Eötvös Loránd Tudományegyetem
Informatikai Kar
Programozási Nyelvek és Fordítóprogramok
Tanszék

LaTeX-alapú prezentációkészítő program

Pataki Norbert
Adjunktus

Varjú János
Programtervező Informatikus BSc

Budapest, 2015

Tartalomjegyzék

1. fejezet

Bevezetés

1.1. Témaválasztás

Mai modern világunk megkerülhetetlen részét képezi az informatika. Nincs ez másképp az oktatásban sem, ahol már nem csak az egyetemeken van jelen, hanem lassan az általános iskolákban is nélkülözhetetlenné válik. Az okostábláknak és az egyre olcsóbb projektoroknak köszönhetően előbb vagy utóbb minden órának fontos szerepét fogják képezni a számítógépek, ezért szükség van olyan alkalmazásokra, melyekkel könnyedén lehet készíteni szép és egységes bemutatókat.

Sokaknak még az elterjedtebb alkalmazások használata is nehézséget jelent a rengeteg stílus és beállítás miatt. Ennek kiküszöbölésére szolgál a LaTeX szövegformázó rendszer, melynek használatával elég csak a tartalmat kitöltenünk, a formázás szinte már magától megy.

Én is ismerek ilyen személyeket, ezért döntöttem úgy, hogy egy minőségi prezentációk készítésére alkalmas, ám mégis könnyen kezelhető, grafikus felülettel rendelkező programot hozok létre.

1.2. Mi is az a LaTeX?

A LaTeX egy általános célú dokumentumszerkesztő rendszer, amely nyomdai minőségű művek előállítására képes. Könyvek, diplomamunkák, internetes cikkek százezrei készülnek a segítségével. A LaTeX nyelvének legfontosabb eleme a tartalom és a vizuális megjelenés szétválasztása, amely a modern dokumentumszerkesztés alap-

elve. A szerzőnek elég az írás logikai struktúráját megadnia, nem kell külön a megjelenítéssel foglalkozni, ugyanakkor lehetséges ennek teljes mértékű testreszabása.

Sok egyéb mellett prezentációk készítésére is használható a LaTeX. Ennek legelterjedtebb módja a Beamer dokumentumtípus használata, mellyel PDF alapú bemutatókat hozhatunk létre. Ennek nagy előnye, hogy a végeredmény minden platformon levetíthető, és mindenhol ugyanúgy fog működni. Ezt használtam fel a szakdolgozatom elkészítésekor is.

2. fejezet

Felhasználói dokumentáció

2.1. Feladat

A feladat egy grafikus felülettel rendelkező alkalmazás, amely prezentációk készítésére, valamint a megtervezett slide-okból LaTeX forráskód, illetve PDF generálására alkalmas.

A program célja, hogy a grafikus felület jelenítse meg az éppen kiválasztott diát nagyobb méretben, és az eddig elkészítettek slide-ok listáját. A felhasználó a diák sorrendjét tetszőlegesen változtathassa, valamint bárhonnan törölhessen, és bárhova szúrhasson be újabbakat. Mindig csak az aktuális dia tartalma legyen szerkeszthető.

A program biztosítson lehetőséget szövegek, felsorolások, forráskódok, táblázatok, ábrák bevitelére és eltávolítására. A szövegek kinézete legyen testre szabható. Választhassunk többféle betűszín, betűméret és stílus közül. A forráskódok nyelvét több lehetőség közül lehessen kiválasztani. A táblázatok méretét, valamint szegélyeit szerkesztés közben bármikor módosíthassuk. A képeknél legyen lehetőség leírás és méretarány megadására. Minden elem rendelkezzen overlay értékkel.

A választható témákat és színsémákat, valamint a slide-ok közötti átmenet-animációkat jelenítsük meg külön ablakban.

A prezentációk legyenek elmenthetők és betölthetők, ezen kívül legyen lehetőség .tex illetve .pdf fájl készítésére a kész prezentáció alapján.

2.2. Célközönség

Az alkalmazás bárki számára hasznos lehet, ha egyszerűen és gyorsan szeretne tetszetős prezentációkat készíteni.

Azoknak, akik kevesebb vagy akár egyáltalán nem rendelkeznek LaTeX ismeretekkel, csak egy egyszerűbb grafikus felület használatát kell elsajátítaniuk ahhoz, hogy egységes, minőségi bemutatókat szerkeszthessenek.

Azok számára is előnyös lehet a használata, akik már jártasak ilyen téren, ugyanis a program segítségével gyorsan generálhatnak részletes, további finomításokra alkalmas .tex forrásfájlt.

2.3. Rendszerkövetelmények, futtatás

A program hibátlan működéséhez szükséges minimális rendszerkövetelmények a következők:

- Windows operációs rendszer
- .NET keretrendszer 4.5
- A legújabb MiKTeX disztribúció teljes verziója (kb. 500MB)

Indításhoz másoljuk a lemezen található BeamerSzerkeszto mappát a számítógépre, majd futtassuk a benne lévő BeamerSzerkeszto.exe -t. Amennyiben előzetesen nem lett .NET keretrendszer telepítve, a felugró ablak utasításait követve töltsük le, majd telepítsük azt, és próbálkozzunk újra!

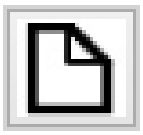
2.4. A program használata

2.4.1. Menü

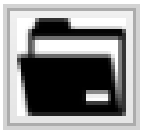
A Menüből lehet a fájlkezelő műveleteket végezni, valamint a prezentáció egészére vonatkozó módosításokat végrehajtani, és a program névjegyét megjeleníteni.

A Fájl menü

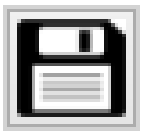
A Fájl menü minden pontjához tartozik egy ugyanazt a funkciót ellátó gomb a felületen, valamint billentyűparancs is van rendelve hozzájuk, ezzel is gyorsítva a program használatát. Új projekt kezdésekor, prezentáció megnyitásakor és kilépés előtt is meg kell erősítenünk szándékunkat egy felugró ablakban.



Új *(Ctrl+N)*: Üres prezentáció létrehozása. Minden beállítás visszaáll alaphelyzetbe. Az aktuális projekt nem mentett változtatásai elvesznek.



Megnyitás *(Ctrl+O)*: Elmentett projekt megnyitása. Ha a betöltendő fájl hibás, hibaüzenetet kapunk és semmi nem töltődik be. Az aktuális projekt nem mentett változtatásai elvesznek.



Mentés *(Ctrl+S)*: Az aktuális projekt mentése egy .prez kiterjesztésű fájlba. A beszúrt, de nem kitöltött szövegek, felsorolások és forráskódok nem kerülnek mentésre.

Fordítás *(Ctrl+F)*: LaTeX forráskód és PDF generálása az éppen szerkesztett projekt aktuális állapota alapján. A menüpontra kattintás után először ki kell választanunk a kimeneti dokumentumok helyét. Az elkészített fájlok neve megegyezik a projekt nevével.

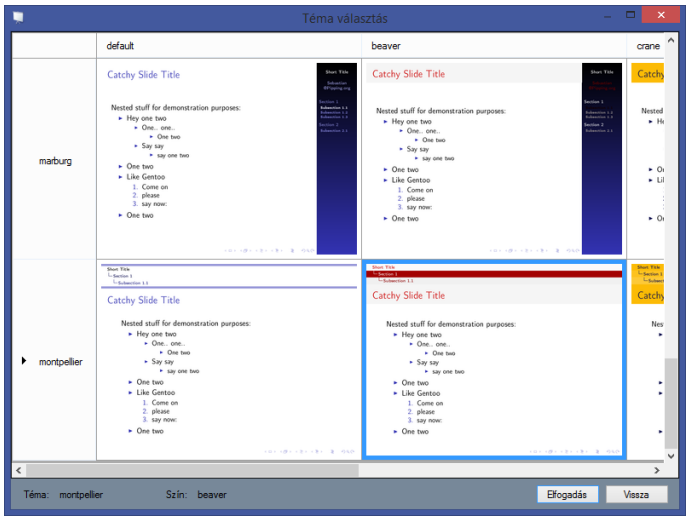
Kilépés: A program bezárása. Az aktuális projekt nem mentett változtatásai elvesznek.

A Tervezés menü

A Tervezés menüben állíthatjuk be a teljes prezentációra vonatkozó tulajdonságokat. Új projekt indításakor az itt található értékek alaphelyzetbe kerülnek.

Téma választás *(Ctrl+T)*: Egy új ablakot nyit meg a témaválasztáshoz.

A megjelenő táblázat sorai egy-egy témát, oszlopai egy-egy színsémát jelölnek. A cellákban az így kapott párok kinézetéről láthatunk mintaképeket [4]. Választani közülük az egér, illetve a billentyűzet nyilai segítségével tudunk. Az ablak bal alsó sarkában láthatjuk az aktuális téma és szín LaTeX-beli elnevezését. Választásunkat az Elfogadás gombra való kattintással tudjuk megerősíteni.

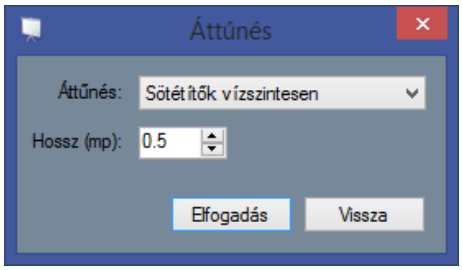


2.1. ábra. A témaválasztó ablak, montpellier témával és beaver színsémával

Áttűnések *(Ctrl+A)*: Új dialógus ablakot nyit meg az áttűnés-beállításokhoz.

Áttűnés: Legördülő menüből választhatjuk ki a nekünk megfelelő keretek közötti átmenet-animációt.

Hossz: Az átmenet-animáció hosszát adhatjuk meg másodpercekben. A 0.1 és 10.0 közötti értéket kézzel írhatjuk a számdobozba, vagy a görgő segítségével növelhetjük, illetve csökkenthetjük.



2.2. ábra. Az áttűnés választó ablak

Névjegy

Információk a programmal kapcsolatban: az alkalmazás és a szerző neve, valamint rövid leírás a használatról.

2.4.2. A szerkesztő felület

A program indításakor egyből a szerkesztőfelületet látjuk, egy üres projekttel. A képernyő bal oldalán található meg az eddig hozzáadott diák listája, középen pedig az aktuális dia, nagyobb méretben. A diáknak két típusa van: címdia és normál dia.

A címdia

Minden prezentáció pontosan egy címdiát tartalmaz, és mindig ez áll a bemutató legelején. Nem helyezhető át (diák áthelyezéséről bővebben később) és nem is törölhető. Öt sort tartalmazhat legfeljebb, melyek kitöltése opcionális. Ezek sorban a cím, az alcím, a szerző, az intézmény neve, valamint a dátum. A prezentáció stílusától függően ezek megjelenhetnek a fej- illetve láblécben is. A címdia szövegeinek formátumát csak a téma- és színbeállításokkal lehet változtatni. Szerkesztéskor mindegyiket egy sorban jelenítjük meg, de amennyiben valamelyik túlnyúlna a dia korlátain, a generált PDF-ben már több sorba tördelve fog megjelenni.

2.3. ábra. A címdia kinézete

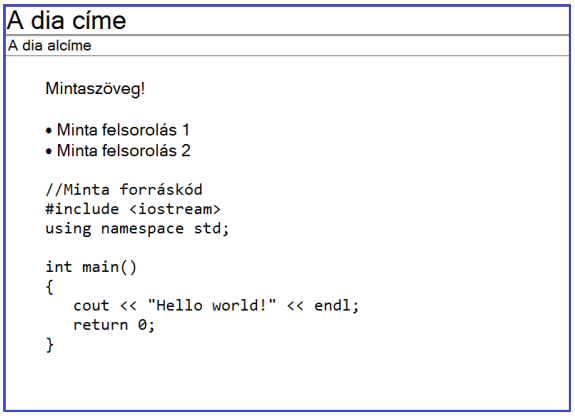
A normál diák

Egy prezentáció tetszőlegesen sok normál diát tartalmazhat, melyek sorrendje bármikor változtatható. Bármelyik alá szúrhatunk be újabbakat, és mindegyiket szabadon törölhetjük is.

Egy normál dia három fő részből áll. Ezek a cím, az alcím és a tartalom. Mindegyik kitöltése opcionális, azonban ha nincs megadva cím, PDF készítésekor az alcím sem fog megjelenni. Ugyanez fordítva azonban már nem mondható el. A címet és

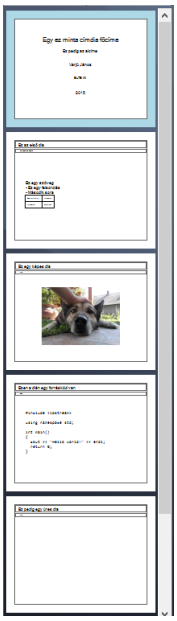
az alcímet egy sorban jelenítjük meg, de ha valamelyik így nem férne ki egy diára, akkor a lefordított PDF-ben már több sorba tördelve fog megjelenni.

Egy dia tartalmát az alkalmazás tetején található gombok segítségével tudjuk feltölteni. Lehetőségünk van szövegek, felsorolások, forráskódok, táblázatok és képek bevitelére (bővebben lásd 2.4.4). Ezekből egy diára tetszőlegesen sokat tudunk felvinni, de akárcsak az elkészített PDF-ben, itt sem fog megjelenni, ami már nem férne ki. Lehetőségünk van beállítani minden diának a függőleges elrendezését. Alapesetben az összes dia középre van igazítva.



2.4. ábra. A normál diák kinézete

A dialista



A képernyő bal oldalán, egy oszlopban található a prezentációhoz hozzáadott diák listája, és a mozgatásukhoz szükséges vezérlők. A lista legelején mindig a címdiát láthatjuk, alatta helyezkednek el sorban a normál diák.

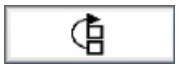
Itt van lehetőségünk a diák közötti váltásra. Ezt megtehetjük a görgő segítségével, a *Ctrl* + fel és le nyilakkal, valamint a kiválasztani kívánt diára való bal kattintással. Az aktuálisan kiválasztott dia kék, míg a többi fehér kerettel rendelkezik.

Ha már nem férne ki a képernyőre az összes dia, akkor a lista jobb oldalán megjelenő görgetősávval tudjuk mozgatni a látható területet.

A listán található diák folyamatosan frissülnek, az aktuális dia változtatásait tükrözve.

A diák sorrendje módosítható, egy lépésben két szomszédos diát lehet felcserélni. Ezt a dialista fölött található két gomb segítségével tudjuk megtenni. Válasszuk ki először az áthelyezni kívánt diát, majd kattintsunk a megfelelő nyomógombra.

Megjegyzés: Ha a lista alján vagy tetején vagyunk, már nem tudjuk az aktuális diát tovább mozgatni lefelé, illetve felfelé. Erre külön hibaüzenet is felhívja a figyelmünket. Ugyanez elmondható akkor is, ha címdiát próbáljuk meg elmozgatni.

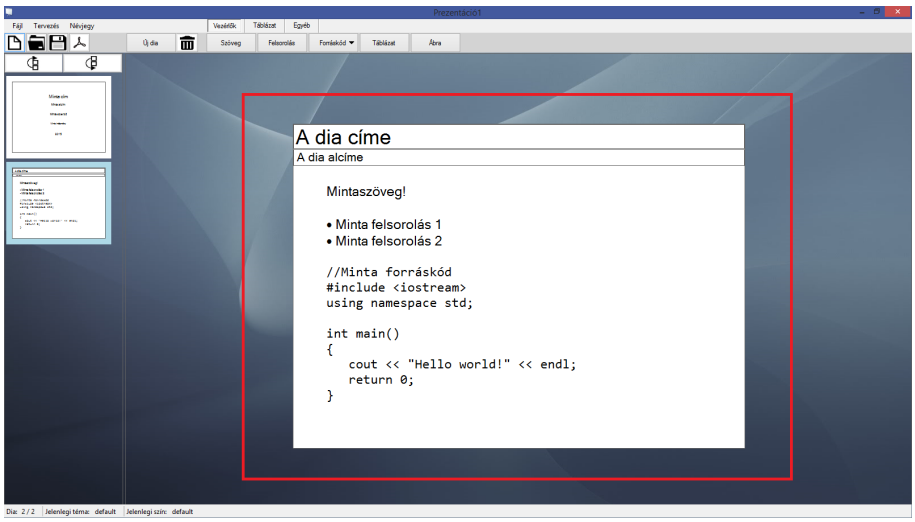


2.5. ábra. Dia mozgatás felfelé

2.6. ábra. Dia mozgatás lefelé

Az aktuális dia

A képernyő közepén, nagy méretben megjelenő slide. Attól függően, hogy a dialistán épp mit választottunk ki, megjeleníthet cím- vagy normál diát is. Itt tudjuk végezni a diák tartalmának szerkesztését, és az újonnan felvett elemek is itt jelennek meg először.

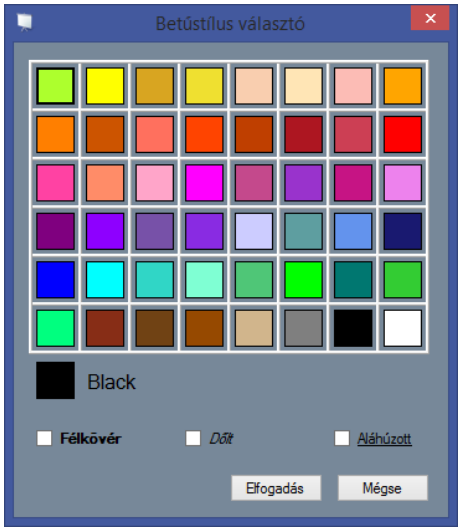


2.7. ábra. Az aktuálisan szerkeszthető dia (pirossal bekeretezve)

A stílus választó:

Egyszerű szövegek és felsorolások esetén lehetőségünk van azok stílusának beállítására is. Ehhez először kattintsunk rájuk jobb gombbal, majd válasszuk ki a Stílus opciót. Ekkor a képen látható ablak fog megjelenni a képernyő közepén.

Az aktuális betűszín és annak LaTeX-beli elnevezése a „Félkövér” felirat felett látható. Ha másikat szeretnénk, kattintsunk a színtáblázat megfelelő elemére. Ekkor az előbb említett mező is frissülni fog, az új színt mutatva.
Lehetőségünk van még annak az eldöntésére is, hogy az adott szöveg **félkövér**, *dőlt* vagy <u>aláhúzott</u> legyen-e. Hogy ezt megtegyük, kattintsunk az adott tulajdonság előtti jelölődobozokba. Azok a tulajdonságok lesznek érvényben, melyek előtt pipa szerepel.

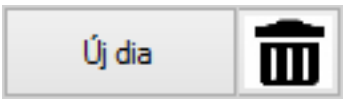


2.8. ábra. A stílus választó ablak

2.4.3. Diák hozzáadása, törlése

Új diák felvételére és a nem kívántak törlésére a vezérlő menütől balra található két gomb szolgál. Mind a beszúrás, mind pedig a törlés csak a normál diákra vonatkozik.

Az újonnan hozzáadott diák mindig a jelenleg kiválasztott alá szúródnak be, és ezt követően a program egyből át is vált rájuk, törléskor pedig mindig a törölt dia előttire áll át a fókusz.

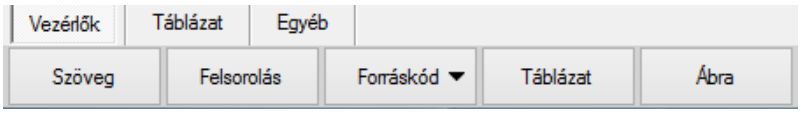


2.9. ábra. Az új dia beszúrása és a törlés gomb

2.4.4. A Vezérlő menü

A dia törlés gombtól jobbra található egy tabcontrol, amely a diák szerkesztéséhez szükséges vezérlőket tárolja. A program indításakor a vezérlő fül elemei láthatók benne, de bármikor átválthatunk a Táblázat vagy az Egyéb fülre is.

A vezérlő fül gombjai csak akkor érhetők el, ha az éppen szerkesztett slide nem a címdia.



2.10. ábra. A vezérlők menü gombjai

Szöveg beszúrása:

Kattintsunk a „Szöveg" feliratú gombra. Ekkor a dia közepén, vagy a többi - már korábban hozzáadott - vezérlő alatt megjelenik egy „Ide írj" felirat. Erre kattintva tetszőleges hosszúságú szöveg bevitelére van lehetőségünk. Amennyiben nem töltjük ki a szöveget, mentéskor és PDF generáláskor törlődni fog.

Ha a szövegre jobb gombbal kattintunk, egy felugró menüben két lehetőség közül választhatunk: Stílus vagy Törlés. Előbbire kattintva a szöveg színét és megjelenését állíthatjuk egy külön ablakban (erről korábban már olvashattunk), utóbbival pedig törölhetjük a szöveget a diáról.

Felsorolás beszúrása:

A „Felsorolás" feliratú gombra kattintással egy új felsorolást kezdhetünk az éppen megjelenített dián. A felsorolás elemeit a szerkesztőben fekete pontok jelölik, de témától függően ez eltérhet a lefordított fájlban. Új listaelemet hozzáadni az Enter billentyű leütésével tudunk.

A sima szövegekhez hasonlóan, a felsorolásra jobb gombbal kattintva lehetőségünk van a stílus módosítására, valamint törlésre.

Forráskód beszúrása:

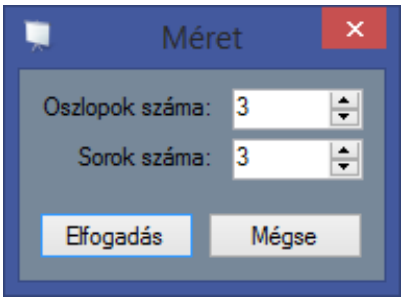
Forráskód beszúrásához először a hozzá tartozó gombra kell kattintani, majd a megjelenő menüből kiválasztani a kód nyelvét. A generált PDF-ben a választástól

függően lesznek kiemelve a nyelvhez tartozó kulcsszavak, szövegek és kommentek.

Az előzőektől eltérően, forráskód esetén csak törlésre van lehetőségünk, stílus kiválasztására nem.

Táblázat beszúrása:

A „Táblázat” gombra kattintás után egy felugró ablakban kell megadnunk a beszúrni kívánt táblázat oszlopainak és sorainak számát. Felső határ nincs megszabva, de vegyük figyelembe, hogy 20 oszlopnál és 15 sornál több normál betűmérettel nem fér ki egy diára.



2.11. ábra. Táblázat mérete

A táblázat hozzáadása után a cellákat külön szerkeszthetjük, vagy akár üresen is hagyhatjuk. Egy cellában a LaTeX alapértelmezés szerint csak egy sor szöveg lehet. A táblázat túlnyúlhat a dia határain, de a kilógó részek a PDF változatban sem lesznek láthatók. A cellákra kattintás után elérhetővé válnak a Táblázat menü gombjai is, melyekkel a táblázat egészére vonatkozó módosításokat végezhetünk el (bővebben lásd 2.4.5).

Ábra beszúrása:

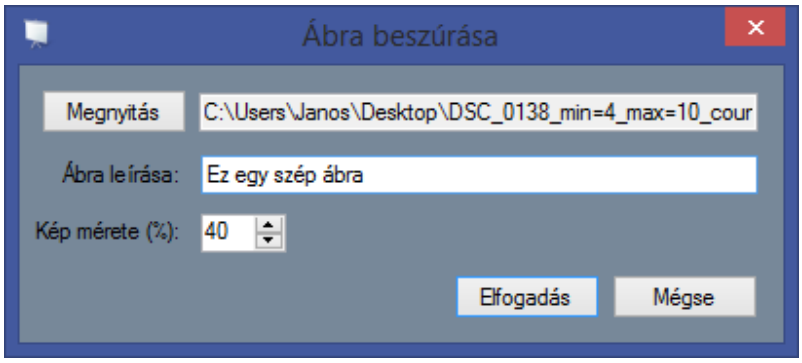
Az „Ábra” gombra kattintás után egy külön ablakban kell megadnunk a szükséges információkat.

„Megnyitás”: A megjelenő ablakban kikereshetjük fájljaink között a beszúrni kívánt képet. Támogatott formátumok: BMP, JPG, JPEG, PNG, TIF. Más kiterjesztésű fájlokat nem nyithatunk meg. A kép mérete legfeljebb 800*800 pixel lehet.

Ábra leírása: A lefordított PDF-ben a kép alatt közvetlenül megjelenő szöveg, „*X*. ábra. *LEÍRÁS*” alakban, ahol *X* a kép sorszáma, a *LEÍRÁS* pedig az általunk megadott sor.

Kép mérete (%): Azt tudjuk beállítani a számdobozba kézzel beírva, vagy a görgő segítségével, hogy a beszúrni kívánt kép szélessége hány %-a legyen a dia teljes szélességének. Értéke 1 és 100 között mozoghat.

Megjegyzés: Nagy méret megadása esetén lehetséges, hogy az ábra (vagy annak leírása) függőlegesen már nem fér rá a diára.

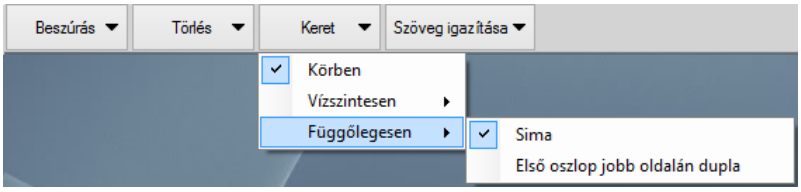


2.12. ábra. Ábra beszúrása

A beszúrt képekre jobb gombbal kattintva lehetőségünk van beállításaik módosítására, valamint törlésükre. Előbbi esetén ugyanazt a felugró ablakot láthatjuk, amit már a kép felvételekor is. Átállíthatjuk ekkor a leírást, a méretet, vagy akár egy teljesen másik képet is betallózhatunk az eddigi helyére.

2.4.5. A Táblázat menü

A dia törlés gomb mellett található tabcontrol második füle tartalmazza a táblázatok testreszabására szolgáló vezérlőket. Ezek akkor érhetők el, ha épp egy táblázat van kijelölve. Minden módosítás csak az adott táblázatra fog vonatkozni. Az egyes gombokra kattintva egy felugró menüből választhatunk a további módosítási lehetőségek közül.



2.13. ábra. A táblázat menü gombjai és a Keretnél megjelenő menü

Beszúrás

A beszúrás gomb segítségével tudunk a táblázathoz új sorokat, illetve oszlopokat adni. A felugró menü két eleme közül ennek megfelelően kell választanunk. Az új sorok és oszlopok rendre a táblázat alján és jobb oldalán jelennek meg.

Törlés

A törlés gombra kattintva lehetőségünk nyílik a táblázat legalsó sorát vagy legjobboldalibb oszlopát törölnünk. Előbbiért kattintsunk a felugró menüben az „Utolsó sor”, utóbbiért pedig az „Utolsó oszlop” feliratú elemre.

Keret

A keret menü segítségével tudjuk az aktuális táblázat celláinak szegélyét megváltoztatni. Öt előre megadott opció van, melyek közül bármelyiket ki- vagy bekapcsolhatjuk a megfelelő elemre való kattintással. Alapesetben minden cella körül sima szegély van.

- **Körben:** A táblázat peremén lévő szegélyt állítja.
- **Vízszintesen, Sima:** A táblázat sorait elválasztó vízszintes vonalakat kapcsolja ki vagy be.
- **Vízszintesen, Első sor alján dupla:** A táblázat első és második sora között dupla vonal jelenik meg.
- **Függőlegesen, Sima:** A táblázat oszlopai közötti vonalakat állítja.
- **Függőlegesen, Első oszlop jobb oldalán dupla:** A táblázat első és második oszlopa közé szúr be két vonalat.

Szöveg igazítása

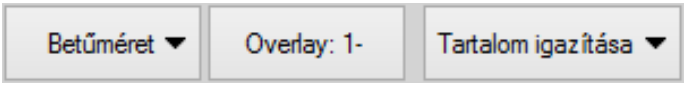
Mindig az aktuális táblázat éppen kiválasztott oszlopának celláira vonatkozik. Segítségével minden oszlopra külön megszabhatjuk, hogy jobbra, balra vagy középre legyenek igazítva a benne található szövegek.

Minta táblázat	Első	Második	Harmadik
Első	1	2	3
Második	2	4	6

2.14. ábra. Minta táblázat keret nélkül, különböző oszlop igazításokkal

2.4.6. Az Egyéb menü

A korábban már említett tabcontrol harmadik és egyben utolsó füle tartalmazza a maradék, a dia tartalmának módosítására szolgáló vezérlőket. Ezek közül az első kettőhöz csak akkor férünk hozzá, ha éppen egy tartalmi elem van kijelölve a slide-on (betűméret esetén ez az elem nem lehet kép), az utolsó viszont mindig elérhető, kivéve a címdia esetén.



2.15. ábra. Az Egyéb menü gombjai

Betűméret

A táblázatok módosításához hasonlóan először itt is ki kell jelölnünk egy szöveget, majd a „Betűméret" feliratú gombra kattintva egy felugró menüből választhatjuk ki, hogy mekkorák legyenek a kijelölt elem karakterei. Táblázat esetén az összes cella mérete külön módosítható, minden más esetben a szöveg egészére vonatkozik a választásunk.

Tíz különböző méret van, ezek fentről lefele növekvő sorrendben vannak. Az újonnan hozzáadott szövegek alapértelmezetten Normál nagyságúak.

Overlay

A LaTeX prezentációk úgynevezett keretekből állnak, melyek akár több diát is tartalmazhatnak. Egy kereten belül a diák címe és részben tartalmuk is megegyezhet. Például ha azt szeretnénk, hogy egy adott szöveg 10 diának is megjelenjen az elején, nem kell mindet külön létrehoznunk és hozzáadnunk az adott sort, hanem elég, ha az overlay értékét helyesen beállítjuk.

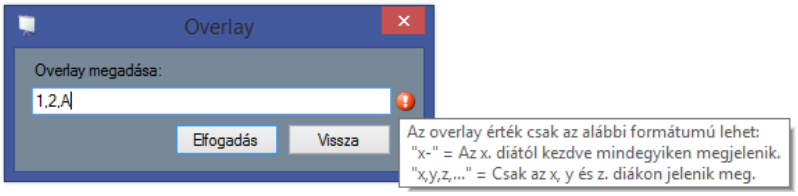
Először is jelöljük ki a módosítani kívánt vezérlőt! Ekkor elérhetővé válik az „Overlay:" feliratú gomb, és megjelenik rajta az elem jelenlegi overlay értéke. Ha ezen állítani szeretnénk, kattintsunk az előbb említett gombra. Egy felugró ablak fog megjelenni, amely csak egy szövegdobozt tartalmaz, a már a gombon is megjelenő értékkel. Ennek módosításával befolyásolhatjuk, hogy az adott szöveg, táblázat vagy kép a keret melyik diáin jelenik meg.

Az overlay-nek kötött formátuma van, és amennyiben ezt nem tartjuk be, a szövegdoboz mellett piros felkiáltójel jelenik meg. Ha fölé visszük a mutatót, rövid súgót láthatunk arról, hogy milyen szabálynak kell megfelelnie a bevitt értéknek. Két formában adható meg:

- ***„X-”***: Ekkor a kereten belüli X. diától kezdve mindegyiken látható lesz.
- ***„X, Y,..., Z”***: Ekkor csak az X., Y., és Z. diákon jelenik meg (a „ ... ” helyén tetszőlegesen sok szám állhat, vesszővel elválasztva).

(X, Y és Z 1-nél nagyobb egész számokat jelölnek)

Egy keret pontosan annyi diából áll, amekkora az általa tartalmazott vezérlők overlay értékeinek maximuma.



2.16. ábra. Hibásan megadott overlay érték

Tartalom igazítás

Az Egyéb menü utolsó gombjával a normál diák tartalmának függőleges igazítását szabályozhatjuk. Minden dia alapértelmezetten középre van rendezve, de a „Tartalom igazítása” gombra kattintás után lehetőségünk van a slide tetejére vagy aljára vinni a felvett elemeket.

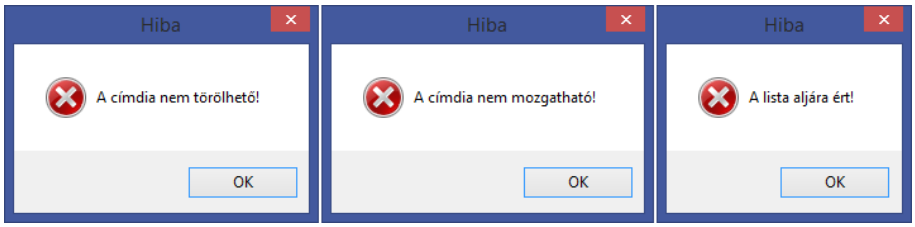
2.4.7. Az állapotsor

Az alkalmazás alján található információs sor három fő dolog megjelenítésére szolgál:

- **Dia: X/Y**: Ahol X jelöli az aktuális dia sorszámát, Y pedig az összes dia számát.
- **Jelenlegi téma**: Az éppen kiválasztott téma elnevezése.
- **Jelenlegi szín**: Az éppen kiválasztott színséma elnevezése.

2.4.8. Hibaüzenetek

Hibaüzenetet kapunk, ha címdiát szeretnénk törölni, vagy elmozgatni, valamint ha egy normál diát szeretnénk az utolsó helynél lejjebb vinni.

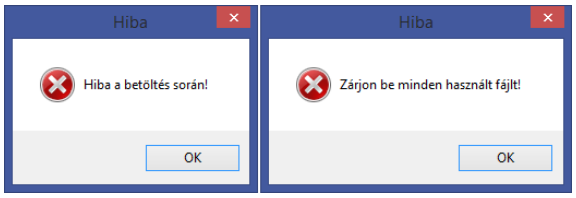


2.17. ábra. A dialistával kapcsolatban fellépő hibaüzenetek

Különböző hibajelzéseket kaphatunk kép beszúrásakor is, például ha nem adtunk meg elérési útvonalat, túl nagy a kép, vagy időközben már törölték.

2.18. ábra. Képek beszúrásánál fellépő hibaüzenetek

Hibák lépnek fel akkor is, ha például egy sérült fájlt próbálunk betölteni, vagy ha PDF generáláskor egy, a fordító számára szükséges fájl éppen meg van nyitva.



2.19. ábra. Betöltéskor és PDF generáláskor fellépő hibaüzenetek

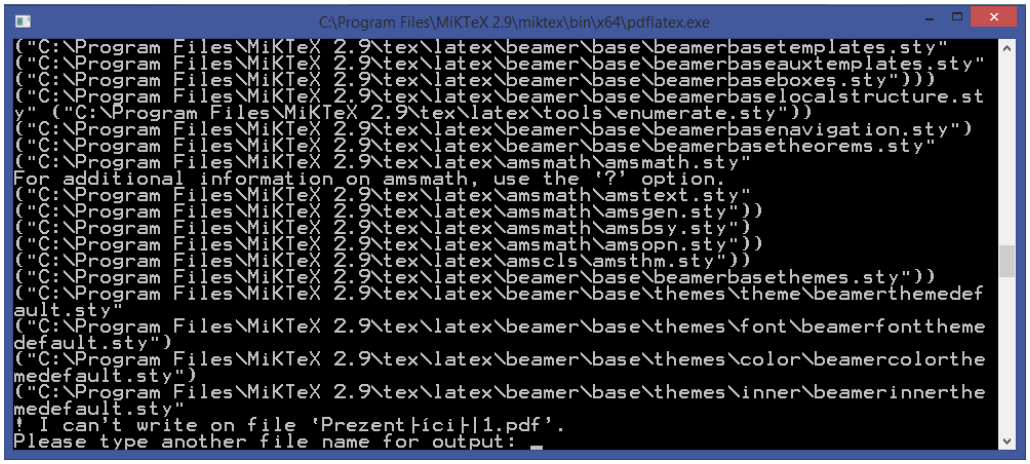
2.4.9. Hibaelhárítási előírások

A hibátlan futás érdekében szükséges, hogy a felhasználó betartson néhány, a LaTeX forrásfájl generáláshoz szükséges szabályt. Amennyiben nem így tesz, a fordítás során megjelenő konzol ablakban hibaüzenet lesz látható.

- A prezentációban használt forráskódok nem tartalmazhatnak ékezetes karaktereket.

- A beszúrt képek elérési útvonalában nem szerepelhetnek ékezetes vagy a LaTeX-ben speciális jelentéssel bíró karakterek (Ezek az alábbiak: # % { } _ ^ $ ~).

Fontos még ezeken kívül, hogy ha már létezik egy ugyanolyan nevű PDF fájl, mint amit készíteni szeretnénk, zárjuk be azt először, vagy amikor a konzolban futó fordító kéri, adjunk meg egy másik nevet.



2.20. ábra. Ha már meg van nyitva egy ugyanilyen nevű PDF fájl

3. fejezet

Fejlesztői dokumentáció

3.1. Megoldási terv

A feladatot ablakos alkalmazásként valósítjuk meg Microsoft .NET keretrendszerben és az abban található Windows Forms grafikus osztálykönyvtár segítségével **C#** nyelven.

3.1.1. Menü

Az ablak tetején helyezzük el a menüt, melyből bármikor kezdhetünk új projektet, megnyithatunk egy korábbit vagy elmenthetjük az aktuálisat. Innen van lehetőségünk még a PDF és TEX fájlok generálására, valamint kilépésre.

Egy másik menüpontban helyezzük el az éppen szerkesztett projekt egészére vonatkozó módosító lehetőségeket, úgymint a prezentáció témája és színsémája vagy a diák közötti áttűnés-animációk. Ezekhez a menüpontra kattintás után külön ablakot nyitunk.

A menüből érhető el a programmal kapcsolatos információkat tartalmazó névjegy is.

3.1.2. A szerkesztőfelület

A program indításakor üres projekttel nyílik meg a szerkesztőfelület. A fejlécben elhelyezünk négy gombot, melyek a menü fájlkezelésre szolgáló elemeivel megegyező funkciót látnak el. Mellettük találhatók a diák beszúrásáért, törléséért és szerkesz-

tésükért felelős vezérlők. Utóbbiakat csoportokba rendezzük, így kevésbé széles képernyőkön is kényelmesen látszik minden gomb. Külön blokkban találhatók a diák elemeinek hozzáadására szolgáló, a táblázatokat módosító, és az egyéb funkciókat ellátó gombok.

A felület bal oldalán található a slide-ok megjelenítésére szolgáló lista, a mozgatásukhoz szükséges vezérlőkkel együtt. A szerkeszteni kívánt diát kattintással vagy a görgő segítségével tudjuk kijelölni, ekkor annak kerete kék színű lesz, minden másiké pedig fehér.

A képernyő közepén található a tényleges szerkesztő felület. Ez tartalmazza a jelenleg kijelölt slide-ot felnagyítva.

A képernyő alján állapotsort helyezünk, amely a felületen nem látható dolgok megjelenítésére szolgál. Ezek az aktuálisan módosított dia sorszáma, az összes eddig hozzáadott dia száma, és a jelenlegi téma és színséma neve.

A szerkesztőfelületen jelennek meg a bonyolultabb műveletekhez tartozó felugró ablakok is, úgymint az ábrák beszúrására szolgáló ablak, vagy a témaválasztó.

3.1.3. Architektúra

Az alkalmazást modell/nézet architektúrában építjük fel. A nézet hozzáfér a modell publikus függvényeihez, de a modell csak jelzéseket küldhet a nézet számára. Ez lehetőséget nyújt arra, hogy ugyanezt a modellt egy másik nézethez is felhasználjuk változtatások nélkül.

A modell

A modellben a projekt szöveges megfelelőjét tároljuk el. Azért, hogy jobban áttekinthető legyen, külön modell osztályt hozunk létre a normál- és címdiáknak, valamint a teljes prezentációnak.

A prezentáció modellje tartalmazza a globális információkat, valamint a beszúrt diamodellek listáját. A címdia modell felépítése egyszerű, csak a rajta megjelenő öt sor szövegét tárolja el, ellenben a normál diák modelljének minden vezérlő szöveges reprezentációját (stílus, szín, sorok) tartalmaznia kell az egyéb tulajdonságokon kívül.

A modellben kezelni kell a frissítést a nézet alapján, valamint itt kell elvégezni a TEX fájl generálását és annak fordítását is. A modell felelős ezen kívül a mentés

fájl elkészítéséért és a betöltés elvégzéséért.

A nézet

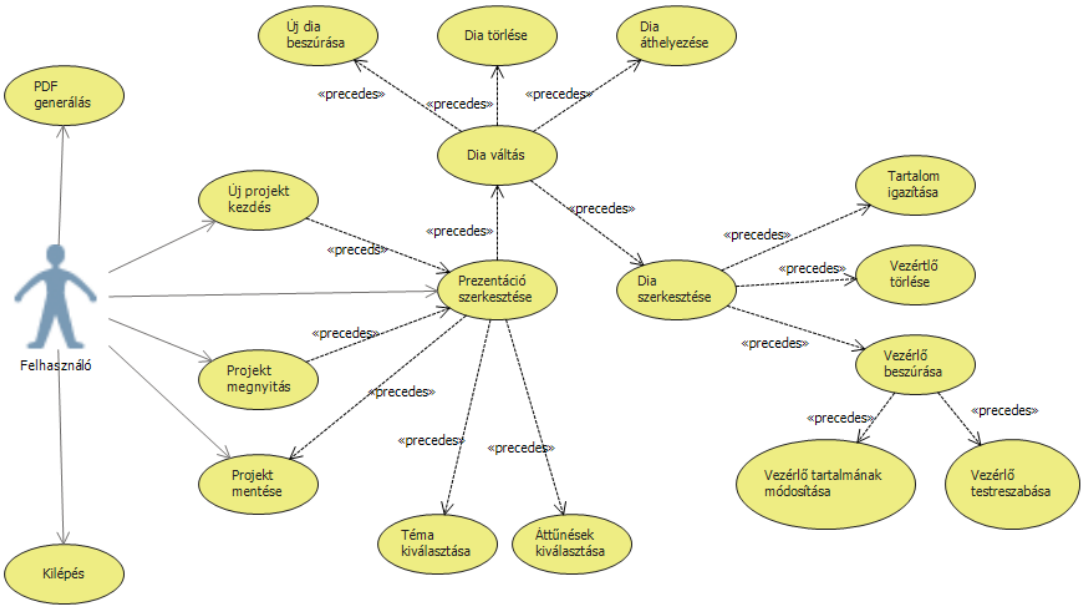
A nézet végzi a grafikus elemek megjelenítését, mint például a diák vagy a rajtuk található vezérlők. Ezek létrehozásához származtassunk saját osztályokat a Windows Forms osztálykönyvtár már előre elkészített típusaiból, a még szükséges adattagok és műveletek hozzáadásával.

Külön osztályokat hozunk létre az összetettebb műveletek ellátására szolgáló felugró ablakoknak is, valamint minden egyéb grafikai elemnek, amelynek nincs nekünk megfelelő előre definiált változata.

3.2. Megvalósítás

3.2.1. A felhasználói esetek

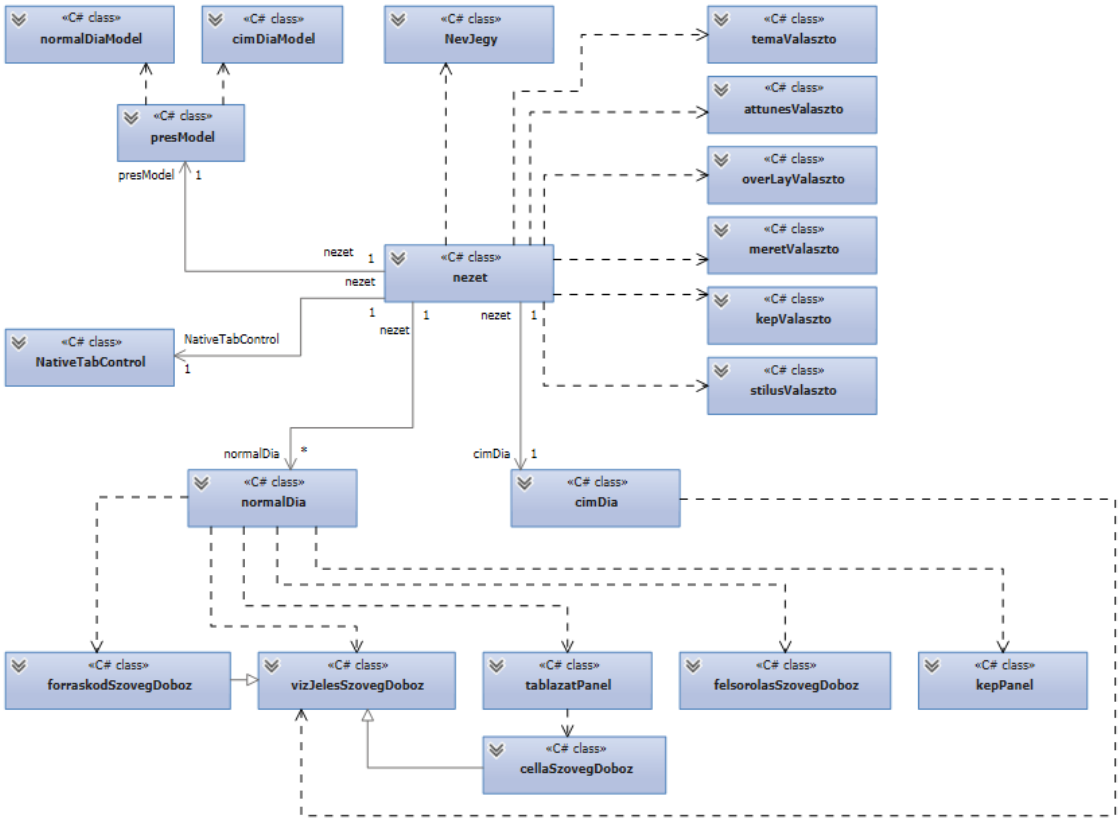
Az alábbi diagramon csak a szerkesztés általános lépései láthatóak, a részletek mennyisége miatt azoktól eltekintünk.



3.1. ábra. Felhasználói esetek diagramja

3.2.2. Az osztályok kapcsolata

Az alábbi diagramon csak a leglényegesebb osztályokat és a köztük lévő kapcsolatokat tüntettük fel.



3.2. ábra. Osztálydiagram

3.2.3. A főablak

A főablakot a `Windows.Forms.Form` osztályból származó `nezet` osztályban valósítottam meg, amely az `IMessageFilter` interfészt is implementálja.

A nézet egy parciális osztály, amely két fő forrásfájlból áll. Az egyik tartalmazza az általam írt kódot, amely a diák és egyéb ablakok megjelenítéséért, valamint a modellel való kommunikációért felel, a másikban pedig a Visual Studio-ban grafikusan megszerkesztett kezelőfelület kódja található. Előbbit a `Nezet.cs`, utóbbit a `Nezet.Designer.cs` fájl tartalmazza.

Az alkalmazás teljes képernyős módban nyílik meg. Később lehetőségünk van a

tálcára letenni, de átméretezni nem. Az osztálydiagramnak ezen ablakhoz tartozó elemét annak terjedelmessége miatt nem mellékelem.

Adattagok

A főablak tartalmaz két publikus `List`-et. Ezeket a projekten belül mindenhonnan szeretném elérni, ezért statikusként definiálom mindkettőt. Az első lista tartalmazza a `szinOsztaly`-okat, a második pedig az áttűnések magyar és angol neveit.

Privát adattagok a megjelenést meghatározó UI elemek, a rajtuk megjelenő gombok, a prezentáció megjelenítését végző `cimDia` és a `normalDia`-k listája, a modell, valamint a használat közben megjelenő `ContextMenu`-k.

Függvények

A `Konstruktor` végzi a statikus listák inicializálását, kezdeti értéket ad az adattagoknak, összeköti a modell és a különböző nézetbeli osztályok által küldött jelzéseket a főablak megfelelő metódusaival, beállítja a felugró menüket, összeköti az egér és a görgő eseményeit a hozzájuk tartozó függvényekkel, és kiszámolja az aktuális dia pozícióját és méreteit a képernyő nagyságához viszonyítva.

Új projekt nyitását az `uresProjekt()`, a mentést és a betöltést a `projektMentes()` és a `projektMegnyitas()` metódusok végzik. A fájlba írás és a fájlból való olvasás a modellben hajtódik végre.

A modell és a nézet adattagjai közötti szinkronizációt a `modelFrissites()` és a `nezetFrissites()` függvények végzik. Azért, hogy az aktuális diának is minden módosítása rögzítésre kerüljön, a fájlműveletek előtt az `aktDiaMentes()` nevű eljárást kell meghívni.

A dialistán való kattintáskor vagy a görgő használatakor a megfelelő eseménykezelő lefutása után a `diaLeptetes()` és a `diaValtas()` függvények biztosítják, hogy a megfelelő dia jelenjen meg nagyobb méretben. Ha a címdiára váltunk vagy arról el, szükséges a vezérlőfelvételt letiltó és engedélyező `vezerloFelvetelTiltas()` és `vezerloFelvetelEngedelyezes()` metódusok használata.

Ahhoz, hogy a kattintásos diaváltás lehetséges legyen, az `IMessageFilter` interfész `PreFilterMessage()` függvényében kell szűrnünk a szimpla vagy dupla gombnyomásokat aszerint, hogy a pozíciójukat tartalmazza-e valamelyik a dialistán látható `Panel`.

A felugró menük tartalmának beállítását és a megfelelő eseménykezelők hozzájuk rendelését a `<menuNev>MenuBeallitas()` függvények végzik.

Új diák beszúrásáért és már létezők törléséért az `ujDiaButton_Click()` és a `diaTorlesButton_Click()` függvények a felelősek. Áthelyezésüket a dialistán belül a `diaMozgatas(Fel/Le)Button_Click()` metódusok hajtják végre.

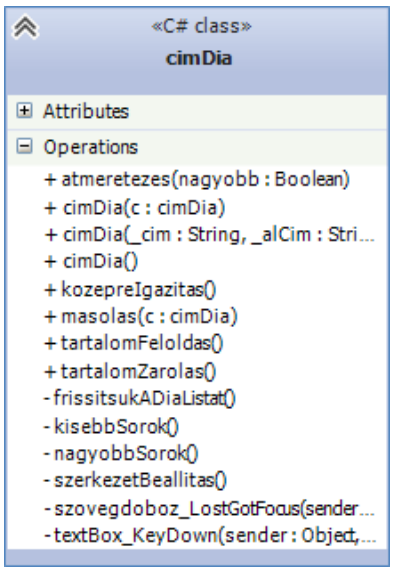
A vezérlőfelvételt intéző gombok kattintásakor az eseménykezelőkben az aktuális dia megfelelő eljárásai hívódnak meg. Táblázatok és ábrák esetén ezt megelőzi a megfelelő dialógusablak példányosítása és mutatása.

A táblázatot módosító vezérlők kattintásakor lefutó függvényekben először mindig a `user32.dll` könyvtárban található `GetFocus()` segítségével meghatározzuk, hogy az adott dián éppen melyik táblázat van kijelölve, és csak ezután hajtjuk végre a kellő módosításokat. Mivel a `Windows.Forms` könyvtárban található `Button`-ök kattintáskor fókuszt kapnak, a `GetFocus()` egy a gombra mutató pointert adna vissza, így létre kell hoznunk egy saját gomb osztályt, amelyben ezt a tulajdonságot kikapcsoljuk (lásd 3.2.7, `NonFocusButton`), és ilyen típust kell használnunk a fentebb említett vezérlők létrehozásakor.

3.2.4. A dia osztályok

A diák megjelenítésére két osztályt hozunk létre és mindkettőt a WinForms `Panel`-jéből származtatjuk.

A címdia



A prezentáció legelső slide-jának megjelenítésére szolgál a `cimDia` osztály.

Privát adattagjai a cím, az alcím, a szerző, az intézmény és a dátum megjelenítésére szolgáló `vizJelesSzovegDoboz`-ok, valamint a kinézet kialakítását végző `TableLayoutPanel`.

Megadunk getter és setter műveleteket, melyekkel az egyes vezérlők tartalmát tudjuk lekérdezni és beállítani. Háromféle `konstruktort` használunk: az első nem kap paramétereket, üres szövegekkel tölti fel a tartalmat, a második paraméterül egy `cimDia`-t kap,

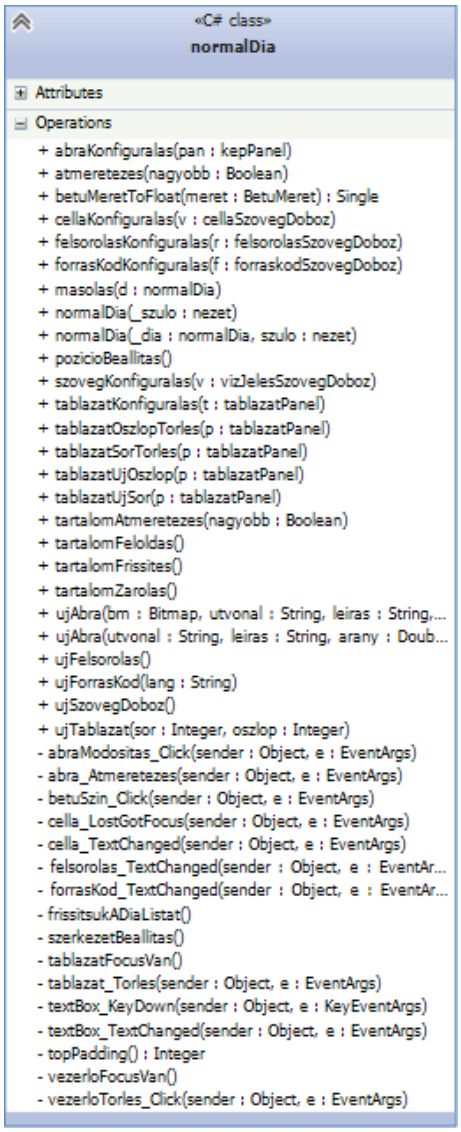
és ezt lemásolva hozza létre az új objektumot, a harmadik pedig megkapja az öt szövegdoboz tartalmát, és létrehozás után ezekkel tölti fel az új `cimDia`-t.

A dia szerkezetének beállítása és a vezérlők konfigurálása is a konstruktorokban történik. Megvalósítunk egy átméretező függvényt (`atmeretezes()`), melynek segítségével a dia méretének változásakor is arányosan fog megjelenni minden szöveg.

A szinkronizációt a dialistával a `nezet`-nek küldött jelzéssel végezzük akkor, amikor az egyik szövegdoboz elveszti a fókuszt. Szükséges ezen kívül szerkesztésnél a *Ctrl+V*-vel beillesztett szövegek stílusának eltüntetése, hogy minden csak úgy jelenhessen meg a szerkesztőben, ahogy majd ténylegesen ki fog nézni.

A normál dia

A prezentáció általános slide-jainak megjelenítésére szolgál a `normalDia` osztály.

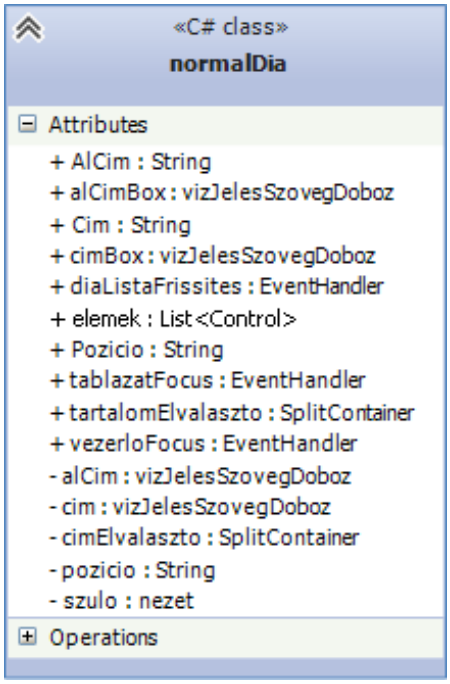


Privát adattagjai a cím és az alcím megjelenítésére szolgáló `vizJelesSzovegDoboz`-ok, a tartalmi elemeket tároló lista, a függőleges pozíció, valamint a kinézet kialakításáért felelős `SplitContainer`-ek.

A privát adattagokhoz létrehozunk getter és setter műveleteket. Két `konstruktort` használunk, ebből az egyik nem kap paramétert és csak a szerkezetet építi fel, a másik pedig egy ugyanilyen típusú objektumot kap, és az új létrehozása után lemásolja a paraméter tartalmát. A diák váltásánál szükséges még egy `masolas(normalDia)` művelet, amely nem hoz új példányt létre, csak lemásolja a paraméterül kapott dia tartalmát.

Az `atmeretezes()` függvény felel azért, hogy a dia méreteinek megváltozása esetén a vezérlők is arányosan megnőjenek vagy összemenjenek. A `tartalomFrissites()` és a `pozicioBeallitas()` metódusok gondoskodnak

arról, hogy a privát listában szereplő összes `Control` megjelenjen, és jó helyen legyenek.



A `normalDia` osztályban is megtalálható a stílus-eltávolító függvény, azonban ebben az esetben több mindent kell kezelni szerkesztéskor. Szövegek, felsorolások és forráskódok esetében minden karakter bevitelekor ellenőrizni kell, hogy nem kell-e új sort kezdeni, mert ilyenkor meg kell növelni a vezérlő méretét. Ezt a feladatot a `_TextChanged()`-re végződő függvények látják el. Táblázat cellák esetén is figyelni kell a tartalom módosítást, és a szöveg hosszához kell igazítani az adott oszlop szélességét (`cella_TextChanged()`).

A dia vezérlőire való jobb kattintással `ContextMenuStrip` jelenik meg, mely a vezérlő típusától függően különböző elemeket tartalmazhat. Mindegyiknél lehetőségünk van törlésre (`vezerloTorles_Click()`, szöveg, felsorolás és táblázat esetén stílus módosításra (`betuSzin_Click()`), ábráknál pedig az ábra tulajdonságainak megváltoztatására (`abraModositas_Click()`). Utóbbi kettő esetén egy felugró ablakban lehet elvégezni a kívánt módosításokat.

Vezérlők felvételére szolgálnak az `ujTablazat()`, `ujAbra()`, stb. függvények. Ezekben először meghívódik a vezérlő konstruktora, majd végrehajtódik a hozzá tartozó konfiguráló eljárás (`tablazatKonfiguralas()`, `abraKonfiguralas()`, ...).

A `tartalomZarolas()` és `tartalomFeloldas()` függvények biztosítják, hogy a dialistán megjelenő slide-ok ne legyenek szerkeszthetők.

A táblázatok szerkesztését a `tablazat`-tal kezdődő eljárások végzik. Ezek paraméterül kapják a módosítandó táblázatot, és annak függvényeinek hívásával, majd átméretezéssel elvégzik a kívánt sor/oszlop törlést vagy beszúrást.

3.2.5. A vezérlő osztályok

A címek és a tartalmi elemek megjelenítéséhez egyedi vezérlőket hozunk létre a `Windows.Forms` könyvtár osztályainak segítségével.

A vízjeles szövegdoboz

A `vizJelesSzovegDoboz` osztályt a címek és sima szövegek megjelenítésére hoztam létre. A vízjel célja, hogy akkor is lássuk a szöveg helyét, ha tartalma éppen üres lenne. A beépített `RichTextBox` osztályból lett származtatva, így támogatja különböző betűszínek és stílusok megjelenítését is.

Privát adattagjai a vízjel szövege, egy logikai változó, amely azt határozza meg, hogy aktív-e épp a vízjel, valamint a betűszín, a stílus, a méret, az overlay érték, és az igazítás.

Előbbiek lekérdezésére és módosítására getter és setter függvények állnak rendelkezésére. A `konstruktor`ban kezdőértéket kapnak az adattagok, és lambda-kifejezések segítségével hozzákötöm a `GotFocus` és `LostFocus` eseményekhez a megfelelő eljárásokat. A `masolas()` függvény lehetővé teszi egy másik `vizJelesSzovegDoboz` tartalmának és stílusának lemásolását.

A vízjel ki és bekapcsolására szolgál a `VizjelKi()` és a `VizjelBe()` függvény, a `stilusFrissites()`-el pedig a privát adattagokban megadottak alapján módosul a megjelenés.

A felsorolás szövegdoboz

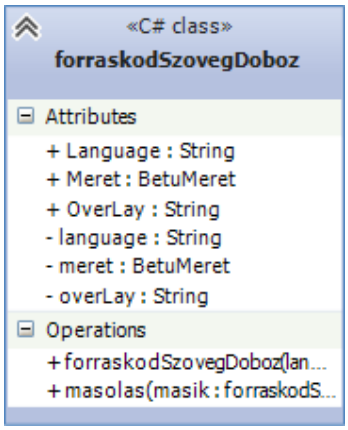
A `felsorolasSzovegDoboz` osztály minden adattagja és metódusa megtalálható a `vizJelesSzovegDoboz`ban is, ezért ezeket itt nem részletezem külön.

A különbség az előbb említett osztályhoz képest az, hogy nem rendelkezik vízjellel, mivel konfigurálásakor `IGAZ`-ra állítjuk a `RichTextBox`-tól örökölt `SelectionBullet` tulajdonságát, ezáltal minden sor elején felsorolást jelző fekete pont jelenik meg, így nem szükséges vízjellel mutatni, hogy hova írhatunk.

Megjegyzés: Több Enter ütése esetén eltűnhet a felsorolást jelző pont a sorok elejéről, de ettől függetlenül a PDF-ben minden sor a felsorolás egy elemeként lesz látható.

A forráskód szövegdoboz

A `forraskodSzovegDobozt` a `vizJelesSzovegDoboz`-ból származtatom, mivel alapesetben ez se tartalmazna semmit, és a vízjel nélkül nem látná a felhasználó, hogy hova írhat.



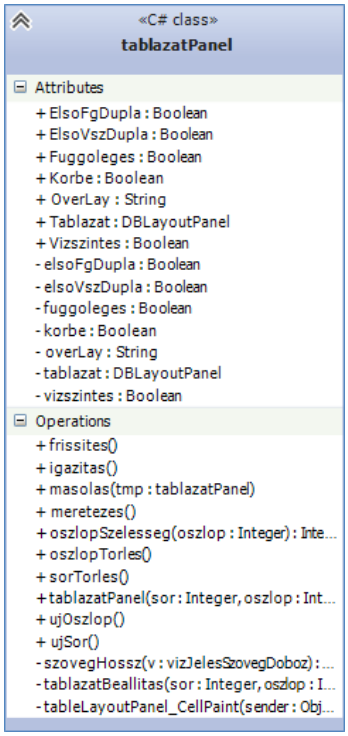
Kiegészítettem ezen kívül egy privát adattaggal és a hozzá tartozó getterrel és setterrel, amely a forráskód programnyelvét tartalmazza. A `konstruktornak` ezt paraméterül kell adni, és a `masolas()` függvényben is át kell venni a másik példány ezen értékét.

A cella szövegdoboz

A `cellaSzovegDoboz` osztályt is a `vizJelesSzovegDoboz`-ból származtatom. Semmi egyedi tulajdonsággal vagy függvénnyel nem rendelkezik, csak azért van rá szükség, hogy a táblázat celláit meg lehessen különböztetni a sima szövegektől.

A táblázat panel

A `tablazatPanelt` a WinForms beépített `Panel` osztályából származtatom. Ehhez adom hozzá a tényleges táblázatot, aminek szintén külön osztályt hozok létre a `Windows.Forms.TableLayoutPanel`-ből (bővebben lásd 3.2.7).



A privát adattagok között van a fentebb említett táblázat, a szegélyeket leíró öt logikai változó és az overlay érték is.

Akárcsak az eddigi vezérlőknél, ezekhez is tartozik getter, setter, hogy kívülről is módosíthatók legyenek. A `konstruktor` a sor- és oszlopszámot kapja paraméterül, ezek alapján hozza létre a táblázatot, de a `cellaSzovegDoboz`ok hozzáadását nem ő végzi. A keret megjelenítéséhez saját rajzoló függvényt írunk a cellának, amely aszerint tesz köréjük szegélyt, hogy melyik keretváltozó `IGAZ` (ez a `tableLayoutPanel_CellPaint()` metódus).

A `meretezes()` függvényben minden oszlopnál meghatározom azt, hogy mennyi a szélessége a leghosszabb szöveget tartalmazó cellának (az `oszlopSzelesseg()` metódussal), és az egész oszlopot erre a méretre állítom. Megvalósítottam ezen kívül egy `igazitas()` nevű eljárást, amely minden cellát vízszintesen balra, jobbra vagy középre igazít, a `vizJelesSzovegDoboz`tól örökölt igazítás változója alapján.

A kép panel

A `tablazatPanel`hez hasonlóan ezt az osztályt is a már előre adott `Panel`-ből származtatom.

Privát adattagjai az ábra tulajdonságait reprezentálják, úgymint fájlnév, elérési útvonal, leírás, méretarány, overlay.

A kép megjelenítésére egy saját, `PictureBox`-ból származtatott osztályt hoztam létre (bővebben lásd 3.2.7). Az ábrát egy `Windows.Forms.Bitmap`-be töltöm be, és ezt adom oda a `SelectablePictureBox`-nak.

Két `konstruktort` készítettem. Az egyik a kapott útvonallal nyitja meg a képet, a másik pedig csak lemásolja a paraméterül átadott `Bitmap`-et, ezzel megspórolva a fájlnyitás költségét. Az első esetben elképzelhető, hogy a megnyitás alatt eltávolítják a fájlt a megadott helyről, ekkor a konstruktor az erőforrásokból tölt be egy NoImage.png nevű képet, és létre is hozza ennek egy megfelelőjét az exe mellé, hogy így is fordítható legyen a prezentáció.

Az osztálynak van egy `kepModositas()` függvénye is, amelyet akkor használok, ha az ábrán utólag változtat valamit a felhasználó (útvonal, leírás, méret).

3.2.6. A dialógus ablakok

A hat felugró ablakot a `Form` osztályból származtatom, akárcsak a főablakot. A vezérlők beállításának nagy része a Visual Studio szerkesztőjében történt, de néhány elem a konstruktorban véglegesítődik. A kívánt értékek lekéréséhez getter műveleteket hozok létre.

A `temaValaszto` ablakban van lehetőség a prezentáció témájának kiválasztására. A mintaképeket erőforrásokból adom hozzá a táblázat celláihoz, az ablak felépítése-

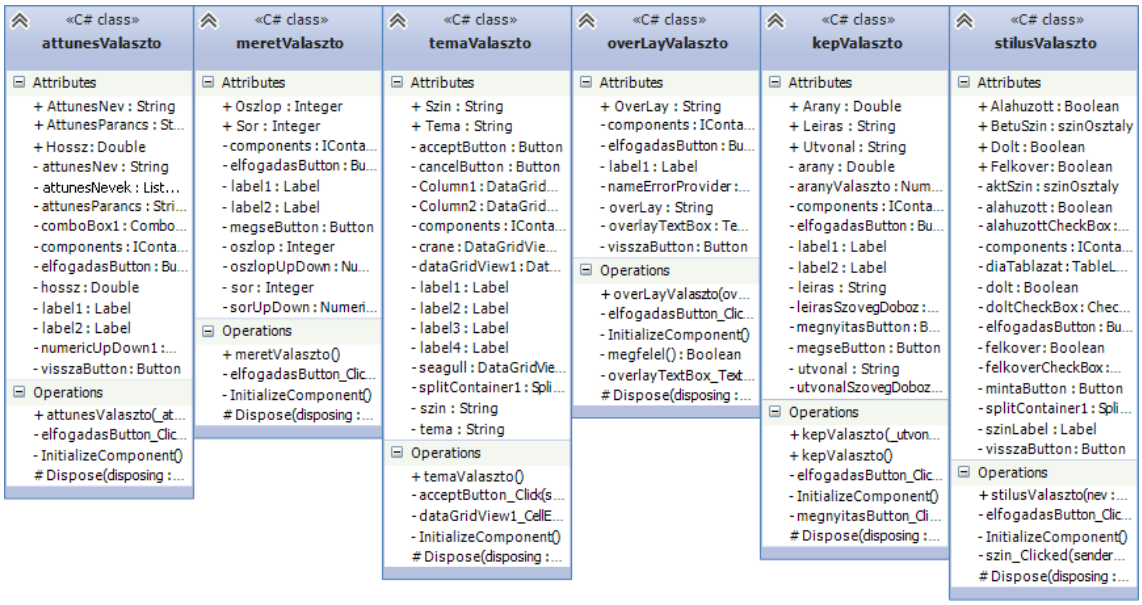
kor.

Az `attunesValaszto` osztály segítségével lehet megadni a diák közötti áttűnés-animációt. A lehetőségeket egy legördülő menüben jelenítem meg.

A `meretValaszto` dialógus ablakban adhatjuk meg a beszúrni a kívánt táblázat sor- és oszlopszámát két `Windows.Forms.NumericUpDown` vezérlő segítségével.

A `kepValaszto` ablakban lehet megadni a beszúrni kívánt kép elérési útvonalát és leírását. Az `elfogadasButton_Click()` függvényében kerül sor az ellenőrzésekre, amik eldöntik, hogy lett-e útvonal megadva (anélkül nem végezhető el a betöltés), megfelelő-e a kép mérete, illetve létezik-e még egyáltalán a fájl. Amennyiben valamelyik nem teljesül, hibaüzenet jelenik meg.

A `stilusValaszto` osztály segítségével lehet a szövegek betűszínét és stílusát megadni. Ehhez szükségünk van egy színpárokat tartalmazó listára, amely a LaTeX-ben használható színeknek megfeleltet egy megjeleníthető RGB értéket. Ezt már a `nezet`-ben létrehoztuk, így erre hivatkozva felveszünk 48 gombot, mindegyikhez hozzárendelünk egy színt, és a gombok kattintásakor módosítjuk az osztály aktuális színt reprezentáló adattagját.



3.3. ábra. A dialógus ablakok osztálydiagramja

3.2.7. A segédosztályok

A kód áttekinthetőségéért és a `Windows.Forms` könyvtár hiányosságainak kiküszöböléséért több kisebb kisegítőosztályt is létrehoztam.

DBLayoutPanel

A programban több helyen volt szükséges `TableLayoutPanel`-ek használata. Ezen osztály nyújtotta funkciók elegek voltak a program elkészítéséhez, azonban a kirajzolása túl lassú volt, így gyors módosításoknál vagy mozgatásnál a tartalma elmosódott.

Mivel a `TableLayoutPanel` nem rendelkezik `DoubleBuffered` tulajdonsággal, saját osztályt kellett származtatnom belőle, és annak konstruktorában engedélyezni a dupla pufferelést.

NativeTabControl

Ezt az osztályt is a WinForms hibáinak kiküszöbölésére hoztam létre, ugyanis a beépített `TabControl` körül meg nem szüntethető keret jelenik meg egy adott színben. A `NativeTabControl` segítségével azonban még kirajzolás előtt tetszőlegesen állíthatók a `TabControl` méretei, így tüntetve el a nem kívánt keretet.

NonFocusButton

Működése teljesen megegyezik a `Windows.Forms.Button`-éval, azzal az egy különbséggel, hogy a konstruktorában a `Selectable` tulajdonságot `HAMIS`-ra állítom. Így amikor rákattint a felhasználó, a fókusz az előző vezérlőn marad, lehetővé téve annak tulajdonságainak lekérdezését a `GetFocus()` metódus segítségével.

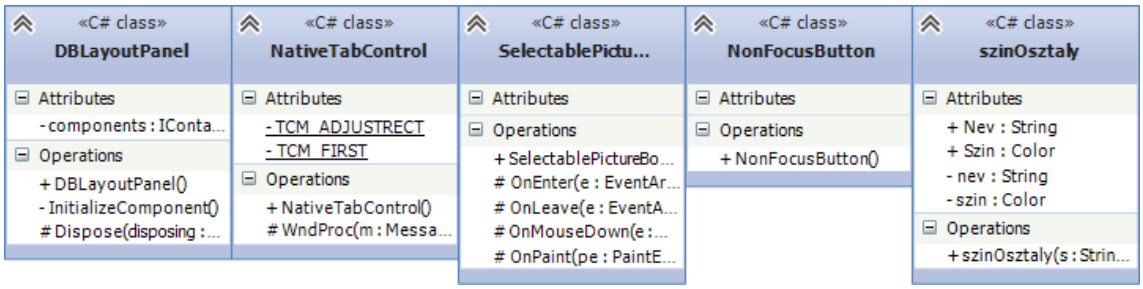
SelectablePictureBox

Az előbbi ellenkezőjét hajtom végre a `SelectablePictureBox` esetében, mivel a `Windows.Forms.PictureBox` alapból nem kijelölhető. A `Selectable` tulajdonság `IGAZ`-ra állításán kívül felüldefiniálom még a `Paint` eseményt is, hogy amikor fókuszt kap a kép, szaggatott vonalú keret jelenjen meg körülötte.

szinOsztaly

Ahhoz, hogy a szerkesztőben is meg lehessen jeleníteni a LaTeX-ben használt színeket, össze kellett párosítanom az RGB értékeket a neveikkel.

Erre szolgál a `szinOsztaly`, amely két privát adattaggal rendelkezik. Az egyik egy `System.Drawing.Color`, ez felel a WinForms-os megjelenítésért, a másik pedig egy szöveg, ami annak LaTeX-beli elnevezését tárolja. A konstruktornak mindkét értéket paraméterül kell adnunk.

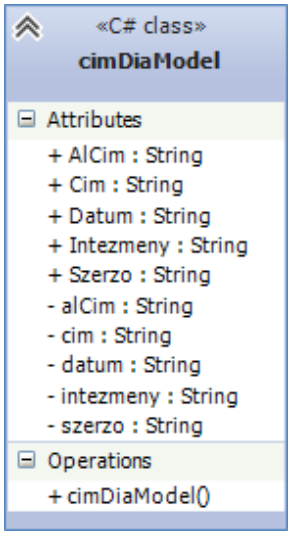


3.4. ábra. A segédosztályok diagramja

3.2.8. A modell

A prezentáció modellje végzi a mentést/betöltést, és a TeX fájl generálást, valamint a fordító indítását. Ennek megkönnyítésére minden információt a projektről szöveges formában tárolunk. Külön modell osztályokat hozunk létre a különböző diáknak, így téve egyszerűbbé azok feldolgozását.

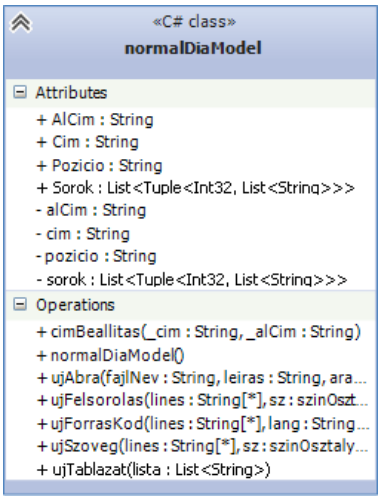
A címdia modell



A `cimDiaModel` öt darab privát adattaggal rendelkezik, melyek mindegyike egy, a címdián megjelenő feliratot reprezentál. Ezek a cím, az alcím, a szerző, az intézmény és a dátum.

A `konstruktor` nem ad semelyiknek se kezdeti értéket, az adattagokat egyesével állítom be a megfelelő setter metódusok segítségével. A lekérdezéshez minden adattaghoz létrehoztam egy getter függvényt is.

A normál dia modell



A `normalDiaModel` osztály privát adattagjai tartalmazzák a diák felépítéséhez szükséges információkat. Felvettem egy változót a címnek, az alcímnek és a pozíciónak, valamint külön listában tároltam el az egyes vezérlők információit.

A `sorok` lista párokat tartalmaz, melyek első tagja egy egész szám. Ennek értéke 0 és 4 között változhat attól függően, hogy milyen típusú vezérlő adatait tartalmazza a pár második eleme. 0 jelöli a szövegeket, 1 a felsorolásokat, 2 a forráskódokat, 3 a táblázatokat és 4 az ábrákat. A már említett második tag is `List` típusú, amely szövegeket tartalmaz. Ennek elemei vezérlőtől függően változnak, de legtöbbször a betűszínt, a stílusjegyeket és a szövegek sorait tartalmazzák.

A `konstruktor` csak a lista létrehozását végzi, egyéb értékeket nem állít be. A `cimBeallitas()` függvénnyel lehet értéket adni a címnek és az alcímnek. A lista feltöltéséhez minden vezérlőnek külön metódust hoztam létre.

`ujSzoveg()`: Paraméterül kapja a szöveg színét, a három stíluselemet meghatározó logikai változót (félkövér, dőlt, aláhúzott), az overlay értéket, a betűméretet végül pedig a szöveget soronként tartalmazó listát. Az előbbiek mindegyikét `String`-gé alakítom, listába rendezem, és hozzáadom a `normalDiaModel` megfelelő adattagjához.

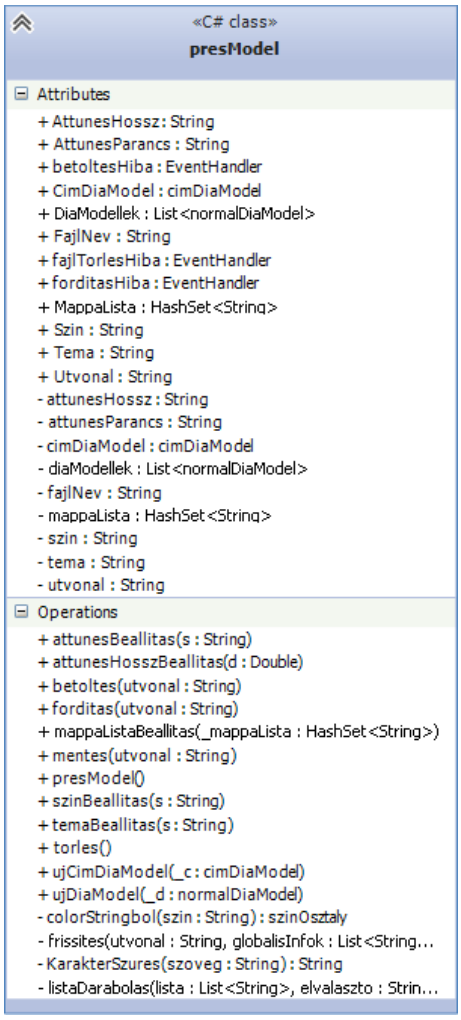
`ujFelsorolas()`: A függvény működése megegyezik az előzőével, azzal az egy kivétellel, hogy mikor a létrehozott listát hozzáadjuk a `normalDiaModel`-hez, a pár első tagja 1 lesz 0 helyett.

`ujForrasKod()`: Forráskódoknál nem adunk át stílus-paramétereket, helyette megkapja a programozási nyelv nevét a metódus az overlay értéken, a méreten, és a tartalmon kívül.

`ujTablazat()`: Táblázat esetén a lista felépítése nem a modellben történik, mivel nem tudjuk előre a méreteket. Helyette a függvény már egy kész listát kap, ami tartalmazza rendre az oszlop- és sorszámot, oszloponként az igazítás értékét, a kereteket meghatározó logikai értékek szöveges formáját, valamint sorfolytonosan a cellák stílusát és tartalmát.

`ujAbra()`: A fájl nevét, leírását, a dia méretéhez viszonyított nagyságát, a teljes elérési útvonalát és az overlay értékét kapja meg. A saját listájának felépítése után hozzáadja azt a modellhez.

A prezentáció modell



Privát adattagjai között a diamodellek mellett a teljes projektre vonatkozó információk találhatóak meg. Ezek a fájlnév, az útvonal, a téma, a színséma, áttűnés neve és hossza, valamint a mappalista, amelyben a beszúrt képek mappáinak útvonalát tárolom. Előbbiek megadásához külön függvényeket hozunk létre, melyek neve `Beallitas()`-ra végződik.

Ha hiba történne a modellben, jelzést küldünk a nézetnek. Minden hibához külön eseményt hoztam létre, ezek pedig a következők: `forditasHiba`, `fajlTorlesHiba` és `betoltesHiba`.

`forditasHiba` a .tex fájl fordítása során következhet be. `fajlTorlesHiba` PDF generáláskor történhet, ha nem sikerül a korábban már lefordított projekt segédfájljait törölni (például épp meg van nyitva valamelyik). Végül `betoltesHiba` sérült mentésfájl megnyitásakor keletkezhet.

A `konstruktor` csak a listák létrehozását végzi, valamint megvalósítottam egy `torles()` függvényt, amellyel a teljes modell alaphelyzetbe állítható.

A `mentes()` függvény egyszerű, mivel már minden szöveges formátumban van eltárolva. Ezeket az információkat egy .prez kiterjesztéssel ellátott szöveges fájlba írom soronként egy `System.IO.StreamWriter` segítségével. Ahhoz, hogy betöltéskor elkülöníthetőek legyenek a diák és a vezérlők, elválasztó sorokat iktatok be közéjük.

A `betoltes()` függvényben először a mentésfájl minden sorát beolvasom egy lis-

tába. Írtam egy függvényt (`listaDarabolas()`), amely paraméterül kap egy `String` listát és egy elválasztó karakterláncot, majd visszaad egy `List<List<String>>`-et, amely az eredeti lista részeit tartalmazza, a megadott szöveg mentén darabolva. Ezt a függvényt alkalmazom először a diákra, majd a vezérlőkre. Az így kapott listák alapján a `frissites()` függvény végzi el a modell felépítését, a megfelelő információk kiolvasásával.

A .tex és .pdf fájlok generálását a `forditas()` függvény végzi. Először is létrehoz egy ideiglenes mappát, amelybe a LaTeX fordító által létrehozott egyéb fájlok kerülnek. Ezután elkészíti a többnyire fix preambulumot, amely a felhasznált csomagokat (például az ábrák és a forráskódok megjelenítéséhez szükséges `graphicx`-et és `listings`-et vagy az ékezetes karakterekhez használt `inputenc`-et és `fontenc`-et), és definíciókat (különböző színek, forráskódok beállításai) tartalmazza, valamint azt, hogy melyik mappákban keresse a fordító a megadott képeket. Ez utóbbi által nem szükséges, hogy minden felhasznált kép egy helyen legyen. A címdia létrehozása után végigiterál a diamodellek listáján, és hozzáadja mindhez a saját vezérlőit a modell alapján. Amikor a .tex fájl elkészült, `pdflatex` segítségével lefordítja azt. Ha a konzol ablak eltűnik, a fordítás hibamentesen lezajlott. Ellenkező esetben szöveges üzenetet láthatunk a problémáról.

A felhasználó a szerkesztőben használhat olyan karaktereket is, amelyek speciális jelentéssel bírnak a fordító számára, ezért minden ilyet lecserélek a `KarakterSzures()` függvény segítségével az ugyanazt a szimbólumot eredményező LaTeX-beli parancsra.

Megjegyzés: Forráskódoknál nem volt elég a LaTeX által nyújtott `\lstlisting`-et alkalmazni, hanem be kellett ágyazni azt egy `block`-ba, mivel az előbbi nem támogatja az overlay-ért felelős `\only` parancsot.

3.3. Tesztelés

Az alkalmazás tesztelése a fejlesztés során folyamatosan történt, az egyes részek gyakorlati alkalmazásával és debug üzenetek segítségével. Külön az osztályokhoz és a függvényekhez nem hoztam létre teszteket.

3.3.1. Nézet

A nézet tesztelésekor ellenőriztem minden menüpont helyes működését és a dinamikusan létrehozott felugró menük tartalmát és jó helyen való megjelenését. Figyelnem kellett, hogy a dialista elemeit módosító függvények után is a megfelelő sorrendben jelennek-e meg a diák. Teszteltem, hogy bizonyos vezérlők vagy diák kiválasztásakor csak azok a gombok érhetők-e el, amelyeknek szabad, és a megfelelő felirat van-e rajtuk.

Nagy szerepet kapott az ablak és a diák helyes méretezésének elemzése. Úgy valósítottam meg a diákat, hogy átméretezésük esetén a vezérlők is arányosan összemenjenek, vagy megnőjenek. Amennyiben az átméretezés pillanatszerű (lásd diák váltása), ez helyesen és gyorsan működik, azonban az ablak méretének folyamatos változtatásakor problémákba ütköztem. Az aktuális dia mérete ilyenkor egyfolytában változik, és amennyiben táblázatot is tartalmaz, annak mindig újra kell rajzolnia önmagát. Windows Forms esetén ez sajnos rettentően lassan történik, ami nagyon sokat elvett a látványból. Hasonló hibába ütköztem, amikor a felület megkapta a végleges háttérképét. Ahhoz, hogy ez jól látszódjon, több `Panel` színét is áttetszőre kellett állítanom. Méretezéskor azonban ez jelentős lassulást eredményezett a kirajzolásban, így a megjelenés szaggatottá vált, sok helyen fekete foltokkal. Ezek hatására úgy döntöttem, hogy a tervvel ellentétben az ablak nem lesz folyamatosan átméretezhető, hanem csak felveszi kezdetben a kijelző méretét.

3.3.2. Dialógus ablakok

A dialógus ablakok tesztelésekor ellenőriztem a vezérlők megjelenését, a dinamikusan felvett elemek helyességét és a beállított korlátok működését.

A stílusválasztó esetén teszteltem, hogy minden szín megfelelően jelenik-e meg mind a szerkesztőben, mind a generált PDF-ben.

A témaválasztónál ellenőriztem, hogy minden témához és színsémához a megfelelő kép tartozik-e, és az értékek helyesen adódnak-e vissza a nézetnek.

Megnéztem, hogy a kép beszúrásával szemben támasztott feltételek helyesen működnek-e.

Teszteltem, hogy overlay megadásakor csak a reguláris kifejezésnek megfelelő értékeket fogadja-e el a program.

3.3.3. Modell

A modellt folyamatosan teszteltem. Minden vezérlő és egyéb funkció implementálása után ellenőriztem, hogy a benne található értékek helyesek maradnak-e, és a generált fájlok is úgy jelennek-e meg, ahogy kell.

A mentés és betöltés megvalósítása után több különböző prezentációval is ellenőriztem azokat. Létrehoztam ezen kívül szövegszerkesztőben is mentésfájlokat. Arra voltam ezzel kíváncsi, hogy több száz diát is jól kezel-e a program. Azt tapasztaltam, hogy ez sem eredményez lassulást vagy hibákat, még akkor se, ha minden dia több szöveget és képet is tartalmaz.

4. fejezet

Összegzés

Összességében elmondható, hogy kisebb eltérésekkel ugyan, de a kezdeti tervnek megfelelő programot sikerült létrehozni. Alkalmas rövidebb vagy akár hosszabb prezentációk elkészítésére, és az egyszerű kezelőfelületnek hála senkinek sem okozhat akadályt a használatának elsajátítása.

Kevés kiegészítéssel további LaTeX elemek és témák támogatása is megoldható, valamint az eddigiek is továbbfejleszthetők. A Modell másik megjelenítési réteg mellett is felhasználható lehet a jövőben.

Átlátható .tex forrásfájlt készít, így a nyelvben jártas felhasználók számára remek kiindulópontot biztosít egyéb módosítások elvégzéséhez. Azoknak pedig, akik csak most ismerkednek a témával, megkönnyítheti az egyes parancsok és vezérlők megértését.

Számomra is hasznos volt a dolgozat elkészítése. Elsajátítottam egy eddig ismeretlen szövegszerkesztési módszert, valamint komoly tapasztalatokra tettem szert a grafikus felületű alkalmazások fejlesztése, a **C#** nyelv használata, és különböző debug-olási módszerek terén.

A LaTeX rendszer megismerésében nagy segítséget nyújtott a Wikibooks ezen témáról szóló könyve [1], és a tex.stackexchange fórumozói [2]. A Windows Forms lehetőségeiről a hivatalos dokumentációban [3] tájékozódtam.

Irodalomjegyzék

[1] LaTeX - Wikibooks, http://en.wikibooks.org/wiki/LaTeX, 2015-05-03

[2] TeX - LaTeX Stack Exchange, http://tex.stackexchange.com/, 2015-04-15

[3] Microsoft Developer Network, https://msdn.microsoft.com/, 2015-04-11

[4] Beamer Theme Matrix, https://www.hartwork.org/beamer-theme-matrix/, 2015-05-04