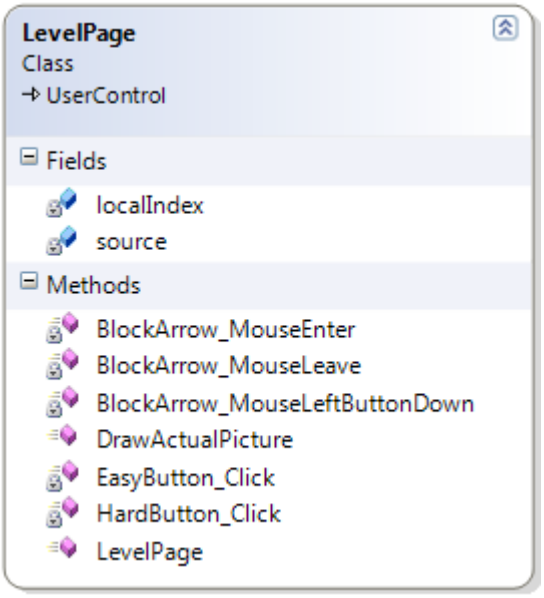
hogy a MultiScaleImage control **Source** attribútumának meg kell adnunk egy elérési útvonalat, amely egy XML fájlra mutat. Ezt a fájlt a **Deep Zoom Composer** nevű program segítségével hozhatjuk létre. A Deep Zoom Composer létrehozza a több képből készült kollekció meta adatait, és ezek között található meg a számunkra szükséges XML fájl, melynek neve általában *dzc_output.xml*. Az osztályban levő metódusok és mezők ehhez a központi vezérlőelemhez kapcsolódnak.

A SelectPage osztály metódusai:

- *SelectPage():* az osztály konstruktora, amely inicializálja a képgyűjteményt.
- *GetSelectedImage (Point p):* a felületen megjelenő kép egy kollekció, tehát vannak elemei. Ezek az elemek is képek, típusuk **MultiScaleSubImage**. A függvény paraméterként megkap egy pontot és megkeresi, hogy ez a pont melyik alelemben van benne. Az alképek köré egy **Rect** objektumot rajzol, és ez alapján tudja megmondani, hogy az adott pont benne van e valamelyik képben (tehát a felhasználó egy adott képre kattintott) vagy sem. Ha egy Rect objektum tartalmazza a pontot, akkor visszatérési értékként szolgáltat egy **int** típusú számot, amely az adott kép azonosítója az adott kollekcióban. Ellenkező esetben -1 értékkel tér vissza.
- *multiScaleImage1_MouseLeftButtonDown (object sender, MouseButtonEventArgs e)*: eseménykezelő metódus, ami akkor hajtódik végre, ha a felhasználó az egér bal gombját lenyomja. Elvégzi az egérkurzor aktuális pozíciójának és a logikai nézet koordinátáinak meghatározását.
- *multiScaleImage1_MouseLeftButtonUp (object sender, MouseButtonEventArgs e)*: eseménykezelő függvény. Az esemény akkor következik be, ha az egér bal gombját a felhasználó felengedi. Az egér aktuális pozícióját paraméterül adva meghívódik a **this** példány **GetSelectedImage** nevű metódusa, ami a előzőkben olvasható működés szerint visszaad egy **index**et. Ha ez az index -1 értéktől különböző szám (tehát a felhasználó az egérkurzorral a képgyűjtemény valamelyik tagján állt), akkor hívásra kerül a **FitImageToScreen** nevű metódus, aminek működését a későbbiekben tárgyalom.
- *multiScaleImage1_MouseMove (object sender, MouseEventArgs e)*: az egér mozgatására bekövetkező eseményt kezelő függvény. Ha az egér bal gombját lenyomva tartva következik be ez, akkor angol kifejezésén nevezve egy **drag** műveletet hajthatunk végre, tehát mozgathatjuk a megjelenő képet. Ha még több képet

szeretnénk a gyűjteményünkbe helyezni, akkor ez a funkció nagyon fontos lehet számunkra. Segítségével a nem látható képekhez navigálhatunk. A metódus a **drag** művelet során mindig újraszámolja a **ViewportOrigin** tulajdonságot, a logikai nézőpont bal felső sarkát.

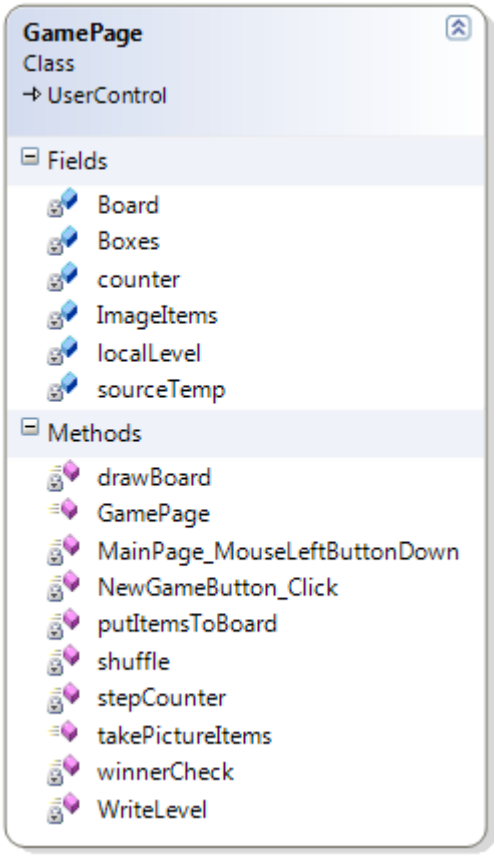
- *multiScaleImage1_MouseLeave (object sender, MouseEventArgs e)*: eseménykezelő függvény, amely az egér mutatójának a **MultiScaleImage** vezérlőelemről való távozására reagál. Ekkor a program észleli, hogy ilyenkor nem történhet kép kiválasztása (nem reagál az egér bal gombjának lenyomására), és nem történhet mozgatás sem.
- *FitImageToScreen (int index)*: paraméterként megkap egy **int** típusú **index**et, amely egy kép azonosítója a kollekcióban, és ha ez nem egyenlő -1-gyel, akkor a kép kitölti a rendelkezésre álló területet és felnagyításra kerül, így jobban láthatóvá válnak a részletek. Ezzel egy időben láthatóvá válik két nyomógomb, melyeknek **Visibility** attribútumai felveszik a **Visible** értéket.
- *Zoom (double zoomFactor, Point zoomPosition)*: a metódusnak két paramétere van. A **zoomFactor** a nagyítás mértékét adja meg, míg a **zoomPosition** a nagyítás kiindulópontjának koordinátáit. Feladata, hogy egy új pontba zoomoljon. Az alkalmazásban bele van építve az egérgörgő mozgatása által bekövetkezett esemény kezelése, de ez a funkció nincs engedélyezve. Ha nagyon sok képet szeretnénk megjeleníteni, akkor lehetne fontos ez számunkra.
- *GameButton_Click (object sender, RoutedEventArgs e)*: ha a J**áték indítása ezzel a képpel** nyomógombra kattint a játékos, akkor bekövetkezik egy esemény, amelyet ez a metódus kezel. Átnavigál egy következő képernyőre, ahol majd a nehézségi szintet választhatjuk ki.
- *ResetMultiscaleImageButton_Click (object sender, RoutedEventArgs e)*: ez az eseménykezelő akkor aktiválódik, amikor a **Másik kép kiválasztása** gombra kattint a felhasználó. Alapállapotba hozza a központi képet. A nyomógombokat pedig inaktív állapotba helyezi.



A **LevelPage** osztály a játék nehézségi szintjének kiválasztását valósítja meg.

A SelectPage osztály metódusai:

- *LevelPage (int index)*: az osztály konstruktora, amely paraméterezve van. Formális paramétere egy **int** típusú **index**, amely az előző képernyőn kiválasztott kép azonosítója. Inicializációt végez és meghívja a **DrawActualPicture** metódust.
- *DrawActualPicture (int index)*: a metódus feladata, hogy a felület hátterébe helyezze a kiválasztott képet. A paraméterként megkapott **index** alapján megkeresi, hogy melyik kép az aktuális, és ennek az elérési útvonalát adja át a **Source** attribútumnak.
- *BlockArrow_MouseEnter (object sender, System.Windows.Input.MouseEventArgs e)*: a képernyő bal alsó sarkában látható egy **VISSZA** feliratú nyíl. Ha a felhasználó az egér kurzorát erre a nyílra viszi, akkor lefut ez az eseménykezelő metódus. Feladata, hogy láthatóvá tegye, hogy a játékos ezen az alakzaton áll az egérrel. Ezt úgy teszi meg, hogy a **BlockArrow** objektum **Fill** tulajdonságának **Opacity** attribútumát megváltoztatja.
- *BlockArrow_MouseLeave (object sender, System.Windows.Input.MouseEventArgs e)*: ez a függvény az előző esemény ellenkezőjének bekövetkezésekor aktiválódik, tehát amikor a nyílról lekerül az egér mutatója. Visszaállítja az eredeti **Opacity** értéket.
- *BlockArrow_MouseLeftButtonDown (object sender, MouseButtonEventArgs e)*: ha a játékos a **VISSZA** nyílon lenyomja az egér bal gombját, akkor a vezérlés erre az eseménykezelőre kerül. Visszanavigál az előző oldalra, ahol kiválaszthatunk egy másik képet.
- *EasyButton_Click (object sender, RoutedEventArgs e)*: ha a **KÖNNYŰ** nehézségi szintet reprezentáló nyomógombon történik kattintás, a metódus átnavigál a **GamePage** osztályra, és tájékoztatja (paraméterül átadja) a könnyű nehézségi szintről, majd a kép forrását is megadja.
- *HardButton_Click (object sender, RoutedEventArgs e)*: az előző függvényhez hasonló működést végez, annyi különbséggel, hogy **nehéz** játékfokozatot állít be.



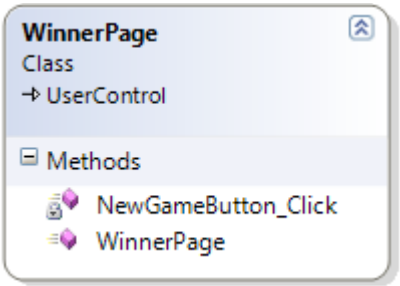
A konkrét játékot megvalósító osztály a **GamePage**, amely egy táblát definiál, amin a képdarabok tologatása megy végbe.

A GamePage osztály metódusai:

- *GamePage (int level, string source)*: az osztály konstruktora, amely paraméterként megkap egy int típusú számot, a nehézségi szintet, valamint a játékba kerülő kép forrását. Ezeknek az adatoknak az ismeretében beállítja a játék főbb tulajdonságait és értékeit.
- *stepCounter ()*: ezen függvény egy egyszerű feladatot valósít meg. A képernyő bal felső sarkában látható **lépésszámláló** értékét növeli mindig eggyel, amikor a felhasználó elmozgat egy képkockát.
- *WriteLevel ()*: a metódus szerepe az, hogy a felület bal felső sarkában levő nehézségi szint felirat mellé kiírja az aktuális fokozatot.
- *drawBoard (int iteration, int imageSize, int count)*: három paramétere van. Az első paraméter egy **int** típusú **iteration** nevű változó, amely a ciklusok lefutásának számát korlátozza, attól függően, hogy 4x4-es vagy 3x3-as táblát szeretnénk létrehozni. A második formális paraméter az **int** típusú **imageSize**, amely ismét a nehézségi szinthez kötődő tulajdonságokat manipulálja. Ha **könnyű** játékot játszunk, akkor a kép 300x300 méretű lesz, amit a gép feldarabol, ha pedig **nehéz** fokozatot választunk, akkor 400x400 képpel kell dolgoznunk. A **count** paraméter 15 vagy 8 értéket vehet fel, attól függően, hogy milyen nehézségi szintben játszik a felhasználó. A képdarabolás folyamán 100x100 méretű **Rectangle** objektumokat hozunk létre. Ezekhez hozzárendeljük a **BitmapImage** értékeknek megfelelő képdarabokat, majd ezeket a **GameContainer** nevű **Canvasban** a megfelelő helyre pozícionáljuk a **Top** és a **Left** attribútumok beállításával.
- *shuffle (int count)*: feladata, hogy összekeverje véletlenszerűen a táblára kirakott képdarabokat. Egyetlen paramétere van, amely attól függően változik, hogy 16 vagy 9 darab képkockát hozunk létre. A léptető algoritmus százszor halad keresztül a tömbön, és minden alkalommal két véletlenszerűen kiválasztott elemet fog meg. Amennyiben a

két elem eltér egymástól, kicseréli azok tartalmát. Az utolsó elemhez a -1 értéket rendeli.

- *putItemsToBoard (int szamossag, int meret)*: a metódus kirajzolja a táblát, elhelyezi a megfelelő pozícióba a képdarabokat. A paraméterek ismét a nehézségi szinttől függő értékek beállításáért felelősek.
- *takePictureItems (int iteration, int size, int nBoxesID, int nBoardLocation, int nEmptyBox, Canvas c)*: A működés első fázisában megkeresi az aktuális **Canvas**-t, amelyre a felhasználó kattintott, majd a következő lépésben ellenőrzésre kerül, hogy ez az objektum mozgatható-e, tehát van e mellette üres képterület. Ha mozgatható, akkor a program áthelyezi az üres területre, majd frissíti a táblát. Ezen kívül minden egyes áthelyezésnél meghívja a **winnerCheck** függvényt.
- *MainPage_MouseLeftButtonDown (object sender, MouseButtonEventArgs e)*: az eseménykezelő feladata a felhasználói interakció kezelése. Ha a felhasználó egy **Canvas**-ra kattint, amely tartalmaz egy **Image** elemet, akkor aktiválódik. Az eseménykezelő a **Canvas** tömb minden egyes eleméhez hozzá van rendelve. A nehézségi szintnek megfelelően felparaméterezi a **takePictureItems** metódust, majd meghívja ezt.
- *winnerCheck (int count)*: a függvény azt ellenőrzi, hogy a játékos sikeresen kirakta-e a képet. A tábla akkor van kirakva, ha a táblatömb n indexénél minden darab értéke egyenlő n-nel, azaz a **Canvas** 0 index 0, a **Canvas** 1 index 1 és így tovább.
- *NewGameButton_Click (object sender, RoutedEventArgs e)*: eseménykezelő metódus, amely akkor aktiválódik, ha a felhasználó a képernyő jobb alsó sarkában levő **Új játék** feliratú gombjára kattint az egér bal gombjával. A kód lefutása során elkezdődik egy új játék, betöltődik a **SelectPage** osztály.



Ha a játékos kirakta az általa választott képet, akkor betöltődik a **WinnerPage** osztály, melynek szerepe a tájékoztatás és lehetőséget ad új játék kezdésére.

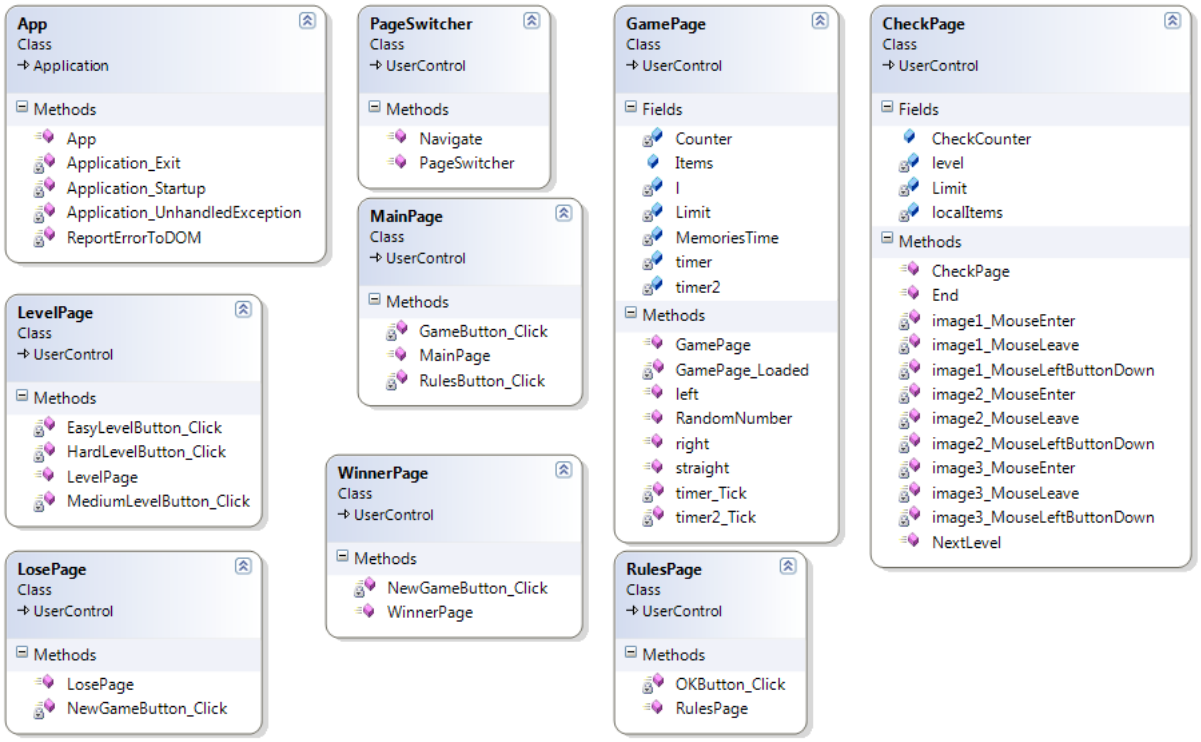
A GamePage osztály metódusai:

- *WinnerPage()*: az osztály konstruktora, amely a felület betöltéséért felelős.

- *NewGameButton_Click (object sender, RoutedEventArgs e)*: ha a játékos az Új játék feliratú gombra kattint, akkor töltődik be ez az eseménykezelő metódus. Elindít egy új játékot, betölti a **SelectPage** osztályt.

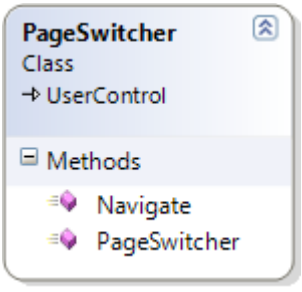
Útvonal játék szerkezete

UML osztálydiagram



Az egyes osztályok által megvalósított tevékenységek

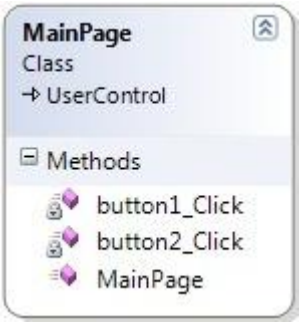
Az **App** osztály az **Application singleton** osztály egy leszármaztatott változata. Ez az osztály töltődik be legelőször, amikor elindítjuk az alkalmazást. Alkalmazásszintű események kezelését teszi lehetővé, illetve alkalmazásszintű erőforrások tárolására is alkalmas.



Az Útvonal játékban a vezérlés során a képernyők (osztályok) közötti váltást a ***PageSwitcher*** osztály végzi. A fentebb említett ***RootVisual*** tulajdonságnál ez a **UserControl** van példányosítva. Lényege, hogy egy olyan control-t hozunk létre, amely nem tartalmaz semmilyen vizuális elemet, és képes arra, hogy felvegye bármelyik control-nak a felületét.

A PageSwitcher osztály metódusai:

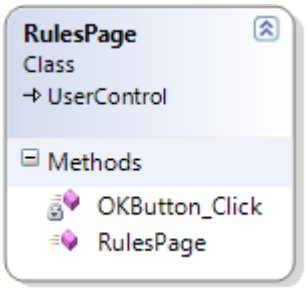
- *PageSwitcher():* az osztály konstruktora. Szerepe az, hogy ha a **UserControl Content** tulajdonsága null értékű, akkor betölti a ***MainPage*** osztályt.
- *Navigate (UserControl nextPage):* a paraméterként megadott **UserControl**-t beleteszi a **Content**-be, így felveszi annak kinézetét és betölti azt.



A ***MainPage*** osztály a játék kezdőfelületét definiálja, amelyből eljuthatunk a játékra vonatkozó leíráshoz és a szabályokhoz, vagy elindíthatjuk az alkalmazást.

A MainPage osztály metódusai:

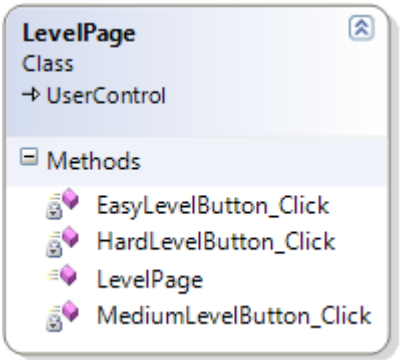
- *MainPage()*: az osztály konstruktora, amely az inicializálásért felelős.
- *GameButton_Click (object sender, RoutedEventArgs e):* eseménykezelő metódus, amelyre a játék indítása gombra való kattintáskor kerül át a vezérlés. A ***PageSwitcher*** osztály *Navigate* függvénye segítségével betölti a játék következő képernyőjét.
- *RulesButton_Click (object sender, RoutedEventArgs e):* eseménykezelő metódus. Feladata, hogy betöltse a játék tudnivalóit leíró képernyőt.



A ***RulesPage*** a játék leírását tartalmazza. Egy tájékoztató szöveget ír ki a képernyőre, amely elmagyarázza a felhasználónak a lépéseket és a játék funkcióit.

A RulesPage osztály metódusai:

- *RulesPage():*az osztály konstruktora
- *OKButton_Click (object sender, RoutedEventArgs e):* eseménykezelő függvény, amelynek lefutása után visszatöltődik a játék kezdeti felülete.

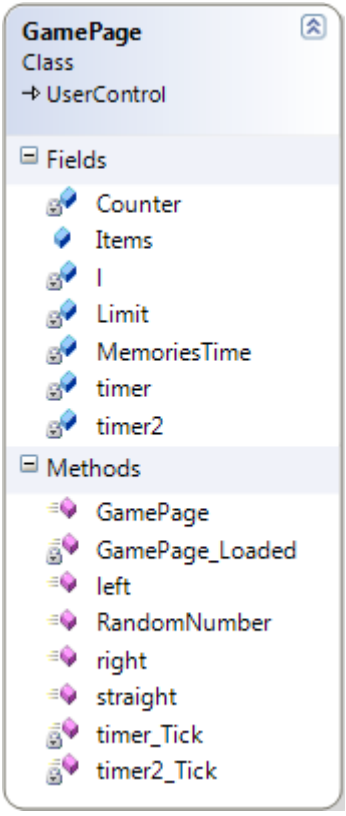


A **LevelPage** osztály a játék nehézségi szintjének kiválasztását hivatott megvalósítani.

A LevelPage osztály metódusai:

- *LevelPage()*: az osztály konstruktora, amely az inicializálást végzi el

- *EasyLevelButton_Click (object sender, RoutedEventArgs e)*: eseménykezelő metódus, amely akkor fut le, amikor a felhasználó az egér bal gombjával a **Könnyű** feliratú gombra kattint. Átnavigál a játék legfőbb képernyőjéhez és a könnyű szintnek megfelelően állítja be a paramétereket.
- *MediumLevelButton_Click (object sender, RoutedEventArgs e)*: a **Közepes** feliratú nyomógombhoz kapcsolódó eseménykezelő. A játék nehézségi szintjét közepesre állítja, és betölti a **GamePage** osztályt.
- HardLevelButton_Click (object sender, RoutedEventArgs e): az előző függvényekhez hasonló. Működése közben a **nehéz** játékfokozat kerül beállításra.



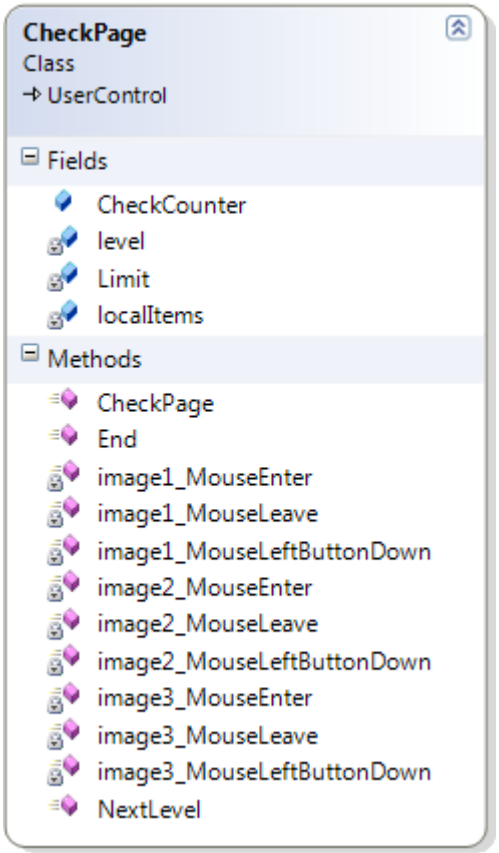
A konkrét játék osztálya a **GamePage**, amelyben definiálva vannak a játéklogikát megvalósító metódusok és attribútumok.

A GamePage osztály metódusai:

- *GamePage (int level, int limit)*: az osztály konstruktora, amely paraméterként megkapja a nehézségi szintet jelző **int** értéket, és egy limitet, amely a program által mutatott útirányok számát korlátozza. Itt történik még a **MemoriesTime** változó beállítása is, amely a megjegyzéshez adott időtartamot jelenti. Az irányok tárolásához használt tömb is itt kerül példányosításra. Egy eseménykezelő is hozzáadásra kerül, amely a felület betöltődésekor fut le, neve **GamePage_Loaded**.
- *GamePage_Loaded (object sender, RoutedEventArgs e)*: ez a függvény a **GamePage** osztály betöltődésének eseményét hivatott kezelni. Kódjában található két darab (**timer**, **timer2**) **DispatcherTimer** példányosítása. Az **Interval** attribútumok is beállításra kerülnek **TimeSpan** objektumokon keresztül, valamint mindkét Timer **Tick** eseménye is hozzáadásra kerül.
- *timer_Tick (object sender, EventArgs e)*: az első **DispatcherTimer Tick** eseményének kezelését végző metódus, mely mindig megvizsgálja, hogy elérte e már a limitet a megmutatott irányok száma. Ha még nem érte el, akkor megkap egy a

RandomNumber függvény által generált véletlen számot, és attól függően, hogy ez milyen számtartományba esik, kirajzolja az adott irány ábráját. Ha már elérte a limitet, akkor betölti a **CheckPage** osztályt.

- *timer2_Tick (object sender, EventArgs e)*: eseménykezelő, amely a második **DispatcherTimer Tick** eseményének bekövetkezésekor aktiválódik. Feladata a képernyő alsó részén található **ProgressBar** értékének változtatása, a hátralévő megjegyzéshez felhasználható idő függvényében.
- *int RandomNumber()*: a **Random** osztály segítségével random számot generál és ad vissza az alkalmazás számára 1 és 3000 között, hogy az értékek nagy tartományból kerüljenek ki, így a szórás nagyobbá válik.
- *left (int count)*: a **bal** irány kirajzolását végzi a függvény. Paramétere egy int típusú érték, amely a tároláshoz használt tömb aktuális indexét adja meg, ahova majd elhelyezi az irány értékét. A felső és az alsó kép **Source** tulajdonságát egy **BitmapImage** objektummal állítja be.
- *right (int count)*: a **left** metódushoz hasonló működéssel bír, annyi különbséggel, hogy a **jobb**ra mutató nyilat helyezi el a felhasználói felületen.
- *straight (int count)*: az előző függvényekhez hasonló tevékenységet végez. Az **egyenes** irány képét helyezi a középpontba.



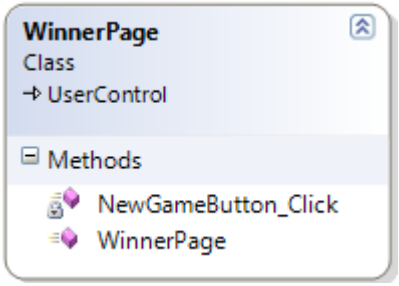
A **CheckPage** a program által mutatott irányok visszaellenőrzésére hivatott osztály.

A GamePage osztály metódusai:

- *CheckPage (int[] items, int limit, int gameLevel)*: három paraméterrel rendelkező konstruktor. Első paramétere az útirányok értékeit tároló tömb, a második a limit, a harmadik pedig a nehézségi szintet jelöli. Inicializálást hajt végre.

- *image1_MouseEnter (object sender, MouseEventArgs e)*: a képernyő bal alsó sarkában elhelyezkedő balra mutató nyílhoz kapcsolódó eseménykezelő metódus. Az **Image Opacity** értékét állítja át, ezáltal kiemeli, láthatóvá teszi, hogy a felhasználó erre az irányra fókuszál.
- *image1_MouseLeave (object sender, MouseEventArgs e)*: az előző esemény ellenkezőjét kezeli, abban az értelemben, hogy ha az egér mutatója lekerül a képről, akkor az **Opacity** attribútum az alapértelmezett értéket veszi fel ismét.
- *image2_MouseEnter (object sender, MouseEventArgs e)*: ha a felhasználó az előre mutató nyílra viszi az egeret, akkor aktiválódik ez az eseménykezelő. Az **Opacity** érték megváltoztatását végzi el.
- *image2_MouseLeave (object sender, MouseEventArgs e)*: a középső képre vonatkozó esemény bekövetkezésekor kerül rá a vezérlés. Alapértelmezetté teszi az **Image Opacity** értékét.
- *image3_MouseEnter (object sender, MouseEventArgs e)*: az **image1_MouseEnter** és az **image2_MouseEnter** metódusokkal megegyező működést végez. A kapcsolódó objektum a jobbra mutató ábra.
- *image3_MouseLeave (object sender, MouseEventArgs e)*: az **image1_MouseLeave** és az **image2_MouseLeave** eseménykezelőkkel ekvivalens feladatot lát el a jobb irány ábrájához kapcsolódva.
- *image1_MouseLeftButtonDown (object sender, MouseButtonEventArgs e)*: az eseménykezelő akkor aktiválódik, amikor a felhasználó a balra mutató nyílon az egér bal gombját lenyomja. Ebben a metódusban történik annak a vizsgálata, hogy a játékos a megfelelő útirányt ismételte e meg. Ha az adott sorrendben ez a bal irány következik és ezen történt az egér eseménye, akkor helyes volt a játékos tevékenysége. Meghívásra kerül a **NextLevel()** és az **End()** metódus. Ha nem megfelelő a bal irány az adott sorrendben, akkor a játék véget ér és betöltődik a **LosePage** osztály.
- *image2_MouseLeftButtonDown (object sender, MouseButtonEventArgs e)*: az egyenes irány ábrájához kapcsolódó eseménykezelő, amely az előző függvénnyel azonos feladatot lát el.
- *image3_MouseLeftButtonDown (object sender, MouseButtonEventArgs e)*: eseménykezelő függvény, amely a jobb irányba mutató nyílra vonatkozik. Az előző két metódus működésével azonos.

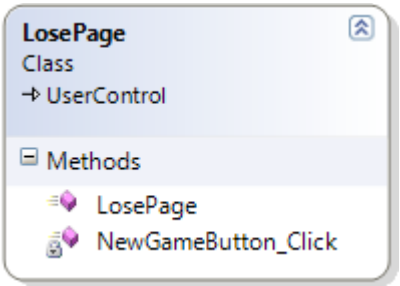
- *NextLevel()*: a függvény megvizsgálja, hogy a felhasználó minden – a program által mutatott – útirányt megismételt-e. Ha elérte a limitet, tehát a mutatott nyilak számát, akkor ez a limit növekszik eggyel és ezzel az új értékkel indul el újból a **GamePage** osztály.
- *End()*: ha a mutatott ábrák száma eléri a 10-et, tehát már a 10. szinten jár a játékos és itt is minden útirányt helyesen eltalált, akkor a játék befejeződik és meghívásra kerül a **WinnerPage** osztály.



Az összes szint sikeres teljesítése esetén kerül a **WinnerPage** osztályra a vezérlés. Értesíti a felhasználót, és lehetőséget ad új játék kezdésére.

A WinnerPage osztály metódusai:

- *WinnerPage()*: az osztály konstruktora, amely egy **InitializeComponent()** metódust tartalmaz, amely az inicializálásért felelős.
- *NewGameButton_Click (object sender, RoutedEventArgs e)*: ha a játékos a **Játék ismétlése** feliratú gombra kattint, akkor töltődik be ez az eseménykezelő metódus. Elindít egy új játékot, betölti a **LevelPage** osztályt.



Ha a felhasználó nem tud teljesíteni egy pályát, akkor a **LosePage** osztályra kerül át a vezérlés. Tájékoztatást nyújt a felhasználónak és segítségével újraindítható a játék.

A WinnerPage osztály metódusai:

- *LosePage()*: az osztály konstruktora.
- *NewGameButton_Click (object sender, RoutedEventArgs e)*:eseménykezelő metódus, amely egy vesztes játék esetén megjelenő képernyőn található Játék ismétlése nyomógombra való kattintáskor aktiválódik. Biztosítja a felhasználónak a játék megismétlését, egy új játék kezdését.

VI. A memória játékokat magában foglaló ASP .NET weboldal

Ezen fejezetben ismertetjük a játékokat magában foglaló **ASP.NET** technológiával készített weboldalt. A weboldal az előbbi fejezetekben említett négy játékot tartalmazza.

A weblap létrehozásánál használt eszközök

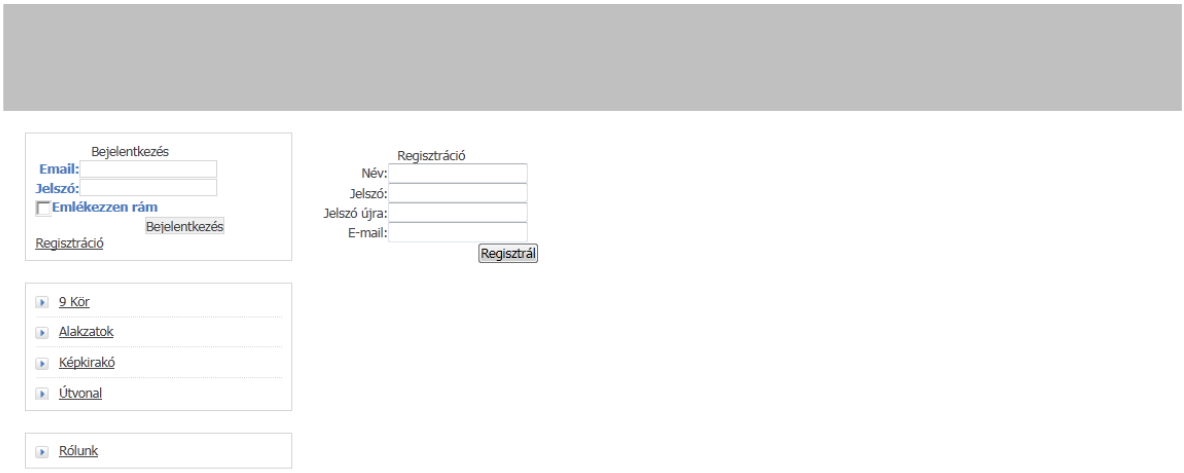
Ennek létrehozásához a **Microsoft Visual Studio 2010 Professional** fejlesztői környezetet használtuk, valamint a **Windows Communication Foundation (WCF)** keretrendszer elemeit, a **Silverlight** alkalmazások és a weboldal továbbá az adatbázis közötti kommunikációhoz [11].

Az oldal által nyújtott funkciók

A weboldalon biztosítja a felhasználó számára a bejelentkezés funkciót, amit az ASP.NET **Login control** valósít meg, illetve felhasználói fiók hiánya esetén regisztráció is lehetséges. Ezen kívül a weblap valósítja meg a memóriajátékok betöltését is. Az alábbi képernyőkép az oldal betöltése után látható:

Autentikáció

Amint látható, a weblap bal oldalán található a **bejelentkezés** opció, ahol egy e-mail és egy jelszó megadásával tudunk bejelentkezni az oldalra. Ez azért fontos, mert a játékok során, ha szeretnénk eltárolni a megszerzett pontjainkat, akkor ez alapján azonosít a rendszer és helyezi el eredményünket az adatbázisban.



Ha még nem regisztrált egy játékos, akkor a **Bejelentkezés** vezérlőelem bal alsó sarkában lévő **Regisztráció** linkre kell kattintania. Ekkor megjelenik a képernyő középső részén egy űrlap, melyen az alábbi adatokat kéri a rendszer:

- Név
- Jelszó
- Jelszó újra
- E-mail cím

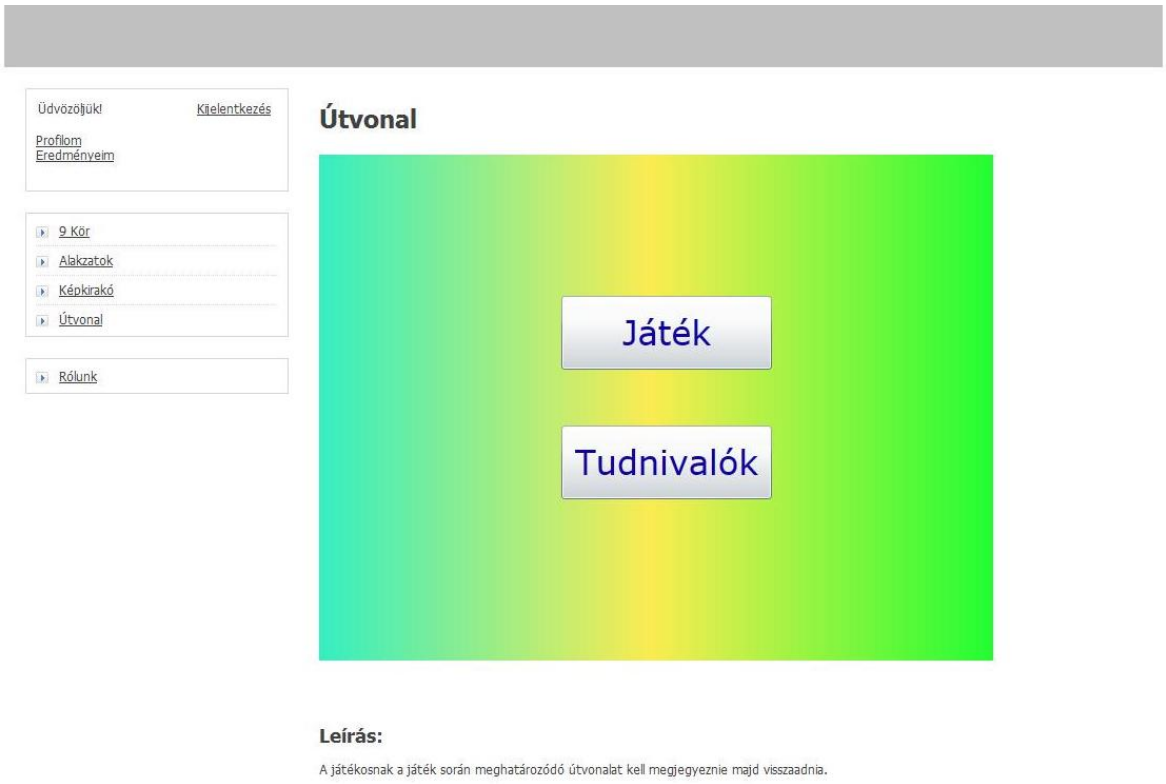
Az adatok megadása után a **Regisztrál** gombra kattintva az alkalmazás eltárolja a felhasználót az adatbázisban.

A bejelentkezés után az alábbi képernyőt láthatjuk:

Bejelentkezés után lehetőség nyílik az aktuális felhasználó **profil** oldalának megtekintésére, az eddigi **eredmények** megjelenítésére. A **Kijelentkezés** linkre kattintva megszűnik a kapcsolat a játékos és az oldal, az adatbázis között.

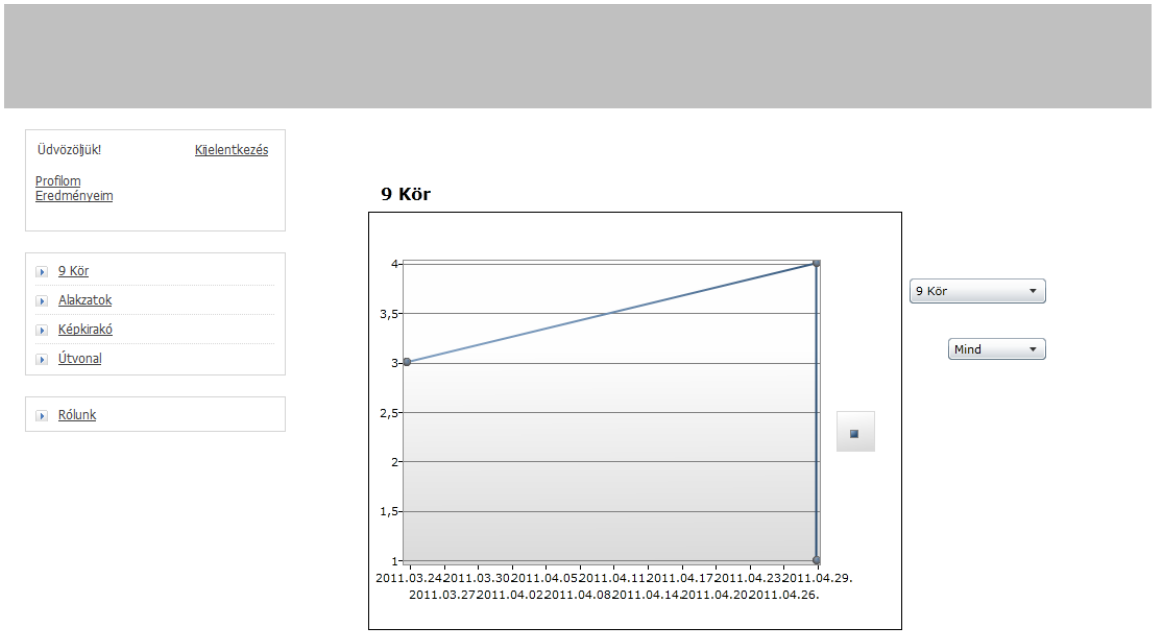
Játékok betöltése

Az oldal bal oldalának közepén láthatjuk a játéklistát, amely az általunk készített négy játékra mutató linket foglalja magában. Ha a felhasználó rákattint valamelyikre ezek közül, akkor az oldal jobb oldalán betöltésre kerül a listából kiválasztott játék. A játékot tartalmazó rész dinamikusan változik a játékképernyő méretének függvényében. A játékképernyő alján található egy rövid leírás az adott játékhoz. Az Útvonal linkre való kattintáskor az alábbi képernyőt figyelhetjük meg:



Az eredmények megtekintése

A weboldal tartalmaz egy az eredmények prezentálásához használatos diagramot:

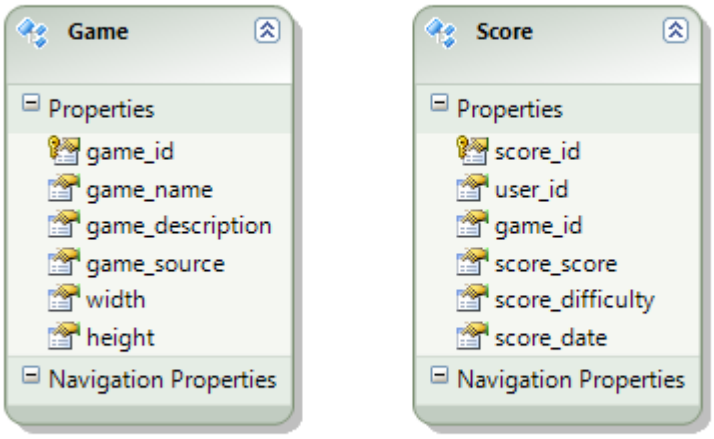


A fenti diagram a weboldal jobb oldalán a felső sarokban található Eredményeim hivatkozásra kattintva hívható elő. A diagramon játékonkénti lebontásban láthatók az eredmények. Ezen diagram most a 9 Kör játékban elért eredményeket szemlélteti. A diagram kirajzolásához a **Microsoft Data Visualization** eszközrendszert használtuk fel, amely biztosítja a szemléletes megjelenítést.

VII. A felhasználó pontjainak tárolására szolgáló adatbázis

Adatbázisséma

Itt ismertetjük, a játékokban elért eredményeket tároló adatbázist. Az adatbázis séma két táblát tartalmaz nevezetesen Game és Score. A táblák definíciója a következőképpen néz ki.



A táblák által tárolt információk

Game tábla

- **game_id**: A játék azonosítója, ezzel hivatkozza a weboldal a kívánt játékot
- **game_name**: A játék neve
- **game_description**: A játék leírása
- **game_source**: A játék elérési útja a weboldalt tartalmazó könyvtárban
- **width**: A játék alkalmazás képernyőjének szélessége
- **height**: A játék alkalmazás képernyőjének magassága

A **Game** tábla arra szolgál, hogy mikor a felhasználó a weboldalon rákattint a kívánt játékot hivatkozó linkre akkor a web felület jobb oldalán megjelenik a játék. Ahhoz, hogy ez

megtörténjen, a weboldal az adatbázisból kinyert információkat használja fel, pontosabban a Game tábla rekordjait.

Score tábla

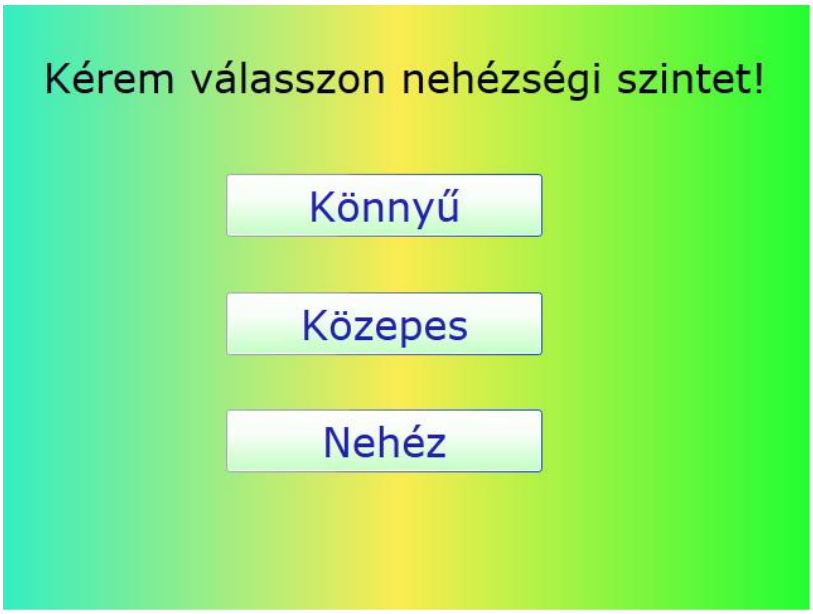
- **score_id:** Az eredmény azonosítására szolgál
- **user_id**: A játékos azonosítója
- **game_id**: A játék azonosítója
- **score_score**: A ténylegesen elért eredmény
- **score_difficulty**: Azt tárolja, hogy a játékos az eredményét melyik szinten érte el.
- **score_date**: Az eredmény táblába kerülésének dátuma.

A **Score** tábla rendeletetése, hogy a játékos az adott játékban elért eredményét tárolja, a fejlődés statisztikája céljából. Minden játék úgy készült, hogy a játék végén az elért eredmény ebbe a táblába tárolódik le.

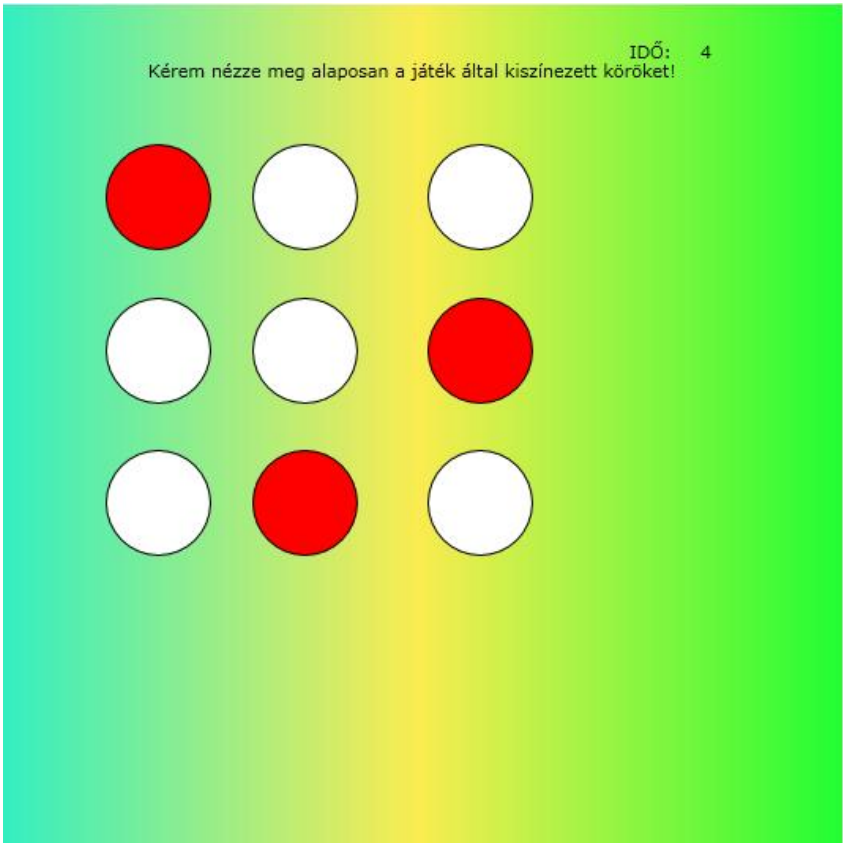
VIII. Felhasználói útmutató

A következő oldalakon, az általunk készített memória játékok használati útmutatóját, az alkalmazásokhoz tartozó felhasználói kézikönyveket tanulmányozhatja a tisztelt olvasó, melyek segítséget nyújtanak a játékokban való tájékozódásban, és a vezérlésben.

A 9 kör játék használata

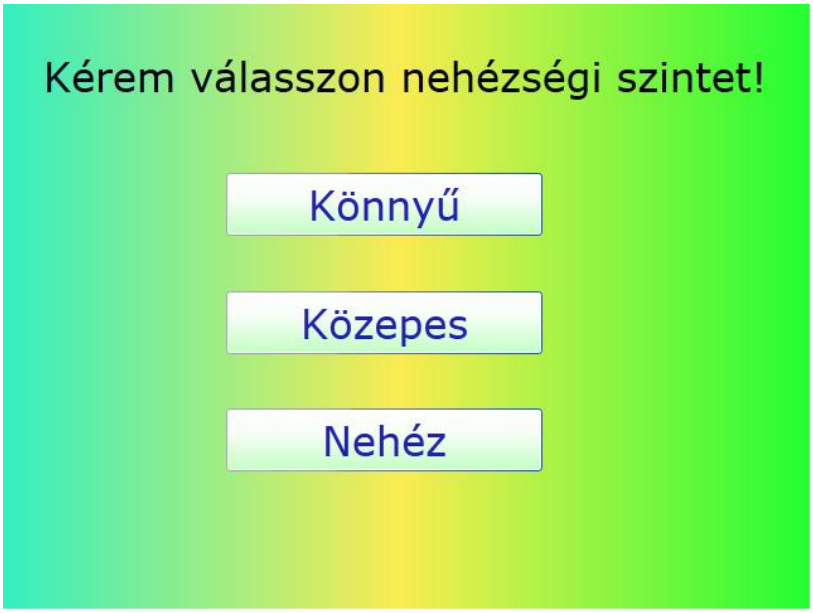


A játékosnak ki kell választania tetszés szerint a neki megfelelő nehézségi szintet, az ábrán látható interaktív gombok segítségével. Ezután megjelenik a játékfelület, amely a következőképpen néz ki.



Amint a képen látható az alkalmazás véletlenszerűen bejelölte 3 kört, a játék leírás az IDŐ címke alatt látható.

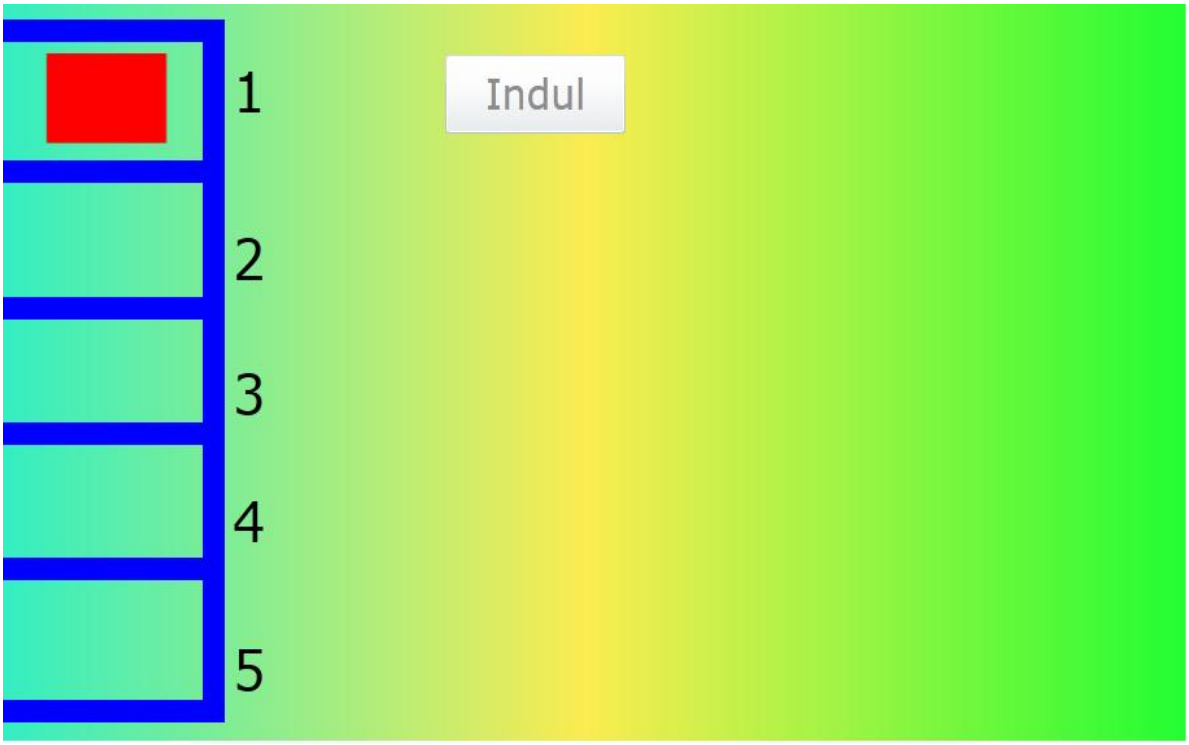
Az Alakzatok játék használata



A játékosnak szintén ki kell választania tetszés szerint a neki megfelelő nehézségi szintet, az ábrán látható interaktív gombok segítségével. Ezután megjelenik a játékfelület, amely a következőképpen néz ki.

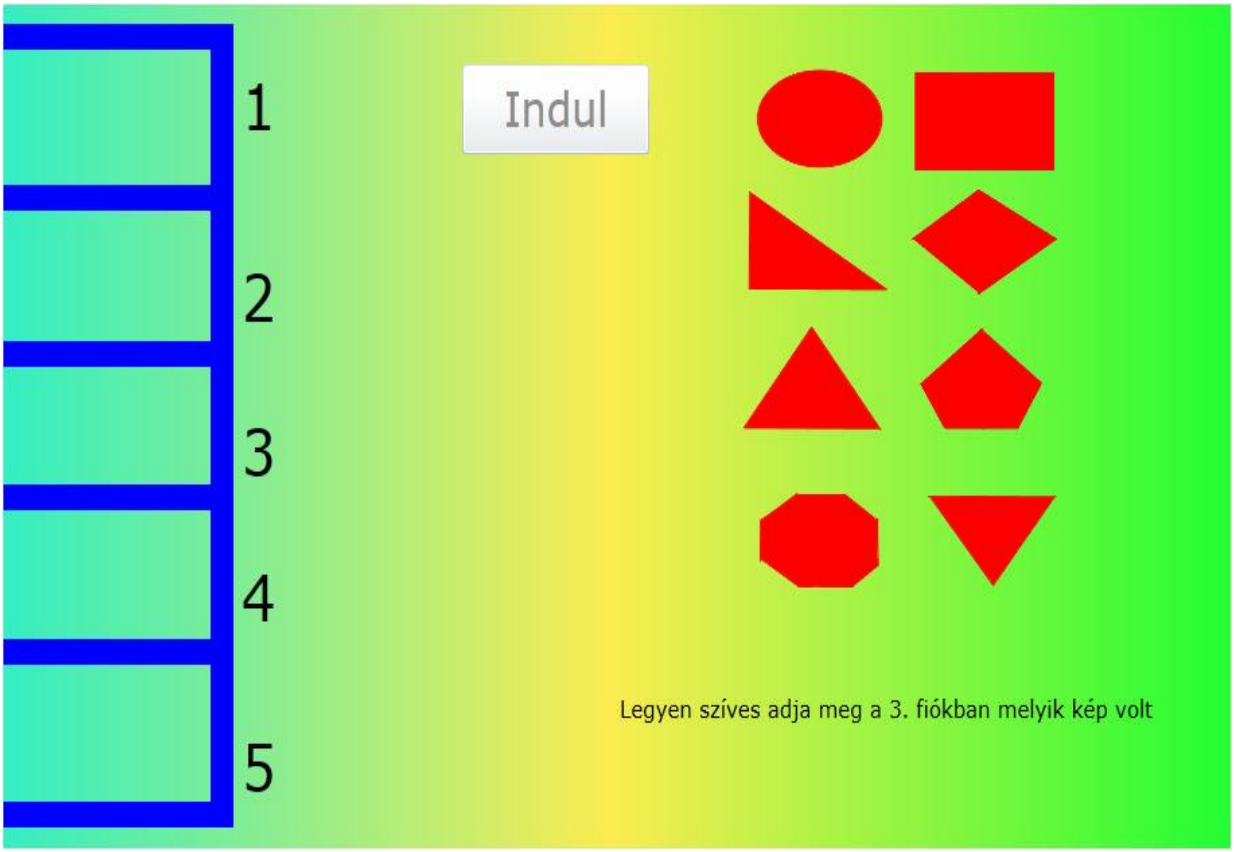
A fenti képernyőn látható az előbb leírt doboz és a fiókok, majd a mellette levő számok, amelyek a fiókok sorszámai. A doboztól jobbra jól látható az Indul interaktív gomb.

Az Indul gombra egyszeri kattintás után a játék felület az alábbi módon néz ki:



Amint látható, a játék elindult. A fenti esetben az 1. számú fiókban egy piros téglalap jelent meg, valamint az Indul interaktív gomb inaktív lett (ez onnan látható, hogy a gomb színe halványabb lett, mint ahogy a benne lévő szöveg is).

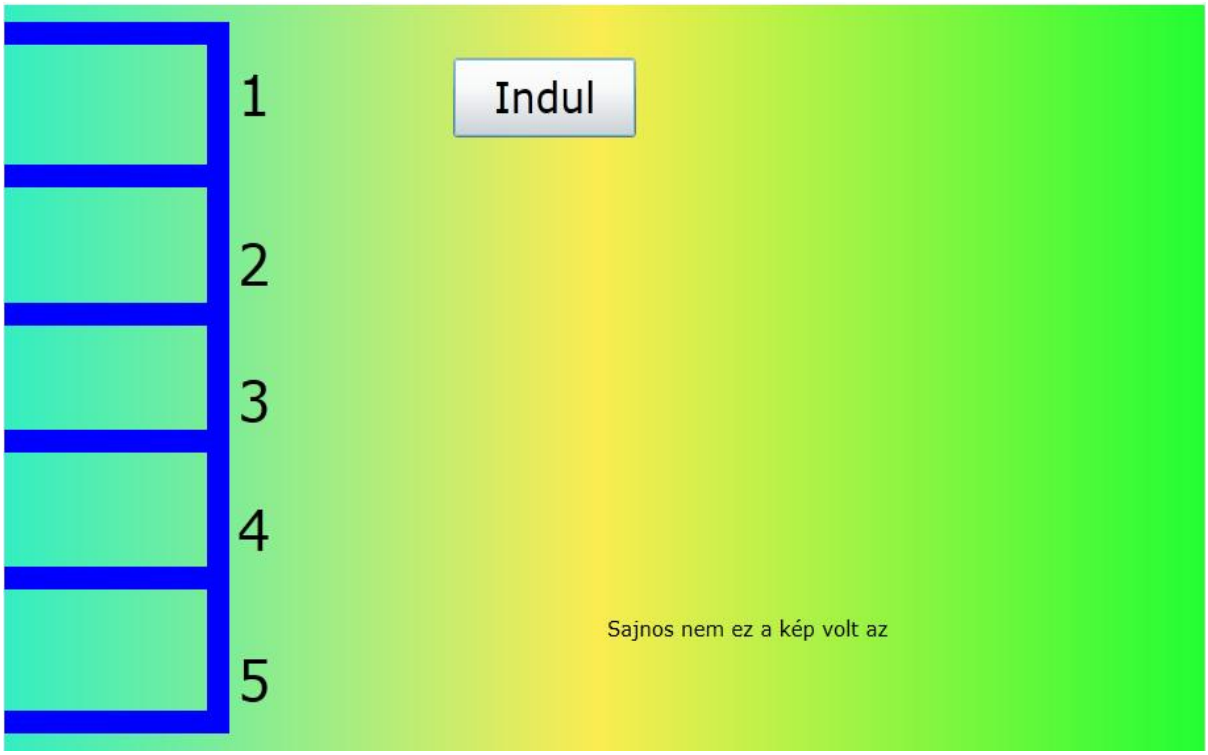
Miután minden fiókban, megjelent a fent ábrázolt alakzatok valamelyike, a játék visszaadja a vezérlést a felhasználónak, vagyis a játékosnak rá kell kattintania, arra az alakzatra amelyik az alkalmazás által meghatározott fiókban helyezkedett el. Ezt az szituációt az alábbi képernyőkép, szimbolizálja:



A mostani szituációban, a játék arra kíváncsi, hogy melyik alakzat volt a 3. fiókban.
Láthatjuk, hogy az Indul interaktív gomb még mindig inaktív.

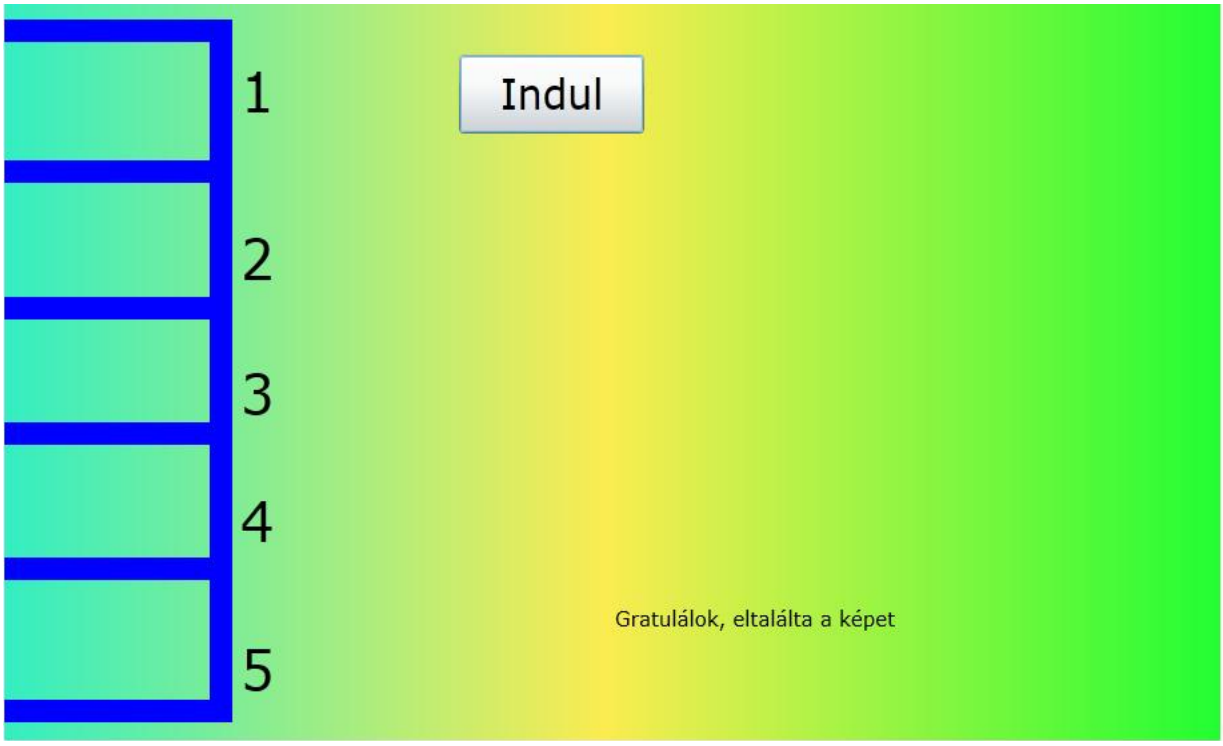
Amint rákattintunk egyszeresen egy alakzatra, úgy az alakzatok eltűnnek, és azok 2 -féle szöveg jelenhet meg: *„Gratulálok, sikeresen eltalálta a képe”* avagy *„Sajnos nem ez a kép volt az”*.

Ez játék közben az alábbi módon néz ki:



A fenti képernyőkép jelenik meg akkor, ha a játékos nem találta el a képet.

Ha viszont a felhasználó sikeresen választott, akkor az alábbi képernyőkép jelenik meg:

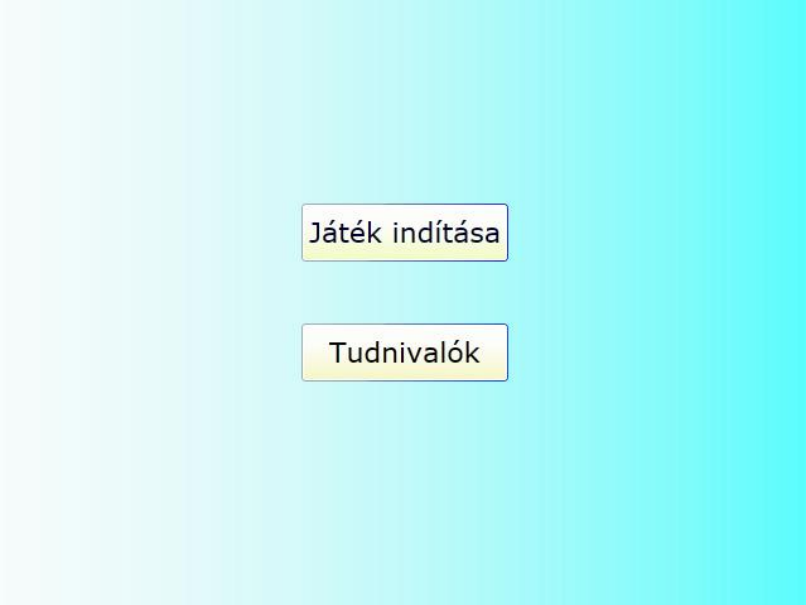


Ezen képernyőkép akkor jelenik meg, mikor a felhasználó sikeresen eltalálta a kívánt képet. Észrevehető ezen és az előbbi képernyőn, hogy az **Indul** interaktív gomb ismét aktív.

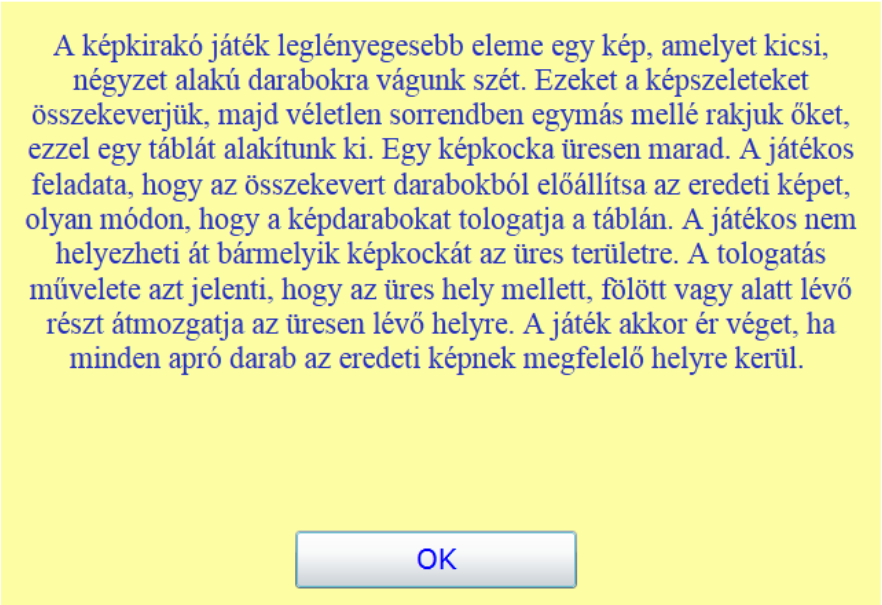
A képkirakó játék használata

A **Képkirakó** memóriajáték betöltése után a kezdő képernyőn két gomb közül választhat a felhasználó:

- ***Játék indítása***
- ***Tudnivalók***

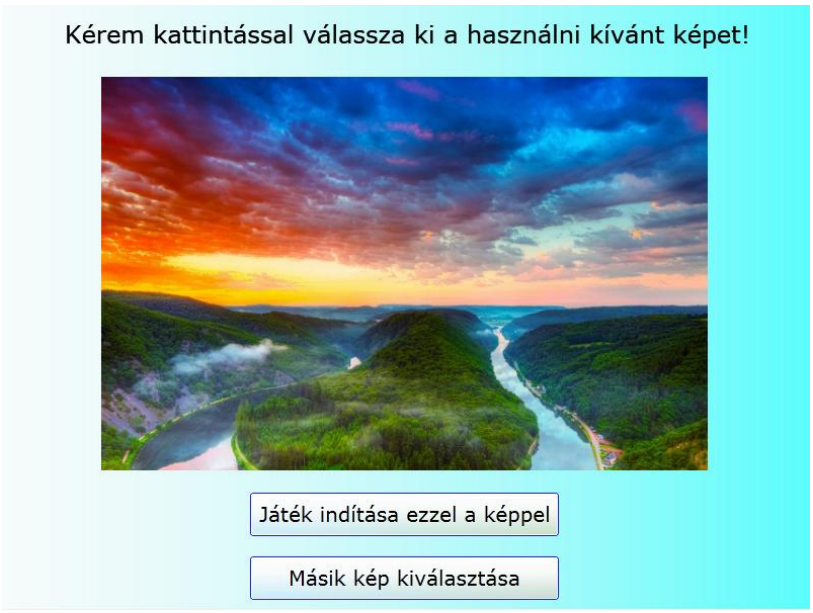


Amennyiben a ***Tudnivalók*** gombra kattintunk, megjelenik egy másik képernyő melyen elolvashatjuk a játék leírását és a szabályokat.



Az ***OK*** gombra kattintva visszatérünk a kezdő képernyőhöz.

Ha a ***Játék indítása*** gombra kattintunk betöltődik egy képkollekció, amely 20 darab képből tevődik össze szorosan egymás mellé rakva. A felhasználónak ki kell választania a neki tetsző ábrát, majd az egér bal gombjával az adott képre kattintva felnagyíthatja azt, annak érdekében, hogy nagyobb felbontásban, több részletet megfigyelve eldöntse, hogy biztosan ezzel a képpel szeretné-e lejátszani a játékot.



Amikor egy kép nagyításra kerül, a felhasználói felületen megjelenik 2 új funkció:

- ***Játék indítása ezzel a képpel***
- ***Másik kép kiválasztása***

Ha egy kép nem nyerte el tetszésünket, akkor a ***Másik kép kiválasztása*** gombot kell választanunk. Ekkor visszakapjuk az előbb látott, sok képből álló gyűjteményt, és kiválaszthatunk egy másik ábrát.
Ha megtaláltuk a számunkra szimpatikus fotót, a ***Játék indítása ezzel a képpel*** nyomógombra kattintva megjelenik egy újabb képernyő, melyen a játék nehézségi szintjét választhatjuk ki.
Két fokozatot kínál számunkra a program:

- ***Könnyű***
- ***Nehéz***

A ***Könnyű*** nehézségi szint kijelölése esetén, a játékosnak egy 3x3-as táblán kell a megfelelő helyre tologatnia a képdarabokat. Ekkor a program 9 darabra bontja az ábrát. A ***Nehéz*** fokozat kiválasztása esetén pedig egy 4x4 méretű táblát tölt be az alkalmazás, azaz 16 db képkockát kapunk.
A háttérben a játékhoz kijelölt kép jelenik meg. Amennyiben meggondoljuk magunkat és mégsem ezen kép darabjainak kirakásával szeretnénk foglalatoskodni, akkor a ***VISSZA*** nyílra kattintva visszatérhetünk a kiválasztó képernyőhöz.

A konkrét játék képernyőjének megjelenése után egy animációt figyelhetünk meg. Ezután a felület bal alsó sarkában láthatjuk az apró darabokra bontott képet, melynek darabjai össze vannak keverve. Ezeket a képkockákat a megfelelő helyükre kell juttatnunk. Ahhoz, hogy ezt a feladatot teljesítsük, a jobb felső sarokban segítségünkre lesz az ábra eredeti példánya, amelyen semmilyen fajta módosítást, darabolást nem végez a program.

A bal felső sarokban tájékoztat az alkalmazás az ***aktuális nehézségi szint***ről és a ***lépések számát*** is kiírja. Ha egy új játékot szeretnénk kezdeni kattintsunk az ***Új játék*** feliratú gombra.

A játék menete a következő:

- Betöltésnél van egy üres négyzet a felbontott kép jobb alsó sarkánál.
- Következő lépésként rá kell kattintanunk az egér bal gombjával a tőle balra vagy a felette levő képdarabra. Ekkor attól függően, hogy melyik négyzetre kattintottunk, az adott darab az üres helyre ugrik és az ő helye válik képmentessé.
- Ezután mindig az üres hely mellett, alatt vagy felett levő képkockákra kattinthatunk, ezáltal mozgatjuk a darabokat.

A cél, hogy minden részt a megfelelő pozícióba helyezzünk el. Ehhez használnunk kell memóriánkat és gondolkozásunkat.
Ha megfelelő helyre sikerült juttatni minden képrészt, akkor a következő nyertes üdvözlő képernyő jelenik meg:

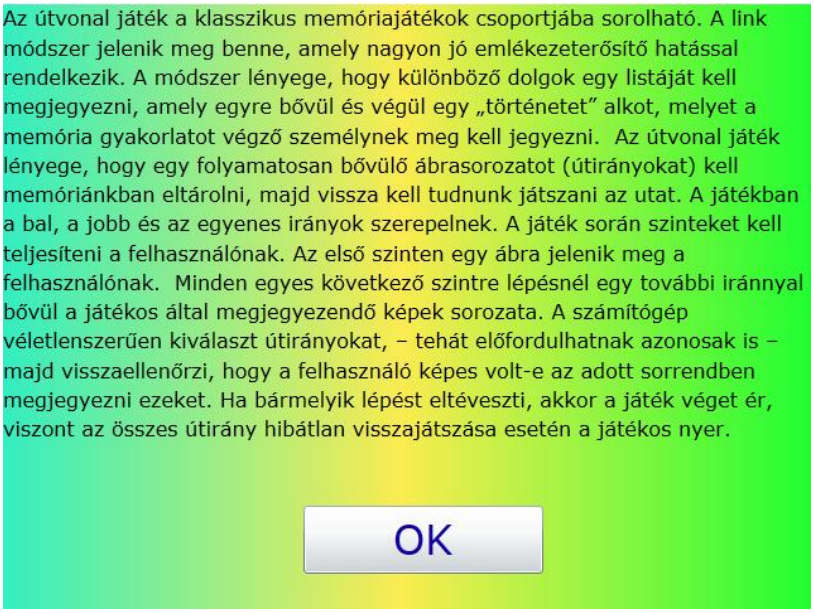
Ha itt az ***Új játék*** feliratú nyomógombra kattintunk az egérrel, kezdhetünk egy teljesen új játékot, akár egy másik képpel.

Az Útvonal játék használata

A ***Útvonal*** memóriajáték betöltése után a kezdő képernyőn két gomb közül választhat a felhasználó:

- ***Játék***
- ***Tudnivalók***

Amennyiben a ***Tudnivalók*** gombra kattintunk, megjelenik egy másik képernyő melyen elolvashatjuk a játék leírását és a szabályokat.

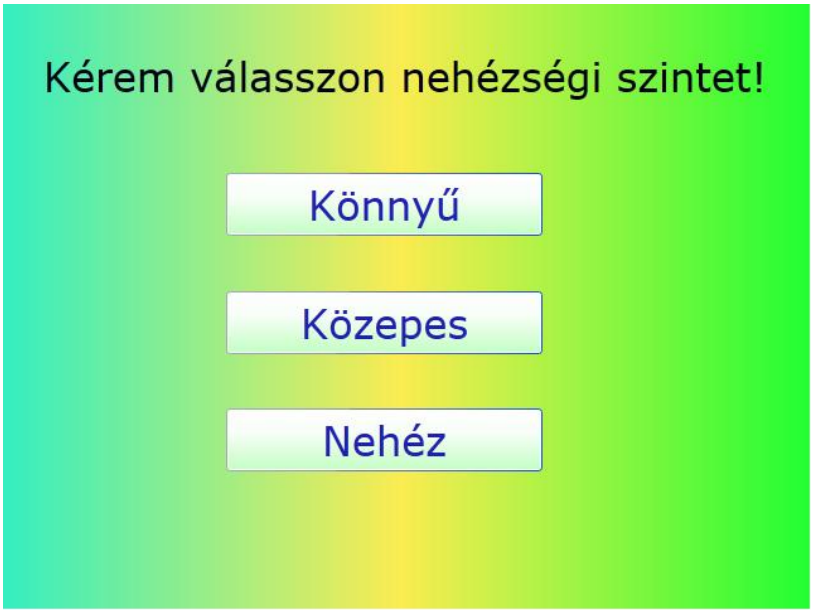


Az ***OK*** gombra kattintva visszatérünk a kezdő képernyőhöz.

A ***Játék*** nyomógombra kattintva megjelenik egy újabb képernyő, melyen a játék nehézségi szintjét választhatjuk ki.

Három fokozatot kínál számunkra a program:

- ***Könnyű***
- ***Közepes***
- ***Nehéz***

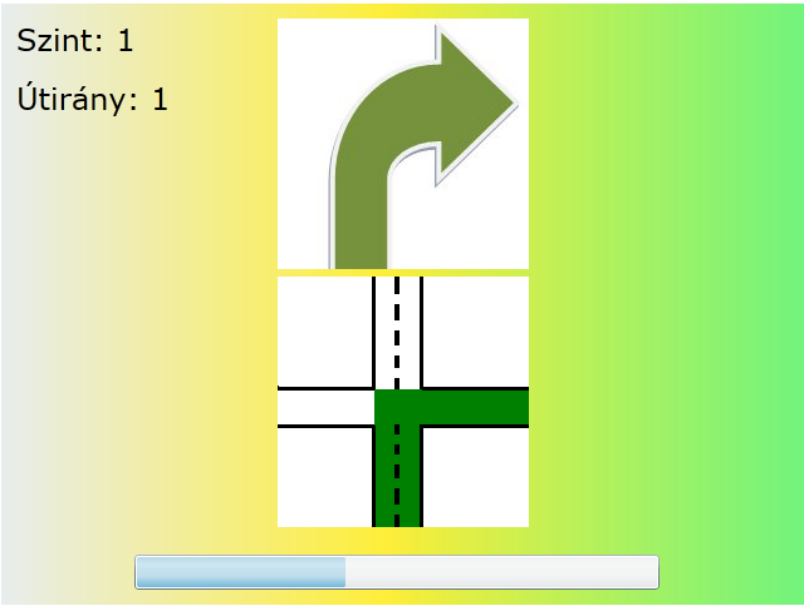


A nehézségi szintek kiválasztása során a program a megjegyzéshez adott időtartam értékét állítja be. A ***Könnyű*** nehézségi szint kiválasztása esetén a játékos ***5 másodperc***ig nézheti a megjelenő útirányt. A ***Közepes*** fokozat kijelölése esetén a megjegyzési időtartam ***3 másodperc***re módosul. A ***Nehéz*** feliratú nyomógombra kattintva a felhasználónak ***1 másodperc*** áll rendelkezésre, hogy megjegyezze az adott irányt.

A nehézségi szint kiválasztása után megjelenik egy újabb képernyő, amely már a konkrét játék felülete. A bal felső sarokban a játékos tájékoztatást kap az aktuális játékszintről. Alatta látható az adott útirány sorszáma, amely a szintek növekedésével párhuzamosan egyre nagyobb sorszámig fut, és ez is segít abban, hogy a felhasználó a sorrendet is könnyebben meg tudja jegyezni.

A felület közepén megjelenik két kép:

- a ***felső kép*** egy nyilat ábrázol
- az ***alsó kép*** egy útkereszteződést mutat, amelyben az adott útirány zöld színnel kiszínezésre kerül. Ez segít az asszociációban és a vizuális megjegyzésben.



A képernyő alsó részében egy sáv látható, amely folyamatosan feltöltődik. Ez ábrázolja a megjegyzésre biztosított időtartam még hátralevő részét, ezáltal a játékos tudhatja, hogy mikor veszi el az irányokat a program.

Ha az idő lejárt, akkor megjelenik egy újabb képernyő, melynek tetején az ***Ismételje meg az utat!*** felirat olvasható. A felületen látható három ábra, amely a három útirányt ábrázolja. A játékos feladata, hogy megismételje az alkalmazás által mutatott utat.

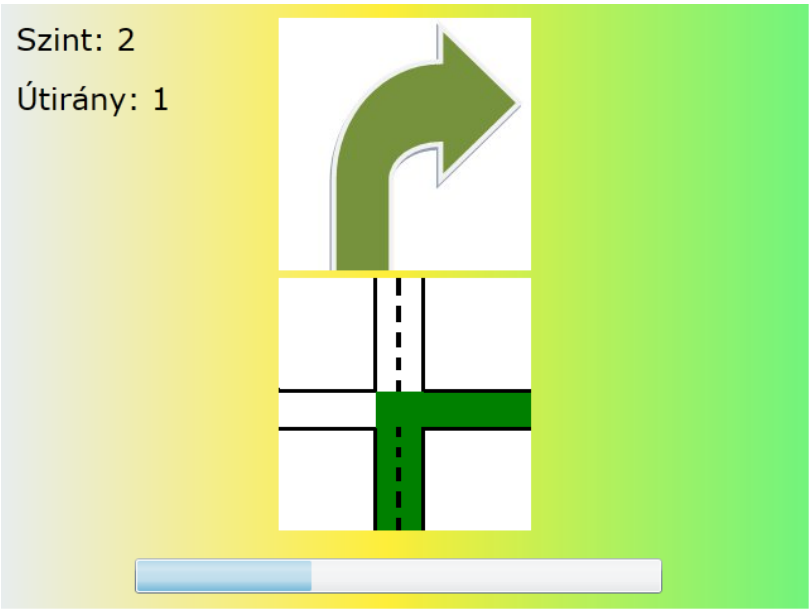
Ezt úgy teheti meg, hogy a megfelelő nyílra viszi az egér mutatóját, ekkor a nyíl színe megváltozik, így láthatóvá válik, hogy a játékos melyiket választja ki.

Ezután két dolog következhet be:

- a játékos eltalálta a megfelelő útirányt
- a játékos eltévesztette az útirányt

Amennyiben a felhasználó sikeresen megjegyezte és megismételte az útirányt, a képernyő középső részén megjelenik egy zöld pipa, amely jelzi a sikerességet, majd újra visszatér az előző felület.

A ***Szint*** feliratnál láthatjuk, hogy a következő pályára lépett az alkalmazás. A működés az előzőekben látottakhoz hasonló, annyiban különbözik, hogy már több irányt kell megjegyezni és megismételni.



A játékban összesen 10 pályán kell végighaladni. A szinteken való lépkedéssel párhuzamosan egyre bővül a megjegyezendő útirányok száma. Ha a felhasználó az összes pályát sikeresen teljesíti, akkor megnyerte a játékot és megjelenik egy újabb képernyő, amely tájékoztatja erről az eseményről és lehetőséget ad újabb játék kezdésére. Ez a ***Játék ismétlése*** nyomógombra kattintva valósítható meg.

Amennyiben bármelyik irányt eltéveszti a játékos, akkor az alábbi képernyő jelenik meg, és ha a felhasználó a ***Játék ismétlése*** nyomógombra kattint, akkor lehetőség nyílik a játék megismétlésére:

IX. Összefoglalás

Dolgozatunk célja volt, hogy olyan memóriajátékokat hozzunk létre, melyek jó memória gyakorlatoknak számítanak, és segítséget nyújtanak a memória fitten tartásában, ezáltal csökkentve az időskori memóriaromlás kialakulásának veszélyét.

Munkánk során létrehozásra került négy darab memória játék, amelyeket a Microsoft .NET 4.0 keretrendszer eszközeivel valósítottunk meg Silverlight 4.0 technológiát alkalmazva, valamint az ezeket magában foglaló weboldal, melynek fejlesztőeszközeit az ASP.NET technológia biztosította.

A memóriajátékok területe egy szinte kimeríthetetlen témakör, így számtalan új fejlesztési irányt meg lehet még valósítani. A továbbiakban igyekszünk még több játékot kifejleszteni, melyek több memóriagyakorlatot is magukban foglalnak. Valamint tervezzük egy úgynevezett játék betöltő alkalmazás implementálását, ami lehetővé teszi megadott interfészek alapján létrehozott játékok migrálását a weboldalba, illetve az adatbázisba. Továbbá játékainkat animációkkal színesíteni fogjuk, hogy a játékmenet érdekesebb, izgalmasabb legyen. A memóriagyakorlatokat megvalósító játékokat minden korosztály használhatja, de elsődlegesen az idősödő emberek számára lettek létrehozva, ezért további fejlesztési irány lehet a még könnyebben használható felhasználói felület megtervezése és implementálása.

X. Köszönetnyilvánítás

Ezúton szeretnénk köszönetet mondani témavezetőnknek, Dr. Juhász Istvánnak a szakdolgozat elkészítéséhez nyújtott segítségéért, gyakorlati és elméleti tanácsaiért.

XI. Irodalomjegyzék

[1] TULVING E. Mälu. Tartu Ülikooli Kirjastus, 2002. P. 345.

[2] http://hu.wikipedia.org/wiki/Endel_Tulving

[3] http://www.mindwellness.eu/descargas/Handbook-HU.pdf

[4] IVÁN, L.: Az idősödés és időskor, mint az edzettség próbája. Szenior Könyvek, Győr. 7-19. (2005)

[5] www.who.int – WORLD HEALTH ORGANIZATION (2002) Active Ageing: A policy Framework.

[6] http://www.mindwellness.eu/

[7] http://www.memory-improvement-tips.com/

[8] http://mipszi.hu/cikk/110414-neurobic

[9] ÁRVAI ZOLTÁN, CSALA PÉTER, FÁR ATTILA GERGŐ, KOPACZ BOTOND, REITER ISTVÁN, TÓTH LÁSZLÓ: Silverlight 4 - A technológia, és ami mögötte van - fejlesztőknek, Jedlik Oktatási Stúdió Bt., 2011

[10] LAURENCE MORONEY: Microsoft Silverlight 3 Első könyv, SZAK Kiadó, 2009

[11] http://msdn.microsoft.com/hu-hu/ff380144

[12] http://devportal.hu/