Szakdolgozat

Miskolci Egyetem

Online grafikus szerkesztő alkalmazás TikZ ábrák készítéséhez

Készítette:

Veréb Tamás

Programtervező informatikus

Témavezető:

Vadon Viktória

Miskolc, 2021

Miskolci Egyetem
Gépészmérnöki és Informatikai Kar
Alkalmazott Matematikai Intézeti Tanszék **Szám:**

Szakdolgozat Feladat

Veréb Tamás (HQYII1) programtervező informatikus jelölt részére.

A szakdolgozat tárgyköre: LaTeX, TikZ

A szakdolgozat címe: Online grafikus szerkesztő alkalmazás TikZ ábrák készítéséhez

A feladat részletezése:

A TikZ egy LaTeX-hez készített csomag, amely segítségével vektorgrafikus ábrákat lehet készíteni. Az előre definiált rajzelemek paramétereinek a megadásával procedurális módon lehet vele rajzolni többek között például gráfokat, grafikonokat, geometriához kapcsolódó illusztrációkat.

A csomag a rajzolási műveletek megadását szöveges formában teszi lehetővé. A dolgozat célja, hogy bemutassa egy online szerkesztőfelületnek a tervezését és elkészítését, amely grafikus szerkesztést tesz lehetővé, az eredményt pedig LaTeX-be visszailleszthető forrásszöveg formájában adja.

A dolgozat felsorolja a hasonló célú alkalmazásokat, összehasonlítja azok funkcióit. Specifikálja a szerkesztőfelületet. Vizsgálja annak lehetőségét, hogy a korábban már szerkesztett ábrák milyen esetekben módosíthatók az alkalmazással a LaTeX-es forráskód ismeretében.

Témavezető: Vadon Viktória (egyetemi tanársegéd)

A feladat kiadásának ideje: 2020. 09. 27.

. . . . . .
szakfelelős

Eredetiségi Nyilatkozat

Alulírott **Veréb Tamás**; Neptun-kód: `HQYII1` a Miskolci Egyetem Gépészmérnöki és Informatikai Karának végzős Programtervező informatikus szakos hallgatója ezennel büntetőjogi és fegyelmi felelősségem tudatában nyilatkozom és aláírásommal igazolom, hogy *Online grafikus szerkesztő alkalmazás TikZ ábrák készítéséhez* című szakdolgozatom saját, önálló munkám; az abban hivatkozott szakirodalom felhasználása a forráskezelés szabályai szerint történt.

Tudomásul veszem, hogy szakdolgozat esetén plágiumnak számít:

- szószerinti idézet közlése idézőjel és hivatkozás megjelölése nélkül;
- tartalmi idézet hivatkozás megjelölése nélkül;
- más publikált gondolatainak saját gondolatként való feltüntetése.

Alulírott kijelentem, hogy a plágium fogalmát megismertem, és tudomásul veszem, hogy plágium esetén szakdolgozatom visszautasításra kerül.

Miskolc, ...év ..... .hó ..... .nap

.....

Hallgató

1.

szükséges (módosítás külön lapon)

A szakdolgozat feladat módosítása

nem szükséges

....

dátum témavezető(k)

2. A feladat kidolgozását ellenőriztem:

témavezető (dátum, aláírás): konzulens (dátum, aláírás):

....

....

....

3. A szakdolgozat beadható:

....

dátum témavezető(k)

4. A szakdolgozat ..... szövegoldalt

..... program protokollt (listát, felhasználói leírást)

..... elektronikus adathordozót (részletezve)

.....

..... egyéb mellékletet (részletezve)

.....

tartalmaz.

....

dátum témavezető(k)

5.

bocsátható

A szakdolgozat bírálatra

nem bocsátható

A bíráló neve: .....

....

dátum szakfelelős

6. A szakdolgozat osztályzata

a témavezető javaslata: .....

a bíráló javaslata: .....

a szakdolgozat végleges eredménye: .....

Miskolc, ...

a Záróvizsga Bizottság Elnöke

Tartalomjegyzék

1. fejezet

Bevezetés

A LaTeX[13][11]-et széles körben használják a tudományos életben tudományos dokumentumok közlésére és közzétételére számos területen, többek között a matematikában, az informatikában, a mérnöki tudományokban, a fizikában, a kémiában, a közgazdaságtanban, a nyelvészetben, a kvantitatív pszichológiában, a filozófiában és a politikatudományban. A LaTeX a TeX szövegszerkesztő programot használja a kimenet formázásához, és maga is a TeX makrónyelvben íródott.

A többi szövegszerkesztő programmal ellentétben, amelyek a WYSIWYG (*what-you-see-is-what-you-get*) elv szerint működnek, a szerző szövegfájlokkal dolgozik, amelyekben szövegesen, parancsokkal jelöli ki azokat a részeket vagy címsorokat, amelyeket egy szövegen belül másképp kell formázni. Mielőtt a LaTeX rendszer a szöveget megfelelően beállítaná, fel kell dolgoznia a forráskódot. A LaTeX által generált elrendezés nagyon tiszta, a képletkészlet pedig kifinomult. A LaTeX különösen alkalmas az olyan terjedelmes munkákhoz, mint a szakdolgozatok és disszertációk, amelyeknek gyakran szigorú tipográfiai követelményeknek kell megfelelniük.

A tanulmányaim során megismerkedtem a JavaScript[5] nyelvvel és egy könnyen elsajátítható programozási nyelvnek találtam. A JavaScript, gyakran JS rövidítéssel, az ECMAScript[9] specifikációnak megfelelő programozási nyelv. A JavaScript magas szintű, gyakran futásidejű fordítású nyelv. Az ECMAScript 2015[8] (*ES6*) bevezetésével egy jól használható objektumorientált nyelv lett. A JavaScript kódot közvetlenül a weboldalakba lehet beágyazni, így a webböngésző gondoskodik a szkripteknek nevezett programok végrehajtásáról. Általában a JavaScript-et a HTML-űrlapokba beírt adatok vezérlésére, vagy a HTML-dokumentummal való interakcióra használják. Dinamikus alkalmazások, átmenetek, animációk létrehozására vagy adatok manipulálására is használható.

A szakdolgozatom egy online grafikus szerkesztő elkészítése és dokumentálása. A szerkesztő a *TikZ* LaTeX csomag nyelvi elemeire épül. A webes fejlesztés miatt a HTML5[4] és a JavaScript nyelv adja az alapokat. A rajzolási felületet a *p5.js*[14] nyújtja.

A következő oldalakban megismerkedhetünk a TikZ[19] csomag telepítésével, a grafikus elemivel. A LaTeX funkcióinak kiegészítésére a felhasználó harmadik féltől származó csomagokat tölthet be. Ezek, akárcsak a függvénykönyvtárak, további parancsokat biztosítanak, az egyszerű szimbólumoktól kezdve az összetett funkciókig. Ezt követően a már meglévő szerkesztők kerülnek jellemzésre, majd összehasonlításra különböző szempontok alapján. Ezek a feltételek kerülnek megfogalmazásra a készülő alkalmazással szemben, mint követelmények.

A következő szegmens már a dokumentáció része az alkalmazásnak. Itt kerülnek kifejtésre a felhasznált függvénykönyvtárak, az elkészült alkalmazás felépítése, funkciói, használata, és a definiált osztályok leírása, a *TikZ*, mint LaTeX könyvtár és a *p5.js*, mint JavaScript függvénykönyvtár eszközkészletében található eltérések.

A végezetül már az elkészült alkalmazással megvalósított ábrák kerülnek bemutatásra.

2. fejezet

A TikZ és eszközkészlete

2.1. Ábrák szerkesztése LaTeX-ben

A rajzolás megkönnyítése érdekében a *TikZ*-t[18][19] használjuk, amely egy frontend réteg a *PGF*-hez. A *PGF* a "*Portable Graphics Format*" rövidítése. A *PGF* egy alacsonyabb szintű nyelv, míg a *TikZ* egy *PGF*-et használó magasabb szintű makrókészlet. A *PGF* és a *TikZ* parancsok TEX makróként hívhatók meg. A *TikZ* a *PGF* 0.95-ös verziójában jelent meg.

Egy ábrát létrehozni azt jelenti, hogy egyenes vagy görbe vonalak sorozatát rajzoljuk meg. A *TikZ* parancsaira és szintaxisára olyan források voltak hatással, mint a *MetaPost*[7], vagy a *PSTricks*[22].

A *MetaPost* egy programozási nyelvre és a *MetaPost* interpreterére egyaránt utal. Mindkettő Donald Knuth nevéhez köthető. A *MetaPost* geometriai vagy algebrai leírásból vektorgrafikus diagramokat készít. A nyelv osztozik a Metafont deklaratív szintaxisán a vonalak, görbék, pontok és geometriai transzformációk manipulálásához.

A *PSTricks* olyan makrók gyűjteménye, amelyek lehetővé teszik a PostScript grafikák TEXbe vagy LATEX -be való beágyazását. Eredetileg Timothy Van Zandt fejlesztette ki. Napjainkban a tudomány és a technológia számos területén nagy jelentőséggel bír a publikációkban.

2.2. A TikZ elemei

2.2.1. Használata

Először is be kell állítanunk a környezetünket. Ehhez a következőképpen néz ki a fájlunk:

```
\documentclass{article}
\usepackage{tikz}
\begin{document}
    <content>
\end{document}
```

Ezután kezdhetjük el az ábrák létrehozását.

2.2.2. Szintaxis

A pontokat és koordinátákat a *TikZ* egy speciális szintaxissal adja meg. A legegyszerűbb, ha kerek zárójelben vesszővel elválasztott két dimenziót használunk, például (4pt, 6pt). Ha az egység nincs megadva, akkor az xy-koordináta rendszerének alapértelmezett értékeit használjuk. Ez azt jelenti, hogy az egységnyi x-vektor 1 cm-rel jobbra, az egységnyi y-vektor pedig 1 cm-rel felfelé halad.

A szabály az, hogy minden *TikZ* grafikus rajzoló parancsnak a *tikz* parancs argumentumaként vagy a *tikzpicture* környezeten belül kell előfordulnia. A környezeten belül megadott összes opció a teljes képre vonatkozik.

A *tikzpicture* környezet LaTeX változata a következő:

```
\begin{tikzpicture}[options]
    <content>
\end{tikzpicture}
```

A *TikZ*-ben minden ábra alapvető építőeleme a vonal. Egy vonalat a kezdőpont koordinátáinak megadásával kezdünk, mint például (0,0), majd hozzáadunk egy vonal bővítő műveletet, erre a legkézenfekvőbb módszer csak a "- -" használata. A műveletet ezután a következő koordináta követi. Minden útvonalnak pontosvesszővel kell végződnie. A vonal megrajzolásához a *draw* parancsot használjuk.

Például a (0,0), (0,1), (1,0) pontok közötti háromszög felrajzolásához (egyik lehetséges módjaként) az alábbi parancsokat kell alkalmazni:

```
\tikz\draw (0,0) -- (0,1) -- (1,0) -- (0,0);
```

vagy

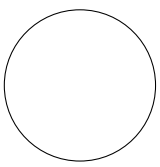
```
\begin{tikzpicture}
    \draw (0,0) -- (0,1) -- (1,0) -- (0,0);
\end{tikzpicture}
```

```
\draw (0,0) -- (0,1) -- (1,0) -- (0,0);
```

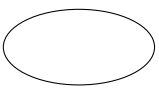
2.2.3. Elérhető diagram elemek

A körök és ellipszisek rajzolásához a *circle* és *ellipse* műveletet használhatjuk. A kör műveletet egy sugár követi kerek zárójelben, míg az ellipszis műveletet kettő paraméter követi, egy az x-irányra és egy az y-irányra vonatkozóan, amelyeket az *and* taggal választunk el, és kerek zárójelben helyezünk el.

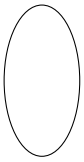
```
\draw (0,0) circle (1);
```



```
\draw (0,0) ellipse (1 and 0.5);
```

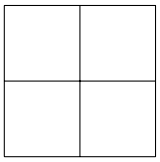


```
\draw (0,0) ellipse (0.5 and 1);
```

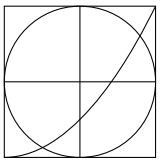


A rácsháló (*grid*), téglalap (*rectangle*), a parabola (*parabola*), és az ív (*arc*) rajzolásához további, a korábbiakhoz hasonlóan működő műveletek állnak rendelkezésünkre. Az alábbiak példák ezen konstrukciók használatára:

```
\draw (-1,-1) grid (1,1);
```



```
\draw (-1,0) -- (1,0);
\draw (0,-1) -- (0,1);
\draw (0,0) circle (1);
\draw (-1,-1) rectangle (1,1);
\draw (-1,-1) parabola (1,1);
```



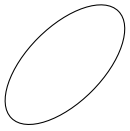
Az ív használható például egy szög ívének megrajzolásához. Megrajzolja a kör adott sugarú részét a megadott szögek között. Ezt a műveletet egy kerek zárójelben lévő hármasnak kell követnie. Az adattagokat kettőspontok választják el egymástól. Az első és a második a kör fokai, a harmadik pedig a sugara.

Például a (*30 : 160 : 2cm*) azt jelenti, hogy egy 2 cm sugarú körön 30 és 100 fok közötti ív lesz:

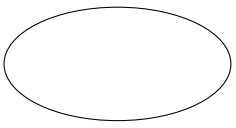
```
\draw (0,0) arc (30:100:2cm);
```

Az elemek elforgatásához vagy skálázásához egy *rotate* vagy *scale* opciót is hozzáadhatunk:

```
\draw[rotate=45]
        (0,0) ellipse (1 and 0.5);
```

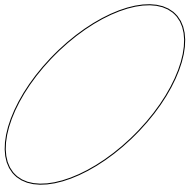


```
\draw[scale=1.5]
        (0,0) ellipse (1 and 0.5);
```



Természetesen használhatjuk a két opciót együttesen is az alábbi szerint:

```
\draw[rotate=45, scale=1.5]
        (0,0) ellipse (1 and 0.5);
```



A *color* csomagot hozzáadva a *definecolor* paranccsal definiálhatunk és használhatunk színeket az alábbiak szerint:

```
\definecolor{myRGB}{RGB}{255, 51, 76}
\definecolor{myrgb}{rgb}{1, 0.2, 0.3}
\definecolor{mycmyk}{cmyk}{0, 0.8, 0.7, 0}
\definecolor{mygray}{gray}{0.3}

\draw [myRGB] (0,1) -- (1,1);
\draw [myrgb] (0,0.5) -- (1,0.5);
\draw [mycmyk] (0,0) -- (1,0);
\draw [mygray] (0,-0.5) -- (1,-0.5);
```

Az általunk definiált színeknek négyféle megadási módja lehetséges:

RGB: a szabványos RGB-modell szerinti megadással megegyezik: a három színkomponens a piros (R), zöld (G), és a kék (B), melyek értéke $[0, 255]$ közötti egész számok.

rgb: a fenti *RGB*-modell $[0, 1]$ közötti normalizált értékei.

cmyk: cián (c), magenta (m), sárga (y) és fekete (k), mely négy [0,1] közötti, vesszővel elválasztott, számból álló lista, amelyek a kereskedelmi nyomtatók által használt szubtraktív CMYK-modell szerint határozzák meg a színt.

gray: ez egy szürke skála, melynek megadása egyetlen szám, ami $[0, 1]$ közötti.

A következő színeket érhetjük el bármiféle definiálás nélkül:

white (), *lightgray* (), *gray* (), *darkgray* (), *black* (), *red* (), *violet* (), *purple* (), *magenta* (), *pink* (), *green* (), *lime* (), *olive* (), *brown* (), *orange* (), *yellow* (), *blue* (), *cyan* (), *teal* ().

A *fill* paranccsal kitölthetjük a megadott színnel a bármely alakzat által határolt tartományt. Az aktuális rajzolt görbe lezárásához használhatjuk a `--` `cycle` parancsot. A szín argumentumhoz használhatjuk a szín nevét, például *green* (), *white* () és *red* (), vagy keverhetjük a színeket, mint a *red!40!white*, ami azt jelenti, hogy 40% pirosat és 60% fehéret fogunk keverni.

```
\fill[red!40!white]
        (0,0) -- (0,1) -- (1,0) -- cycle;
```

A kitöltést meg lehet adni egyszerűen a *draw* parancs paramétereként is, ám ilyenkor a körvonal is rajzolva lesz:

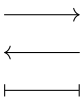
```
\draw[fill=red!40!white]
        (0,0) -- (0,1) -- (1,0) -- cycle;
```

Ez a színmegadás természetesen használható a körvonal színének módosítására is:

```
\draw[draw=green, fill=red!40!white]
        (0,0) -- (0,1) -- (1,0) -- cycle;
```

A vonalakat végpontjait testreszabhatjuk, így nyilakat hozhatunk létre:

```
\draw [->] (0,1) -- (1,1);
\draw [<-] (0,0.5) -- (1,0.5);
\draw [|-|] (0,0) -- (1,0);
```

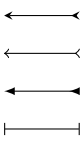


Hasonlóan tudunk görbét is rajzolni. A *TikZ* Bézier-görbe rajzolását teszi lehetővé. A lenti példában egy harmadfokú Bézier-görbe szerepel, így négy paraméter található: a kezdőpont, a két kontrollpont, és a végpont. A kontrollpontok a *.. controls* kulcsszó után adhatók meg *and* szóval elválasztva, és a két paraméter után szintén szükséges a két pont kirakása.

```
\draw [latex-latex]
    (0,0) .. controls (0,1) and (1,0) .. (1,1);
```

A nyílhegyek esetében is rengeteg lehetséges opció van, a két végponton lévő nyílhegy szabadon variálható. Egy pár ezek közül:

```
\draw [stealth-stealth reversed] (0,1.5) -- (1,1.5);
\draw [to-to reversed] (0,1) -- (1,1);
\draw [latex-latex reversed] (0,0.5) -- (1,0.5);
\draw [|-|] (0,0) -- (1,0);
```



Az *arrows.meta* könyvtár további nyílhegyeket tartalmaz, ebből két darabot emelnék ki, melyek jól használhatók a fentiekkel kombinálva:

```
\draw [{Stealth[open]}-stealth] (0,0.5) -- (1,0.5);
\draw [{Latex[open]}-latex] (0,0) -- (1,0);
```

Ha több szegmenst rajzolunk, a nyilak az első és az utolsó szegmens végpontjainál helyezkednek el. Ez, többek között, a tengelyek rajzolásához kényelmes:

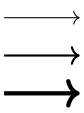
```
\draw [<->] (0,1) -- (0,0) -- (1,0);
```

Az attribútumokkal a vonalvastagságok módosítására is van lehetőség:

```
\draw [thin] (0,1) -- (1,1);
\draw [thick] (0,0.5) -- (1,0.5);
\draw [ultra thick] (0,0) -- (1,0);
```

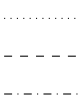
Azonban ügyelni kell rá, hogy a vonal vastagsággal arányosan nő a nyíl mérete is:

```
\draw [->, thin] (0,1) -- (1,1);
\draw [->, thick] (0,0.5) -- (1,0.5);
\draw [->, ultra thick] (0,0) -- (1,0);
```



A vonal stílusának módosítására is van mód, a jelentős számú lehetőség közül az alábbiakat emelném ki:

```
\draw [dotted] (0,1) -- (1,1);
\draw [dashed] (0,0.5) -- (1,0.5);
\draw [dashdotted] (0,0) -- (1,0);
```



A pontok és szaggatott vonalak sűrűsége a *loosely* (lazán) és a *densely* (sűrűn) jelzővel adható meg.

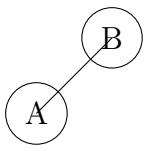
```
\draw [dashed] (0,1) -- (1,1);
\draw [loosely dashed] (0,0.5) -- (1,0.5);
\draw [densely dashed] (0,0) -- (1,0);
```

Az eddigi testreszabással kapcsolatos paraméterek akár együttesen is szerepelhetnek:

```
\draw [|-latex, red, thick, dashed] (0,0) -- (1,0);
```

Ahhoz, hogy szöveget adjunk a képhez, csomópontot kell hozzáadnunk az útvonalhoz, mint a következőben:

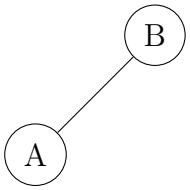
```
\draw
    (1,1) node[circle,draw]{A}
        --
    (2,2) node[circle,draw]{B};
```



A csomópontok az útvonal aktuális pozíciójába kerülnek, a [*circle,draw*] opció a szöveget egy körrel veszi körül, amely az aktuális pozícióba rajzolódik.

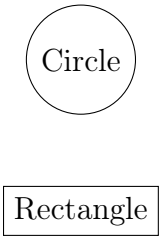
Adott esetben szükség lehet rá, ha a csomópont az aktuális koordináta jobb oldalán vagy fölött lenne. Ehhez az *anchor* attribútumot kell megadni, és a megfelelő oldal nevét:

```
\draw
(1,1) node[anchor=north east, circle, draw]{A}
        --
(2,2) node[anchor=south west, circle, draw]{B};
```



A csomópont stílusok egyenként tetszés szerint módosíthatóak. Ezekre szintén attribútumokat használunk:

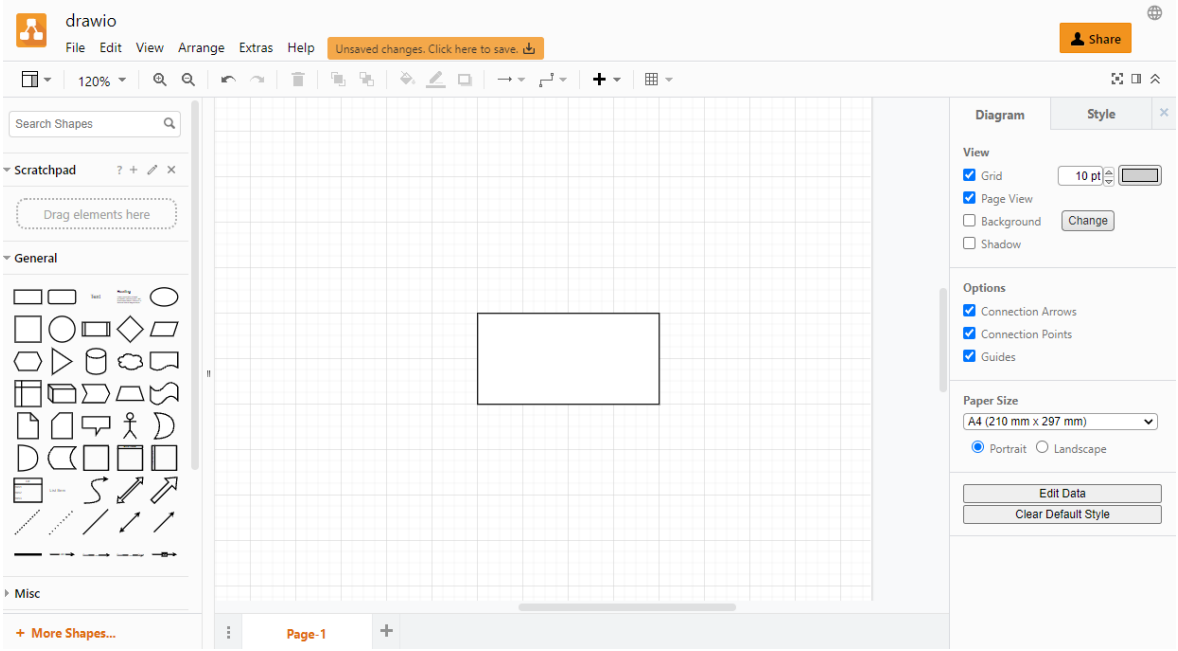
```
\draw (0,2) node[circle, draw]{Circle};


\draw (0,0) node[rectangle, draw]{Rectangle};
```



2.3. Szerkesztőeszközök

A következőkben a már meglévő, és népszerű szerkesztőkörnyezetek kerülnek bemutatásra, valamint kifejtésre kerülnek az összehasonlító szempontok, amik kritériumok lesznek a készülő alkalmazás felé.

2.3.1. draw.io



2.1. ábra. A draw.io kezelőfelülete [12]

A kezelőfelület fejlécében az általános menüpontok szerepelnek. Az alkalmazás közepén van maga a felület, ahol az ábrák szerkeszthetők. Ezt fogja körbe a bal oldalról az

ábrákkal kapcsolatos, míg jobb oldalon a diagram beállításai, és előre definiált stílusok közül lehet választani.

Az alapelemek csoportosítva vannak kinézetük alapján lenyitható menüpontokként.

Az ábra kiválasztása után rögtön megjelenik a felület közepén, ezt követően lehet elhelyezni. Az ábrák szerkesztésénél több lehetőség áll rendelkezésünkre. Az előre definiált pár stíluson felül többféleképpen is meg lehet adni saját színeket is: RGB színválasztóval, hexakódokkal, és gyakran használt színekkel egyaránt. Be lehet állítani az adott ábrára kitöltőszínt, betűszínt, vízszintes és függőleges igazítást, árnyékot, bemetszést.

Bármelyik ábrára helyezhető szöveg, ennek a stílusa igazodik az alakzat stílusához. Az objektumok bárhol összekapcsolhatók, de megjelennek ajánlott pontok is: 0, 25, 50, 75 és 100%-on. A rétegek nem jelennek meg külön oldalon, az ábrák sorrendje számít, hogy melyik jelenik meg felül. A kijelölés téglalap alapú, a kijelölt elemeket lehet másolni, törölni, szerkeszteni. Az ábrák csak rácspontokhoz igazíthatók.

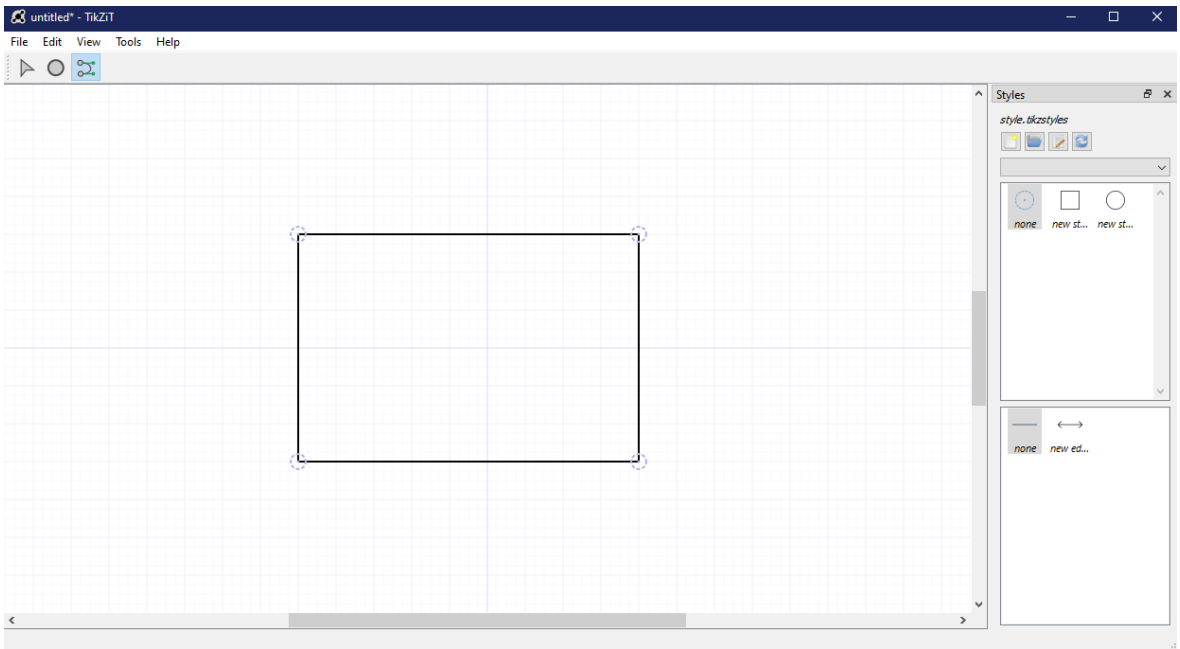
Az alkalmazás rendelkezik Undo-Redo funkciókkal.

Az elkészített diagram menthető különböző fájlformátumokban, automatikusan mentésre kerülnek a módosítások.

2.3.2. TikZiT

A *TikZiT* leginkább gráfok rajzolására használható. A vezérlő elemek az alkalmazás fejlécében helyezkednek el. A grafikus alapelemek szintén itt találhatóak, számuk nem kiemelkedő: a kijelölésen kívül van egy gráf csomópont lerakásához és egy gráf éleinek berajzolásához egy gomb.

Mind a csomópont, mind a gráf stílusa módosítható, fájlként menthető, és betölthető, a programon belül és kívül is (a kód ismeretében) szerkeszthetők. Három lehetőség van: alak, szín, kitöltés.



2.2. ábra. A TikZiT kezelőfelülete [21]

A szín és kitöltés kiválasztása történhet előre definiált alapszínekből, de lehetőség

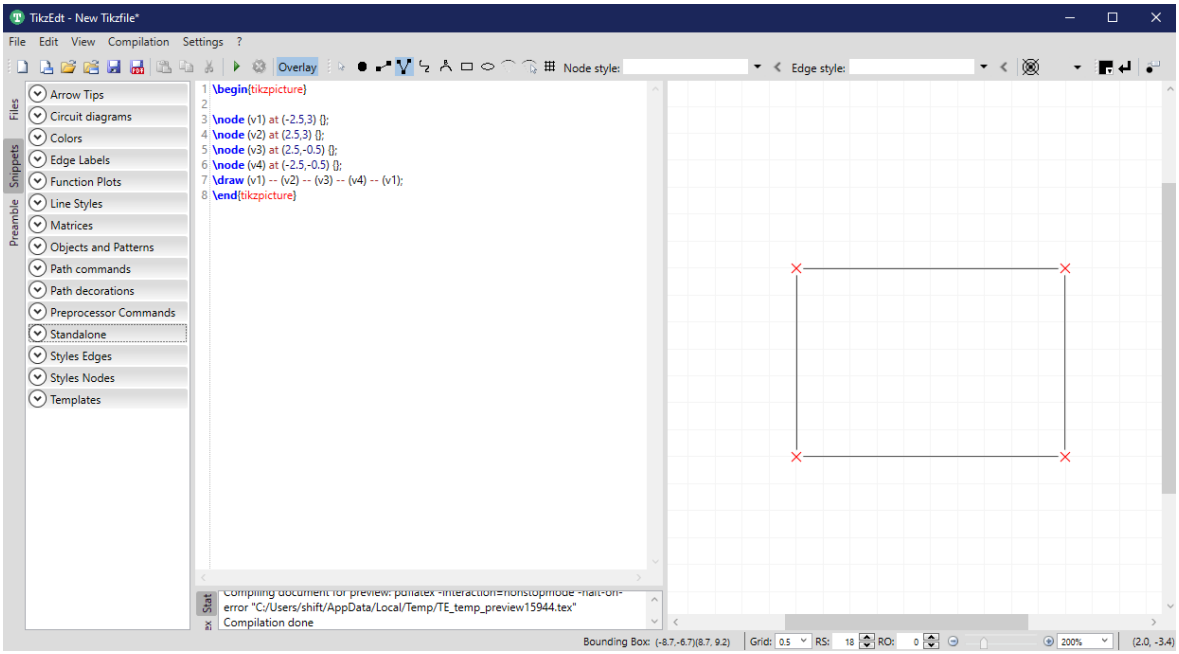
van RGB színskálából kiválasztásra vagy hexakód megadására is. Szöveg a gráf csomópontjainak adható. A csomópontokat összekötő élek a két pont helyzetétől függenek, csak az él hajlítására van lehetőség.

A csomópontok rétegződését a lerakás sorrendje határozza meg, utólag csak kijelölés után van lehetőség előre vagy hátra küldeni az adott elemet. A kijelölés téglalap alakú, a kijelölt elemek másolhatók, törölhetők, stílusuk szerkeszthető.

A program rendelkezik Undo-Redo funkciókkal, a rajzolófelületen lehetőség van nagyításra és kicsinyítésre egyaránt.

A kész ábrák mentése fájlba történik, ezek betölthetők későbbi szerkesztésre is, automatikus mentés nincs.

2.3.3. TikzEdt



2.3. ábra. A TikzEdt kezelőfelülete [20]

A *TikzEdt* esetében három hasábra osztható a felület: elsőben az előre definiált LaTeX kódrészletek, másodikban a LaTeX kódszerkesztő, és végül a harmadikban a lefordított kód előnézete jelenik meg.

Az alapelemek a felső sávban jelennek meg: főleg gráfokat és egyszerűbb elemeket tartalmaz. Az elemek, gráfok, egyenletek, szövegek stílusa és színei csak kód szinten szerkeszthetők, de vannak választható opciók.

A rétegződés a lerakás sorrendjében jön létre, utólagos módosításra nincs lehetőség. Az elemek automatikusan rácspontokhoz igazodnak.

A kijelölés téglalap alakú, a kijelölt elemek csak törölhetők. A program rendelkezik Undo-Redo funkciókkal, a rajzolófelületen lehetőség van nagyításra és kicsinyítésre egyaránt.

A kész ábrák mentése fájlba történik, automatikus mentés nincs.

2.3.4. tikzcd-editor

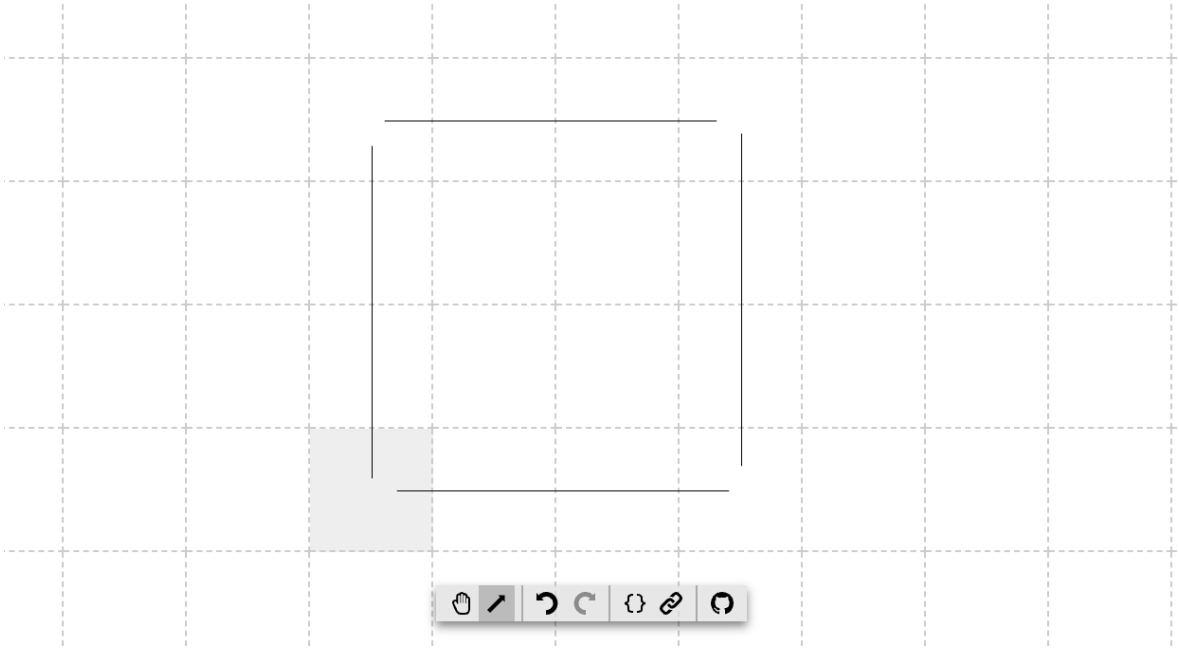
A *tikzcd-editor* klasszikus felülettel nem rendelkezik, csak egy négyzet alapú rácsháló fogad megnyitáskor. A rácsok közepére igazítva lehet nyilakat rajzolni, és szövegeket írni, de utólag van lehetőség a fel-le mozgatásra.

A stílusok nem testreszabhatók, csak pár alap stílusból lehet választani, mint a szaggatott vagy dupla vonal, módosítható a nyíl eleje, illetve vége. A rácsokon belül gombnyomásra tudunk hurkot rajzolni. A nyilakra és vonalakra kifejezések rakhatók. Az elemek színei nem módosíthatók, ezen a felületen kizárólag fekete vonalszínnel lehet dolgozni.

A rétegződés a lerakás sorrendjében van. Az elemek kijelölése csak egyesével működik, nincs lehetőség téglalap vagy esetleg lasszó alapú kijelölésre. A kijelölt rács a felületen szabadon mozgatható, és áthelyezhető üres cellákba.

A szerkesztő rendelkezik Undo-Redo funkciókkal, de mentésre nincs lehetőség, csak a LaTeX kód kimásolására, és az ábrához tartozó link utólagos betöltésére.

Az alkalmazás felépítését megnézve a felület egy felsorolás, aminek a stílusa erőteljesen testre lett szabva, hogy a rácsháló kinézetet megkapja. A rácsok szerkesztésével igazából a felsorolás pontjai kerülnek módosításra.



2.4. ábra. tikzcd-editor kezelőfelülete [15]

2.3.5. Összehasonlítás

Szempontok, amik alapján az összehasonlítást végzem:

Elrendezés: Az elrendezés az, amivel a szerkesztő megnyitásakor a felhasználó először találkozik. Fontos, hogy átlátható, és egyszerűen kezelhető legyen. A leggyakoribb alkalmazás elrendezések közül kettőt emelnék ki: egyik az fejléc és maga a szerkesztőfelület, a másik a hasáb alapú felépítés, ahol az alkalmazás részei egymás mellett jelennek meg.

Alapelemek megjelenítése: Az alapelemek helyzete is épp ilyen fontos, a felhasználó szempontjából kifejezetten előnyös, ha megfelelő (logikus) szempontok alapján csoportosítva vannak az alapelemek, és nem csak egymás után fel vannak sorolva, akár egy mátrix alapú felépítésben.

Tulajdonságok szerkesztése: Ha már leraktunk egy objektumot, akkor elvárható az, hogy a lent lévő elem szerkeszthető is legyen: ez érinti a helyzetet, színeket, és a stílusokat is. Itt szükséges lehetőséget adni a szerkesztendő elemek kiválasztására, akár magán a rajzolófelületen, akár egy listában kijelölve.

Szövegek szerkesztése: Elérhetővé kell tenni a felhasználó számára a szövegek szerkesztését: szabadon, vagy a csomópontokon belüli megjelenítést. Ebben az esetben is kiválaszthatónak kell lennie a szerkesztendő szövegeknek.

Színek megadási módja: Az ábrák színezhetők, így a szerkesztőnek is tudni kell kezelni a színeket: legalább az előre definiált színeket kiválaszthatóvá kell tenni.

Objektumok összekapcsolási módja: A már kész objektumokat milyen módon lehet összekötni: előre definiált fix helyeken, az körvonalain bárhol, vagy egyáltalán nincs lehetőség összekapcsolásra.

Objektumok automatikus igazítása: Rendelkezik-e a szerkesztő olyan funkcióval, amely az objektum pontjait automatikusan a vászon bizonyos pontjaira illeszti, és ez az opció esetleg be- és kikapcsolható.

Rétegek kezelése: Van-e lehetőség rétegek létrehozására, vagy sem, az elemek sorrendje a mérvadó, vagy van lehetőség utólag rendezni az objektumokat.

Kijelölés: Milyen formában van lehetőség objektumok kijelölésére: téglalap, lasszó, kattintás alapú kijelölés vagy egyáltalán nincs lehetőség objektumok kijelölésére. Téglalap és lasszó alapú kijelölés esetén egyszerre több elem is kijelölhető, még a kattintás alapú kijelölés esetén egy kattintással csak egy elem jelölhető ki, esetleg funkciógombokkal (mint például a *CTRL*, vagy az *ALT* billentyű) van lehetőség több elemet kijelölni.

Másolás és beillesztés: A kijelölt objektumot van-e lehetőség másolni valamilyen formában a vászonra: vezérlőgombok a szerkesztőben vagy az alkalmazások között népszerűnek számító $CTRL + C$ és $CTRL + V$ billentyűkombinációk elérhetők az alkalmazáson belül.

Zoom-olás: A vászont lehet-e valamilyen formában nagyítani a precízebb rajzolás miatt vagy esetleg kicsinyíteni a nagyobb rajzolási felület érdekében.

Undo-Redo funciók: Az alakzat rajzolásának vagy módosításának visszavonására van-e lehetőség. A véletlenül visszavont szerkesztések megismétlésére milyen lehetőségek állnak rendelkezésre. A könnyű elérés miatt célszerű ezeket is billentyűkombinációkra rakni: $CTRL + Z$ és $CTRL + Y$ billentyűkombinációk.

Automatikus mentés: A szerkesztő menti-e a rajzolás közben a munkamenetet, vagy teljes mértékben a felhasználóra van bízva a mentés folyamata.

2.1. táblázat. Szerkesztőeszközök összefoglalása az előző szempontok szerint

<table>
<tr><td></td><td>draw.io</td><td>tikzit</td><td>tikzedt</td><td>tikzcd</td></tr>
<tr><td>Elrendezés</td><td colspan="3">hasáb felépítés</td><td>alul</td></tr>
<tr><td>Alapelemek megjelenítése</td><td>mátrix alapú, csoportosítva</td><td>sorban</td><td>lenyíló listás rendezés</td><td>sorban</td></tr>
<tr><td>Tulajdonságok szerkesztése</td><td colspan="2">előre definiált stílusok</td><td>csak kód szinten</td><td>kijelölés után</td></tr>
<tr><td>Szövegek szerkesztése</td><td>bárhol</td><td colspan="2">csak csomópontokon</td><td>rácsok közepén</td></tr>
<tr><td>Színek megadási módjai</td><td colspan="2">előre definiált színek, RGB, és HEX megadás</td><td>csak kód szinten</td><td>nem</td></tr>
<tr><td>Objektumok összekapcsolási módjai</td><td>bárhol</td><td colspan="2">középen</td><td>nem</td></tr>
<tr><td>Objektumok automatikus igazítása</td><td>igen</td><td colspan="2">nem</td><td>igen</td></tr>
<tr><td>Rétegek kezelése</td><td colspan="4">rajzolási sorrend</td></tr>
<tr><td>Kijelölés</td><td>téglalap alapú</td><td colspan="3">kattintás alapú</td></tr>
<tr><td>Másolás és beillesztés</td><td colspan="2">igen</td><td>csak kód szinten</td><td>nem</td></tr>
<tr><td>Zoom-olás</td><td colspan="3">igen</td><td>nem</td></tr>
<tr><td>Undo-redo funkciók</td><td colspan="4">igen</td></tr>
<tr><td>Automatikus mentés</td><td>igen</td><td colspan="3">nem</td></tr>
</table>

3. fejezet

Követelmények a saját szerkesztővel szemben

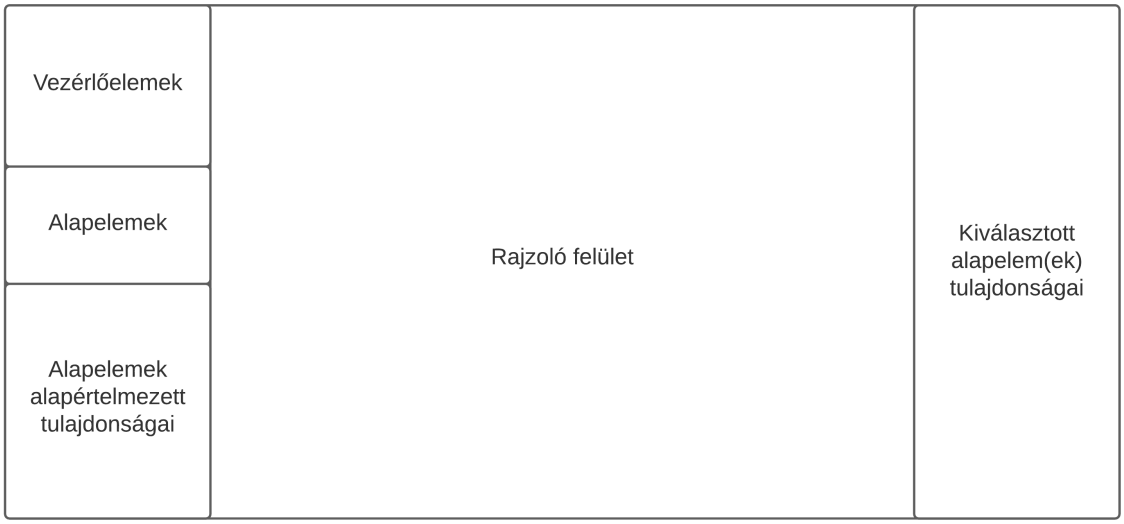
Egy weboldal tartalmának felhasználói megértése gyakran attól függ, hogy a felhasználó hogyan érti meg a weboldal működését. Ez a felhasználói élménytervezés része. A felhasználói élmény egy weboldalon az elrendezéssel, az egyértelmű utasításokkal és a címkézéssel függ össze. Az, hogy a felhasználó mennyire érti meg, hogyan tud interakcióba lépni egy webhelyen, szintén függhet a webhely interaktív kialakításától. Ha a felhasználó érzékeli a weboldal hasznosságát, nagyobb valószínűséggel fogja azt továbbra is használni. Azok a felhasználók, akik gyakorlottak és jártasak a webhelyek használatában, ennek ellenére hasznosnak találhatnak egy markánsabb, de kevésbé intuitív vagy kevésbé felhasználóbarát webhelyfelületet. A kevésbé tapasztalt felhasználók azonban kisebb valószínűséggel látják a kevésbé intuitív weboldal-felület előnyeit vagy hasznosságát. Ez a tendencia az univerzálisabb felhasználói élmény és a könnyebb hozzáférés irányába mutat, hogy a lehető legtöbb felhasználónak megfeleljen, függetlenül a felhasználói készségektől. A felhasználói élménytervezés és az interaktív tervezés nagy részét figyelembe veszik a felhasználói felület tervezésénél.

3.1. Tipikus elrendezések vizsgálata

A felhasználói felület kialakításának egy részét befolyásolja az oldal elrendezésének minősége. Az elrendezés tervezésekor például figyelembe kell venni, hogy a webhely oldalelrendezésének a különböző oldalakon konzisztensnek kell-e maradnia. Az oldal pixelszélessége szintén létfontosságúnak tekinthető az objektumok igazításához az elrendezés tervezésében. A legnépszerűbb fix szélességű weboldalak általában ugyanolyan beállított szélességgel rendelkeznek, hogy megfeleljenek az aktuálisan legnépszerűbb böngészőablaknak, a képernyőfelbontás és a monitorméret mellett.

Az elkészült webes alkalmazás elrendezés szempontjából átláthatónak kell lennie, ehhez a legegyszerűbb megvalósítás a hasáb alapú elrendezés: két, esetleg három hasáb. Egyikben az alkalmazás menüje, másik hasábban pedig maga a szerkesztő foglal helyet, a harmadik hasábban a funkciókhoz tartozó ablak jelenjen meg.

Az alkalmazás legyen reszponzív akár a felület, akár a funkciók megjelenítése szempontjából. A reszponzív webdesign egy újabb megközelítés, amely a *CSS3*-on és a *CSS @media*[3] szabály továbbfejlesztett használatán keresztül az oldal stíluslapján belül a készülékenkénti specifikáció mélyebb szintjén alapul.



3.1. ábra. Az alkalmazás felépítése

3.2. Alapelemek megjelenítése és kiválasztása

Az alapelemek megjelenése és kiválasztása egyaránt fontos a felhasználó megragadásában. Ha ezek átláthatók, és jól használhatók, akkor nagyobb célközönséghez juthat el az alkalmazásunk.

3.2.1. Alapelemek megjelenítése

A funkciók a weboldal első hasábját foglalja magába, itt találhatók meg az alapelemek is a vezérlőelemeken és a kiválasztott elem tulajdonságainak kiválasztásán felül.

Az alkotóelemek a kis hely miatt két oszlopban lelhetők fel, tehát az alkalmazás mátrix alapú elrendezést használ. Célszerű az elemeket csoportosítani olyan tulajdonságok alapján, amiben megegyeznek: például a vonal és a görbe jelenjen meg egymás mellett a mátrixban.

3.2.2. Alapelemek kiválasztása

A rajzolási mód kiválasztása után automatikusan ki legyen választva a mátrix első eleme, mint aktuális elem. Az alapelemek kiválasztása a mátrix alapú listában történik beavatkozással. A felhasználó egérrel kiválasztja a számára szükséges elemet. A kiválasztás után megjelenik az adott elemekhez tartozó tulajdonságok listája, amivel ezek módosíthatók is.

3.3. Tulajdonságok szerkesztési lehetőségei

A tulajdonságok szerkesztése szintén fontos akár az új, akár a meglévő elemek tulajdonságait szeretnénk módosítani. A felhasználó nem feltétlenül csak fekete-fehér ábrákat szeretne szerkeszteni.

3.3.1. Új elemek esetében

Új elemek esetében minden újonnan lerakott elemre érvényesüljön az itt megadott beállítás. Ez érinti az elemek színét, kitöltési színét, vonal vastagságát és mintázatát is. Az elemek kezdeti és végpontjai kattintásra kerülnek a vászonra. Ahol szükséges egyéb tulajdonság is, mint például körvonal esetében a kezdő és végszögek külön beírhatók legyenek.

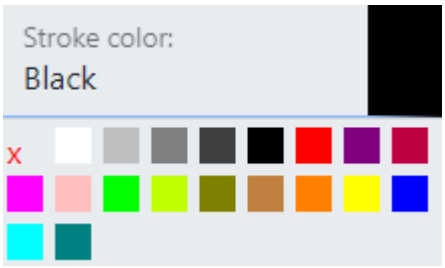
3.3.2. Meglévő elemek esetében

A már lent lévő elemek esetében kijelölés után minden tulajdonságnak szerkeszthetőnek kell lennie. Ehhez a szerkesztő jobb oldalán lévő hasáb ad majd helyet. A kijelölt elemek meglévő tulajdonságait be kell tölteni, és változtatás esetén az adott elem tulajdonságait ez alapján módosítani. Fontos, hogy csak a releváns tulajdonságok jelenjenek meg.

3.4. Színek megadási módjai

A szerkesztőnek tudnia kell kezelni a színeket, és ezeknek kiválaszthatónak kell lenniük. Az LaTeX által előre definiált színeket elérhetővé kell tenni a felhasználónak.

A színek kiválasztásához célszerű lenyíló menüt használni, ezzel megkönnyítve ezeknek az elérését.



3.2. ábra. Példa a színek kiválasztására

Ebben az esetben megjelenítésre kerül a módosított tulajdonság, a kiválasztott szín, és a választható színek egyaránt. Az első elem a címek között jelöli azt, amikor nem kerül szín kiválasztásra: használható a körvonal eltüntetésére, vagy az ábra fehérrel történő kitöltés elkerülésére.

3.5. Objektumok automatikus igazítása

Az alkalmazásnak tudnia kell kezelni a rácspontokat, és az ábrák egyes pontjait ezekhez kell igazítania. Az igazításnak meg kell történni ábra lerakásakor és már lent lévő elem kezdeti és végpontjai módosításakor. A Bézier-görbe esetében a kontrollpontok szintén paraméteresen módosíthatónak kell lenniük.

3.6. Objektumok összekapcsolási módjai

A szerkeszthetőség érdekében lehetőséget kell adni a felhasználónak a már lent lévő objektumok összekapcsolására. Az összekapcsolás legegyszerűbb módja a pontok rácspontokhoz kapcsolása, és a kijelölés során lehetőséget adni több elem kijelölésére. Ebben az esetben a kijelölt pontok összekapcsolódnak, és mozgatásra együtt kerülnek kiszámolásra.

3.7. Szövegek szerkesztése

Az alkalmazásnak kezelni kell a szöveget. A szövegeknek a vászonra lerakhatóknak és utólag szerkeszthetőeknek kell lenniük.

A szövegekbe beletartoznak a LaTeX matematikai módban írt kifejezései is, tehát például az

$$\int_a^b \qquad \bigcap_a^b \qquad \bigcup_a^b \qquad \sum_a^b \qquad \prod_a^b$$

kifejezéseknek meg kell tudnia jelenni.

3.8. Kijelölés

Az alkalmazásnak lehetővé kell tenni az elemek kijelölését. A kijelöléshez egérbillentyűket kell használni. A kijelölés után az adott elemek tulajdonságait módosítani lehet, másolni, és törölni.

Téglalap alapú kijelölés esetén egyszerre több elem is kijelölhető, még a kattintás alapú kijelölés esetén egy kattintással csak egy elem jelölhető ki, esetleg funkciógombokkal (mint például a *CTRL*, vagy az *ALT* billentyű) van lehetőség több elemet kijelölni.

3.9. Másolás és beillesztés

A "*másolás és beillesztés*" kifejezés a szöveg vagy más adatok forrásból a célba történő másolásának népszerű, egyszerű módszerére utal. A módszer népszerűsége az egyszerűségéből ered, valamint abból, hogy a felhasználók vizuálisan - állandó tárolás nélkül - könnyen mozgathatják az adatokat a különböző alkalmazások között. Miután adatokat másoltunk a vágólapra, a vágólap tartalmát beilleszthetjük a céldokumentumba.

3.9.1. Másolás

A kijelölt elemek másolása billentyű lenyomásra működjön, a szokásnak megfelelően a *C* billentyűre vagy a *CTRL + C* billentyűkombinációra. A kijelölt elemek tulajdonságai (pozíció, szín, kitöltési szín...) kerüljenek mentésre egyaránt.

3.9.2. Beillesztés

A másolt elemek beillesztése szintén billentyű lenyomásra működjön, a szokott módon a *V* billentyűre vagy a *CTRL + V* billentyűkombinációra.

A beillesztett elemek ne a másolt elemeken, hanem kicsit eltolva jelenleg meg az átláthatóság és könnyebb szerkeszthetőség érdekében.

3.10. Kicsinyítés és nagyítás

Az oldal nagyításának két különböző módja van:

- a szövegek átméretezése a betűméret növelésével vagy csökkentésével, a vízszintes görgetés elkerülése érdekében a képek méretének változatlanul hagyásával.
- valódi átméretezés, amely a képeket, egyéb multimédiás objektumokat és a nézetablakokat is átméretezi.

Az alkalmazásnak támogatnia kell a kicsinyítését és nagyítását, legalább a vásznon megjelenő alapelemek méretének nagyításával vagy csökkentésével. Ezt célszerű az egér görgőjére hivatkozva módosítani: ha felfelé görgetünk, akkor nagyítsuk a vásznak, ellenkező esetben kicsinyítjük.

3.11. Undo-Redo funkciók

Az elkészült alkalmazásnak biztosítani kell a visszavonás és az újra lerakás funkcióit. Sokszor fordul az elő, hogy a felhasználó véletlenül más elemet rajzol a vászonra, vagy esetleg más elem tulajdonságait módosítja, így a felhasználói élmény növelése céljából szükség van ezeknek a hibáknak a visszavonására.

3.12. Mentés, és automatikus mentés

3.12.1. Mentés

Mentés során a felhasználó megkapja a szerkesztett ábrát LaTeXbe visszailleszthető állapotban. Ez egyaránt jelenti a LaTeX kódot, valamint egy opcionálisan letölthető *.tex* formátumú fájlt, amely az *input* paranccsal betölthető tetszőleges LaTeX állományba.

A mentés során a felhasználó kap egy hivatkozási kódot is, amely a későbbi betöltéshez szükséges. Ez egy kódolt szöveg lenne, amelynek az a jelentősége, hogy elfedje a háttérben lévő "adatbázist", amelyből a kimentéshez szükséges *JSON*[10] objektumot kapjuk. A JSON egy könnyű, szövegalapú, nyelvfüggetlen szintaxis az adatcsereformátumok definiálására. Az ECMAScript[9] programozási nyelvből származik, de nyelvtől független. A JSON a strukturált adatok hordozható ábrázolására szolgáló strukturálási szabályok kis halmazát határozza meg.

3.12.2. Automatikus mentés

Az automatikus mentés számos számítógépes alkalmazás és videojáték mentési funkciója, amely automatikusan elmenti a program vagy játék aktuális változásait vagy előrehaladását, így segít csökkenteni az adatvesztés kockázatát vagy hatását összeomlás, lefagyás vagy felhasználói hiba esetén. Az automatikus mentés jellemzően vagy előre meghatározott időközönként, vagy egy összetett szerkesztési feladat megkezdése előtt, közben és után történik. Hagyományosan olyan funkciónak tekintették, amely

alkalmazás- vagy rendszerhiba (például összeomlás) esetén védi a dokumentumokat, és az automatikus mentéses biztonsági mentéseket gyakran törlik, amikor a felhasználó befejezi a munkáját. A megvalósítás a fájl, az alkalmazás és az operációs rendszer szintjén is kihívásokkal jár.

Ezek alapján az alkalmazásnak támogatnia kell az automatikus mentés funkciót. A legegyszerűbb megvalósítása ennek a fix időközönkénti állapotmentés. Web alkalmazás révén ehhez a megoldás a *HTTP-sütik* használata.

A HTTP-sütik (vagy más néven böngészősütik vagy egyszerűen sütik) olyan kis adatblokkok, amelyeket egy webszerver hoz létre, miközben a felhasználó egy webhelyet böngészik, és amelyeket a webböngésző helyez el a felhasználó számítógépén vagy más eszközén. A sütik a weboldal eléréséhez használt eszközön kerülnek elhelyezésre, és egy munkamenet során egynél több süti is elhelyezhető a felhasználó eszközén.

4. fejezet

JavaScript implementáció

4.1. A p5 függvénykönyvtár

Az elkészült szerkesztő jelentős mértékben támaszkodik a *p5.js* függvénykönyvtár adta lehetőségekre.

4.1.1. Telepítés

A p5 függvénykönyvtár telepítéséhez csak hozzá kell adni a fájlunkhoz:

```
<html>
  <head>
    <script src="../p5.min.js"></script>
    <script src="sketch.js"></script>
  </head>
  <body>
  </body>
</html>
```

4.1.2. Használat

A p5.js függvényhívások a *sketch.js*-ben kapnak helyet globális névtér használat esetében:

```
function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(220);
  ellipse(50, 50, 80, 80);
}
```

A *setup* függvény a *sketch.js* betöltésekor fut le, célszerű iderakni a p5 vászon létrehozását (*createCanvas* függvényhívás, mely paraméterként várja a magasságot és a szélességet). A *draw* függvény a futás során folyamatosan meghívódik.

A dolgozatban a *p5.js* példányosított módban fut a modulok miatt.

```
import {setup, draw} from "./canvas/canvas.js";

const sketch = s => {

  s.setup = () => {
    setup();
  }

  s.draw = () => {
    draw();
  }

...

}

const P5 = new p5(sketch);

export {
  P5
}
```

Ebben az esetben a *P5* névtér alá kerül a vászon és a modulokban importálni kell a névtér hivatkozását, valamint az eddig globális *p5.js* függvényhívások is ezen névtér alá kerülnek. Ebben az esetben a fenti példa így kerül definiálása:

```
import {P5} from "../sketch.js";

const setup = () => {
  P5.createCanvas(400, 400);
}

const draw = () => {
  P5.background(220);
  P5.ellipse(50, 50, 80, 80);
}
```

4.2. Az alkalmazás felépítése

Az alkalmazás a 3.1. ábrán lévő blokkdiagram alapján készült el. Az ábrán lévő elemek nem minden funkcióban jelennek meg.

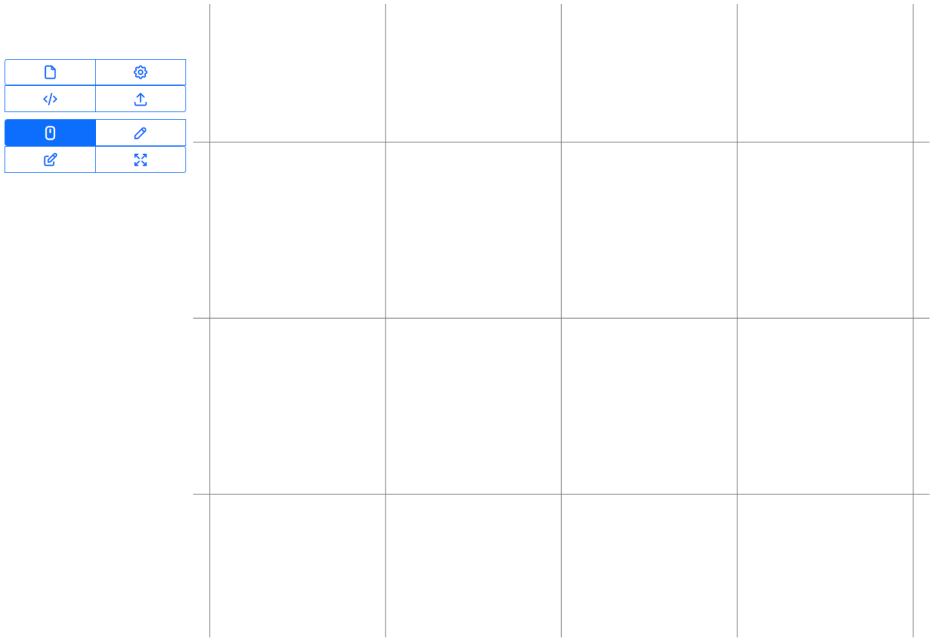
A következő fejezetben az egyes funkciók képernyői láthatók. Leírásra kerülnek az adott pont alatt lévő funkciók, és használatuk.

A fejezetben megjelenő képernyő mintaképeken a vászon ötszörös nagyításban van a jobb átláthatóság miatt.

4.2.1. Kezdőképernyő

Ez a képernyő jelenik meg az alkalmazás betöltésekor. A bal oldali menüben automatikusan ki van választva a vászon mozgatására alkalmas funkció.

Az egér bal gombjával tudjuk pozicionálni, míg görgővel kicsinyíteni vagy nagyítani a vásznat. Az egér középső gombját lenyomva pedig az alapértelmezett pozícióra és nagyításra visszaáll a vászon.



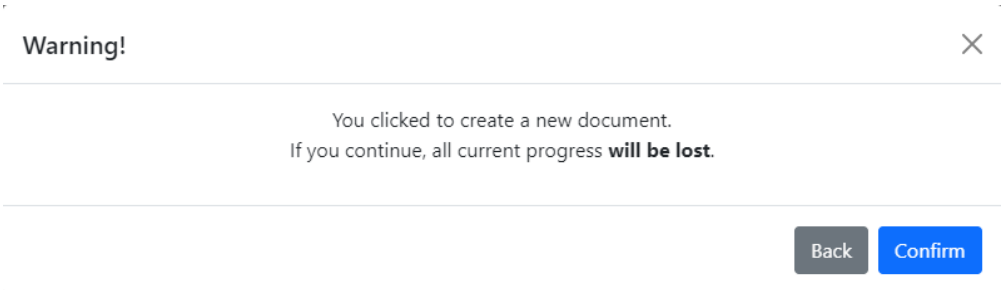
4.1. ábra. Az elkészült alkalmazás kezdőképernyője

Az alkalmazás bármely menüpont alatt támogatja a visszavonás műveleteit: az adott művelet visszavonható a *CTRL + Z* billentyűkombinációval, míg a *CTRL + Y* kombináció lehetőséget biztosít a visszavont művelet megismétlésére.

4.2.1.1. Vezérlőgombok

A bal oldalt megjelenő gombok közül a felső mátrix biztosítja az új dokumentum létrehozását, a beállításokat, a meglévő ábra mentését, és visszatöltését.

Az első ikonra (🗋) kattintva új dokumentumot hozhatunk létre. Ekkor a már meglévő ábra eltűnik, visszatöltésre nincs lehetőség.

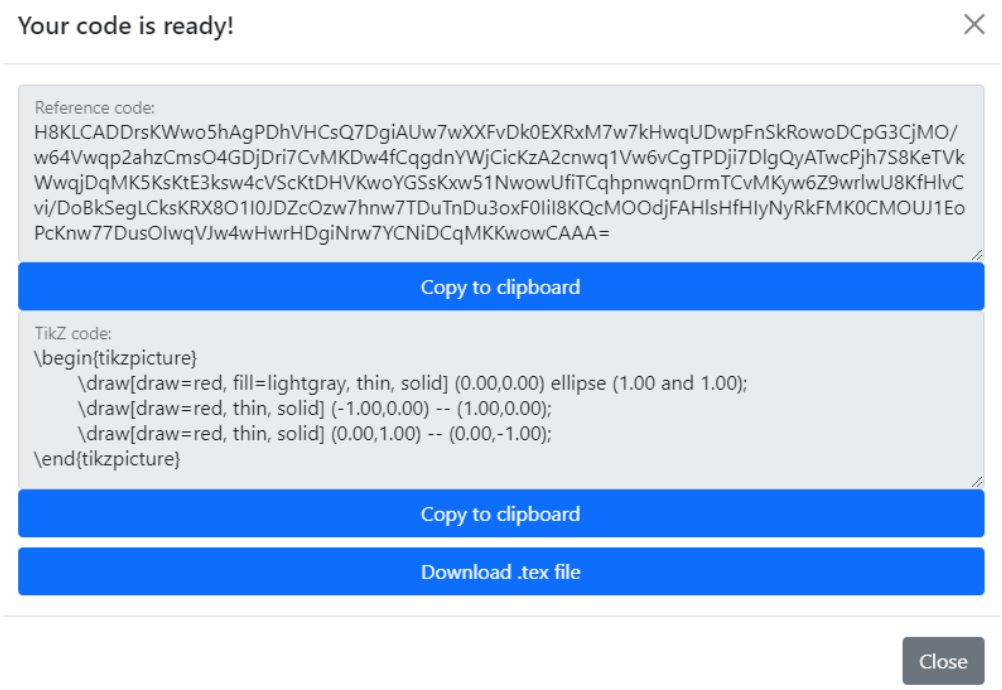


4.2. ábra. Új dokumentum létrehozásakor megjelenő figyelmeztető üzenet

A fogaskerék ikonnal (⚙) megnyithatjuk a beállításokat, itt van lehetőség az alapértelmezetten bekapcsolt rácsponthoz illeszkedést kikapcsolni, és az előnézetet bekapcsolni.

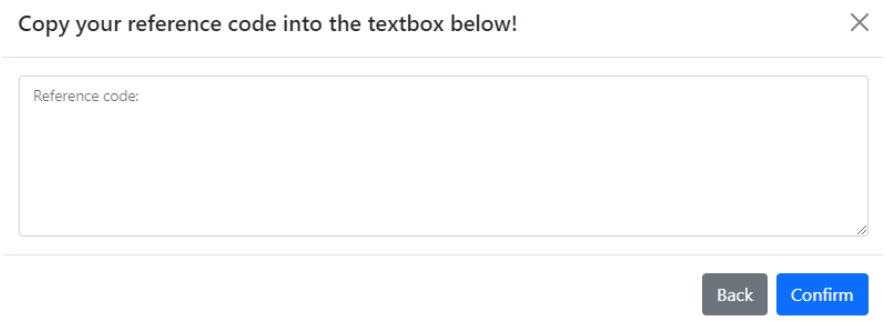
A következő két ikon felelős a már kész ábra kimentésére (</>), valamint a letöltött ábra visszamásolására (↥).

Az ábra kimentésekor egy felugró ablak fogad. Ekkor mentésre kerül a jelenlegi vászon állapota is süti formájában. A hivatkozási kód és a *Tikz* kód kimásolható a mezőből gombnyomásra is, valamint egybefűzve le is tölthetők. Ebben az esetben a letöltött *.tex* fájlban egymás alatt jelenik meg a hivatkozási kód, a *TikZ* ábra kódja, valamint a letöltött állomány létrehozásának időpontja. Ezek a *.tex* fájlok módosítás nélkül importálhatók az **input** paranccsal a meglévő dokumentumokba.



4.3. ábra. Az ábra mentésekor megjelenő ablak

Az ábra betöltésekor a már meglévő hivatkozási kód bemásolása után az alkalmazás betölti a kódhoz tartozó ábrát és kezdhetjük is a már meglévő ábra szerkesztését és kibővítését.

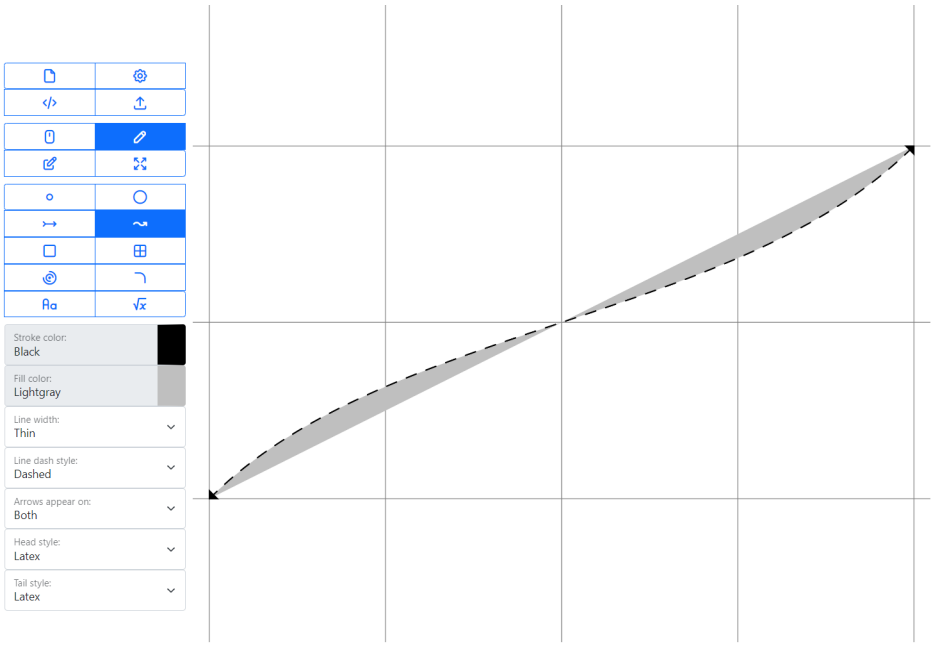


4.4. ábra. Az ábra betöltésekor megjelenő ablak

4.2.2. Rajzolás

A rajzolás mód kiválasztásával megjelennek a vezérlőelemek alatt az alapelemek mátrix alapú felsorolásban és a kijelölt elemhez tartozó tulajdonságok.

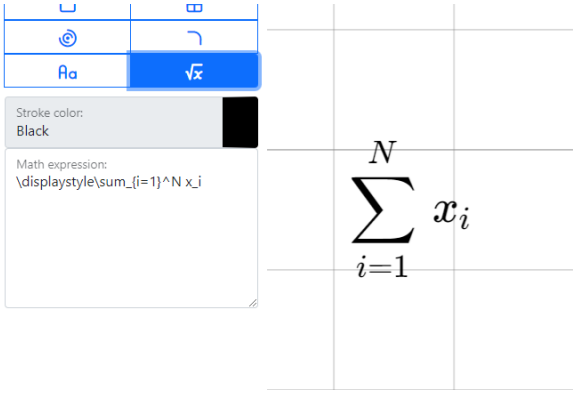
A megfelelő tulajdonságok beállítása után a rajzolás az egérrel történik. A pont, szöveg és matematikai kifejezések kirajzolása kattintásra történnek, míg a maradék alapelem "drag and drop" (*fogd és vidd*) megoldással kerül kirajzolásra, vagyis a kezdőpont a kattintás helye, és a végpont az egér gombjának felengedésének a pozíciója lesz.



4.5. ábra. A rajzolás funkciói rajzolt elemmel a vásznon

Amennyiben az előnézet be van kapcsolva a beállításokban, úgy megjelenik az egér helyén egy körvonal, amely jelzi a rajzolandó elem kezdőpontját, valamint a szöveg és a matematikai kifejezések esetében a teljes tartalom megjelenik a beállított tulajdonságok szerint.

4.2.2.1. Matematikai kifejezések elhelyezése

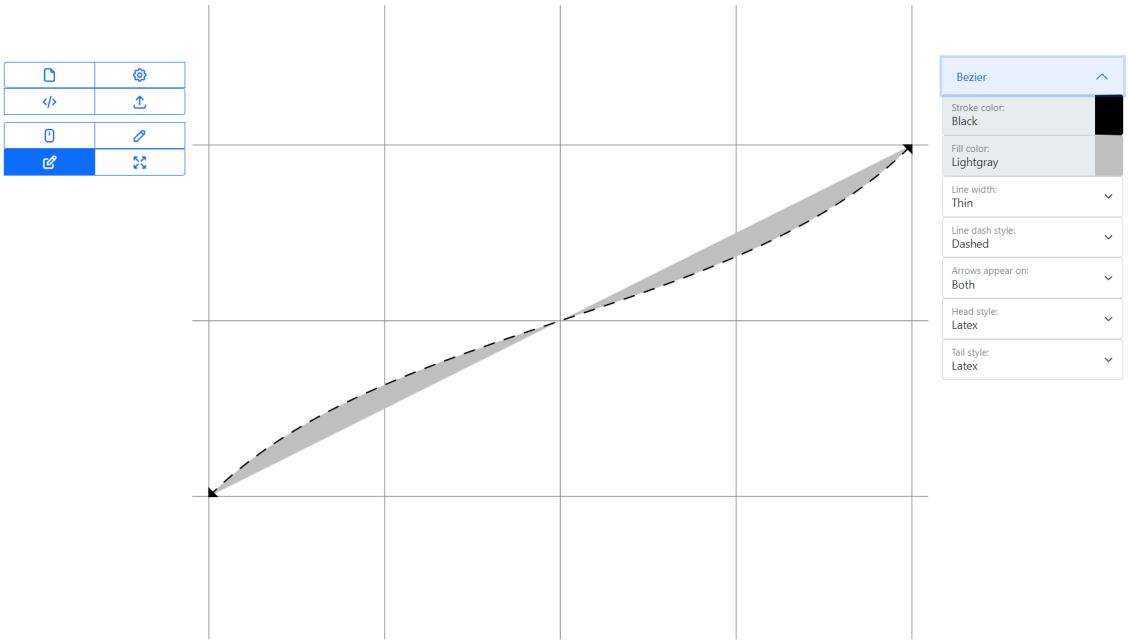


4.6. ábra. Példa egy matematikai kifejezés megjelenítésére

A matematikai kifejezések megjelenítéséhez egy szövegdobozba kell megadni a kifejezéseket. A szerkesztő támogatja az egyszerűbb matematikai környezeteket is.

4.2.3. Szerkesztés

A szerkesztés funkcióban lehetőség van a már lent lévő elemek kiválasztására. A kijelölés téglalap alakú, és a kezdő- és végpont szintén "drag and drop" módszerrel kerül megállapításra. A kijelölés közben egy világosszürke téglalap jelzi az aktuális kijelölt területet, így látszódik mely alapelemek esnek bele.



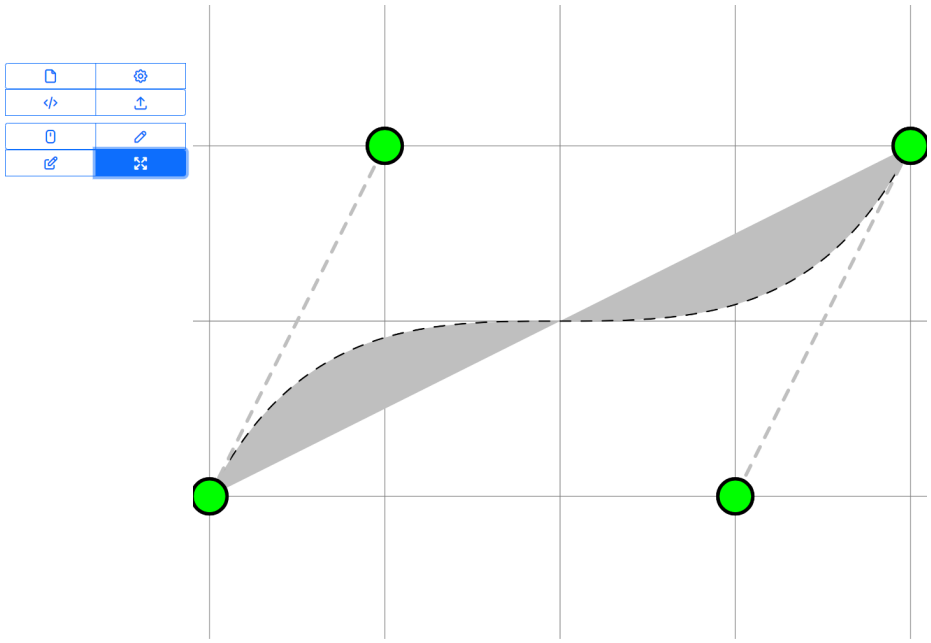
4.7. ábra. A szerkesztés képernyő kiválasztott elem után

A kijelölés véglegesítése után a rajzoló felület jobb oldalán megjelennek a kijelölt területbe beleeső alapelemek. Itt lehetőség van egyenként szerkeszteni a már lent lévő elemek tulajdonságait. A jelenlegi tulajdonságok automatikusan betöltődnek, módosításkor a vásznon automatikusan megjelenik, nincs szükség a módosítás mentésére.

Kijelölés után lehetőségünk van a másolásra, beillesztésre és törlésre egyaránt, ezek mindegyike billentyű lenyomásra működnek. A másolás a *CTRL + C*, a beillesztés a *CTRL + V*, még a törlés a *CTRL + X* billentyűkombinációval, és a *DELETE* billentyűvel egyaránt működik.

4.2.4. Mozgatás

A mozgatás funkcióban lehet a már lent lévő ábrák pontjait kijelölés után mozgatni. A mozgatandó elem kijelölése kattintásra történik, *CTRL* billentyű lenyomása közben van lehetőség több pont kijelölésére. A mozgatás szintén "drag and drop" megoldással működik. A kijelölt pont a kezdeti zöld helyett piros színre vált, és egér gombbal mozgatható. A felengedéssel egy időben, ha be van kapcsolva a rácspontra mozgatás, akkor ebben az esetben is megtörténik.



4.8. ábra. A mozgatásra alkalmas pontok megjelenítése a vásznon

4.3. Definiált osztályok bemutatása

4.3.1. Alapelemek osztályai

Ebbe a részbe kerülnek a menüben kiválasztható alapelemek osztályai, valamint az aktuális elem esetében a *p5.js* és *TikZ* közötti különbségek és megoldásuk.

4.3.1.1. Shape osztály

A *Shape* osztály minden alapelem szülője, itt kerül beállításra a majdnem az összes elemnél előforduló szín, betűszín, vonal vastagság és mintázat. A *draw* metódus beállítja a megadott tulajdonságokat a rajzolás előtt. A *toLatex* és a *fromJson* metódusok még itt üresek, értékeket a származtatott osztályokban kapnak.

4.3.1.2. Point osztály

A *Point* osztály egy pont tulajdonságait tárolja. Példányosítás során meg kell adni a pont két koordinátáját, és a tulajdonságokat tömb formájában. A jobb láthatóság érdekében *p5.js* és *TikZ* esetében egy kis méretű kör kerül kirajzolásra.

4.3.1.3. Ellipse osztály

Az *Ellipse* osztály ellipszis rajzolását teszi lehetővé. A példányosításhoz meg kell adni két koordinátapárt (egy x és egy y koordinátát), valamint a tulajdonságokat.

A *p5.js* és a *TikZ* is ugyanúgy támogatja az ellipszis rajzolását: meg kell adni a középpontot és a sugár méretét, így szükséges egy metódus, ami ezt kiszámolja. A *getCenterDiameter* metódus ezt teszi lehetővé:

```
static getCenterDiameter(start, end) {
  let center = {
    x: start.x + (end.x - start.x) / 2,
    y: start.y + (end.y - start.y) / 2
  };

  let diameter = {
    x: end.x - start.x,
    y: end.y - start.y
  };
  return {center, diameter}
}
```

4.3.1.4. Line osztály

A *Line* osztályban vannak definiálva a vonal rajzolásához szükséges metódusok. A példányosításhoz meg kell adni két koordinátapárt (egy x és egy y koordinátát), valamint a tulajdonságokat. A nyilak rajzolását is ez az osztály végzi, ugyanis tulajdonságként megkapja, hogy hol és milyen típusú nyíl kirajzolása szükséges.

A *p5.js* megvalósítás kicsit komplikáltabb, mint a *Tikz* megoldás. A *p5.js* esetében a *positionHead* metódus a kiválasztott végpontra lép, a *drawArrowHead* metódus pedig az aktuális pontra kirajzolja a kiválasztott nyílhegyet. A *drawArrowLine* metódus csak egyszerűen összeköti a két végpontot a nyílhegy függvényében.

A *TikZ* esetében pedig csak a *draw* paramétereként meg kell adni a nyílhegyet.

4.3.1.5. Bezier osztály

A *Bezier* osztály egy harmadfokú Bézier-görbe kirajzolását teszi lehetővé. A példányosításhoz meg kell adni két koordinátapárt (egy x és egy y koordinátát), valamint a tulajdonságokat. A program automatikusan számol lerakáskor két kontrollpontot a harmadolópontok helyére, mely utólag természetesen módosítható. A vonalaknál látott nyílhegyek itt is elérhetők.

A *p5.js* és a *TikZ* megadás rendre megegyezik: először meg kell adni a kezdőpontot, majd a két kontrollpontot és végül a végpontot.

4.3.1.6. Rectangle osztály

A *Rectangle* osztály egy tetszőleges téglalap létrehozására szolgál. A példányosításhoz meg kell adni két koordinátapárt (egy x és egy y koordinátát), valamint a tulajdonságokat.

A *p5.js* és a *TikZ* megadás megegyezik: meg kell adni a kezdő- és a végpontot, amennyiben a *p5.js* esetében megadjuk előtte, hogy a pontok a szemközti sarkakat jelölik a kezdőpont, a magasság és szélesség helyett. Erre szolgál a *rectMode(CORNERS);* metódus és paramétere.

4.3.1.7. Grid osztály

A *Grid* osztály egy rácsháló létrehozását teszi lehetővé. A példányosításhoz meg kell adni két koordinátapárt (egy x és egy y koordinátát), valamint a tulajdonságokat.

A *p5.js* nem rendelkezik rácsháló kirajzolásához beépített függvénnyel, így saját megoldással történik a kirajzolás: a kezdőpont és a végpont között függőlegesen és vízszintesen is vonalak kerülnek kirajzolásra így megkapva a hálót.

A *TikZ* alapból rendelkezik ezzel a funkcióval, ebben az esetben csak a két pont megadása szükséges.

A rácsháló kirajzolását végző kódrészlet:

```
for (let i = Math.min(this.dimension.start.x, this.dimension.end.x);
     i <= Math.max(this.dimension.start.x, this.dimension.end.x);
     i += grid_density) {
         P5.line(i, this.dimension.start.y, i, this.dimension.end.y)
}

for (let i = Math.min(this.dimension.start.y, this.dimension.end.y);
    i <= Math.max(this.dimension.start.y, this.dimension.end.y);
    i += grid_density) {
        P5.line(this.dimension.start.x, i, this.dimension.end.x, i)
}
```

4.3.1.8. Arc osztály

Az *Arc* osztály egy körív rajzolását teszi lehetővé. A példányosításhoz meg kell adni két koordinátapárt (egy x és egy y koordinátát), valamint a tulajdonságokat. A tulajdonságok között szerepel az ív kezdő és végpontja fokban az adott sugarú körön.

A kezdőpont és a sugár meghatározásához itt is a *Ellipse* osztálynál megismert *getCenterDiameter* metódus szolgál.

A *p5.js* és a *TikZ* egyaránt rendelkezik beépített megoldással körív rajzolására, azonban máshogy kerülnek kirajzolásra. A *p5.js* esetében a középpont és az azon lévő köríven a megadott fokok, míg a *Tikz* esetében a megadott pont a kezdőpont, és onnan kerül a körív kirajzolásra, nem a középpont a mérvadó. Az exportált *TikZ* a *p5.js* megoldásával egyezik meg, vagyis a kezdőpontba el van tolva a megfelelő irányba.

4.3.1.9. Parabola osztály

A *Parabola* osztályban vannak definiálva egy parabola kirajzolásához szükséges metódusok. A példányosításhoz meg kell adni két koordinátapárt (egy x és egy y koordinátát), valamint a tulajdonságokat.

A *p5.js* nem rendelkezik parabola kirajzolásához beépített függvénnyel, így saját megoldással történik a *TikZ*-ben szereplő parabola kirajzolása.

A *TikZ* parabola egyenlete:

$$y = \frac{y_1 - y_0}{(x_1 - x_0)^2}(x - x_0)^2 + y_0, \qquad x \in [x_0, y_0]$$

A kódrészlet, amely a vászonra rajzolja ennek az egyenletnek a képét:

```
let a = (this.dimension.end.y - this.dimension.start.y) /
    Math.pow(this.dimension.end.x - this.dimension.start.x, 2)

if (isFinite(a)) {
  P5.beginShape();
```

```
  for (let i = Math.min(this.dimension.start.x, this.dimension.end.x);
      i <= Math.max(this.dimension.start.x, this.dimension.end.x);
      i += 0.25) {
        P5.vertex(i, a * (Math.pow(i - this.dimension.start.x, 2)) +
            this.dimension.start.y)
  }
  P5.endShape();
} else {
  P5.line(this.dimension.start.x, this.dimension.start.y,
      this.dimension.end.x, this.dimension.end.y)
}
```

4.3.1.10. LMath osztály

A *LMath* osztály matematikai kifejezések létrehozását teszi lehetővé. Példányosítás során meg kell adni a pont két koordinátáját, és a tulajdonságokat, mely között szerepel a kirajzolandó kifejezés.

A *p5.js* nem támogatja a LaTeX kifejezések megjelenítését a vásznon, így egy másik függvénykönyvtár használatával kellett megoldani: ez pedig a KaTeX[6], mely egy gyors, könnyen használható JavaScript könyvtár a LaTeX matematikai kifejezések webes megjelenítéséhez.

A *renderToCanvas* függvénnyel lehet a KaTeX által generált kifejezéseket a vászonra rakni, paraméterként meg kell adni a kirajzolandó kifejezést, magát a vásznat, mint kirajzolási hely, valamint a kifejezés pozícióját (az x és y koordinátát egyaránt).

```
let canvas = document.getElementById(P5.canvas.id);
katex.renderToCanvas(this.latex, canvas,
    this.x + grid_density / 8, this.y - grid_density / 2.5, {
      fontSize: 19.5
});
```

A *TikZ* megoldás egyszerűbb, ott egy *node* parancs paramétereként kell megadni a kifejezést matematikai módban.

4.3.1.11. Text osztály

A *Text* osztály szöveg vászonra helyezését teszi levetővé. Példányosítás ugyanúgy zajlik, mint a matematikai kifejezések esetében.

Annak ellenére, hogy a *p5.js* rendelkezik olyan funkciókkal ezek "kirajzolása" szintén a KaTeX függvénykönyvtár segítségével történik, annyi eltéréssel, hogy a kirajzolandó szöveg megkapja a \text{} paramétert, hogy ne érvényesüljön a matematikai mód.

```
let canvas = document.getElementById(P5.canvas.id);
katex.renderToCanvas(`\\text{${this.text}}`, canvas,
    this.x + grid_density / 8, this.y - grid_density / 2.5, {
      fontSize: 19.5
});
```

A *TikZ* megoldás hasonlóan a *node* parancs paramétereként kell megadni a szöveget.

4.3.2. Funkciók osztályai

Itt a két funkció kerül bemutatásra, ami osztályba lett szedve a könnyebb elérés miatt. A funkciók jelentős része modulként szerepel az alkalmazásban, nem osztályokba csoportosítva.

4.3.2.1. Control osztály

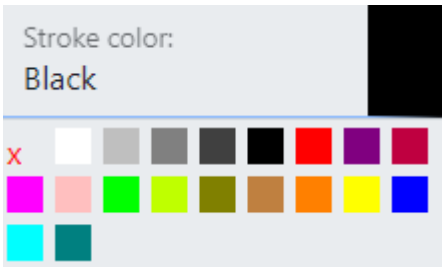
Ez az osztály felelős a vászon mozgatásáért és a nagyításért. Az osztályban eseménykezelők kerültek létrehozásra: kezelésre kerül az egér kattintása, húzása, felengedése valamint a görgetés. A vászon mozgatása szintén a "drag and drop" elvet követi.

Az eseménykezelők beállítása a *p5.js* példányára, amely a *sketch*-ben kerül létrehozásra:

```
P5.mousePressed = e => Control.mousePressed(e)
P5.mouseDragged = e => Control.mouseDragged(e);
P5.mouseReleased = e => Control.mouseReleased(e);
P5.mouseWheel = e => Control.zoomCanvas(e);
```

4.3.2.2. ColorPicker osztály

Ez az osztály kezeli az alkalmazásban szereplő színválasztó megfelelő működését. Itt kerül kezelésre az adott színválasztó kiválasztása, a színpaletta megjelenítése és eltüntetése, az adott szín kijelölése.



4.9. ábra. Példa a színek kiválasztására

A színválasztó előredefiniált színek alapján dolgozik, ezek megegyeznek a LaTeX által definiáltakkal:

```
const COLOR = {
  NONE: "#E9ECEF", // for no stroke or fill color
  WHITE: "#FFFFFF",
  LIGHTGRAY: "#BFBFBF",
  GRAY: "#808080",
  DARKGRAY: "#404040",
  BLACK: "#000000",
  RED: "#FF0000",
  VIOLET: "#800080",
  PURPLE: "#BF0040",
  MAGENTA: "#FF00FF",
  PINK: "#FFBFBF",
  GREEN: "#00FF00",
```

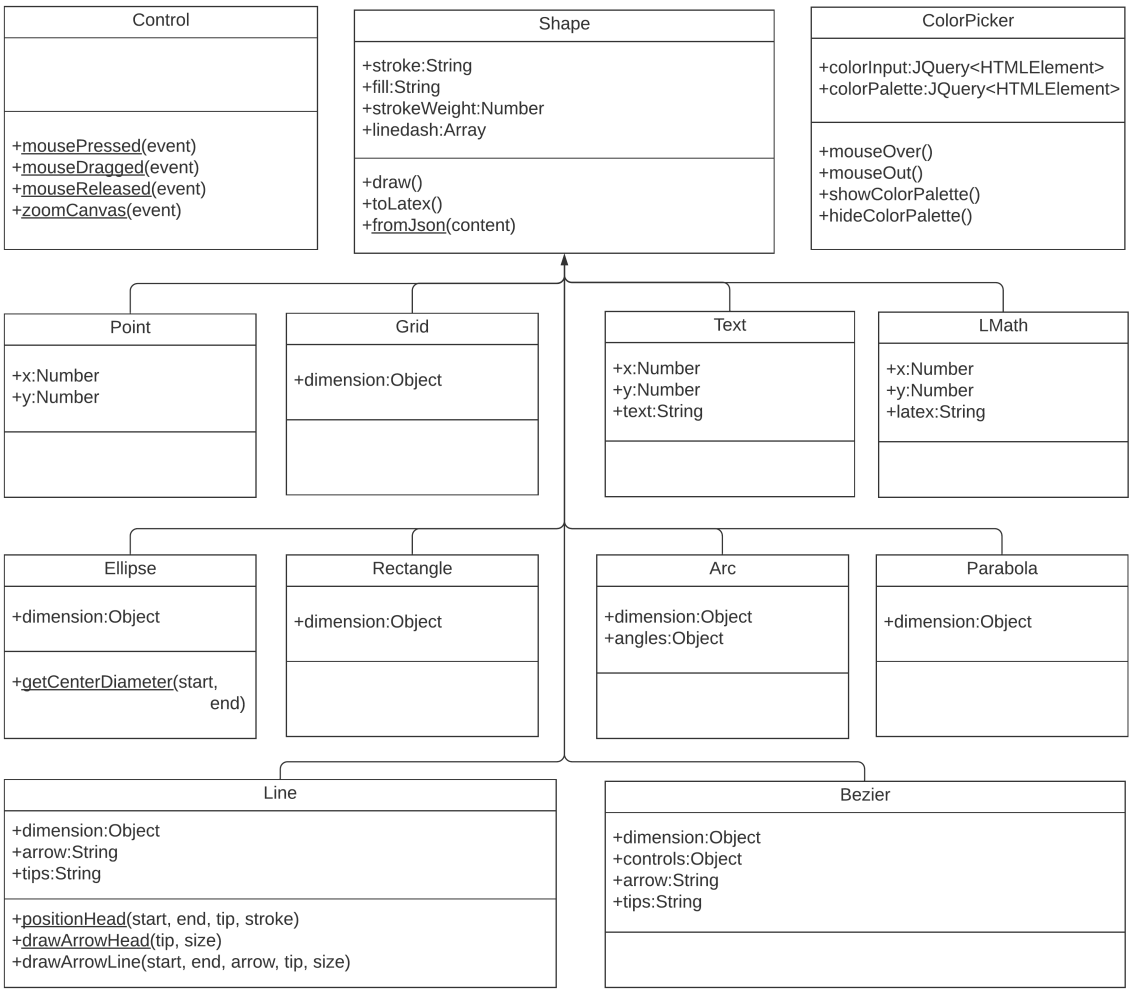
```
  LIME: "#BFFF00",
  OLIVE: "#808000",
  BROWN: "#BF8040",
  ORANGE: "#FF8000",
  YELLOW: "#FFFF00",
  BLUE: "#0000FF",
  CYAN: "#00FFFF",
  TEAL: "#008080"
}
```

Azért ilyen formában kerülnek tárolásra a színkódok, mert a *p5.js* tudja kezelni a hexadecimális színkódokat. A *COLOR*-ra hivatkozva meg tudjuk adni a színeket. Például piros körvonal, és világosszürke kitöltés esetében:

```
P5.stroke(COLOR.RED);
P5.fill(COLOR.LIGHTGRAY);
```

A *TikZ* ábrák esetében a hexadecimális kódokhoz visszakereshetőek a kulcsok is:

```
const getKey = (keyof, value) => Object.keys(keyof).find(key =>
                JSON.stringify(keyof[key]) === JSON.stringify(value));
```



4.10. ábra. Az osztályok UML diagrammja

5. fejezet

Példák ábrák szerkesztésére

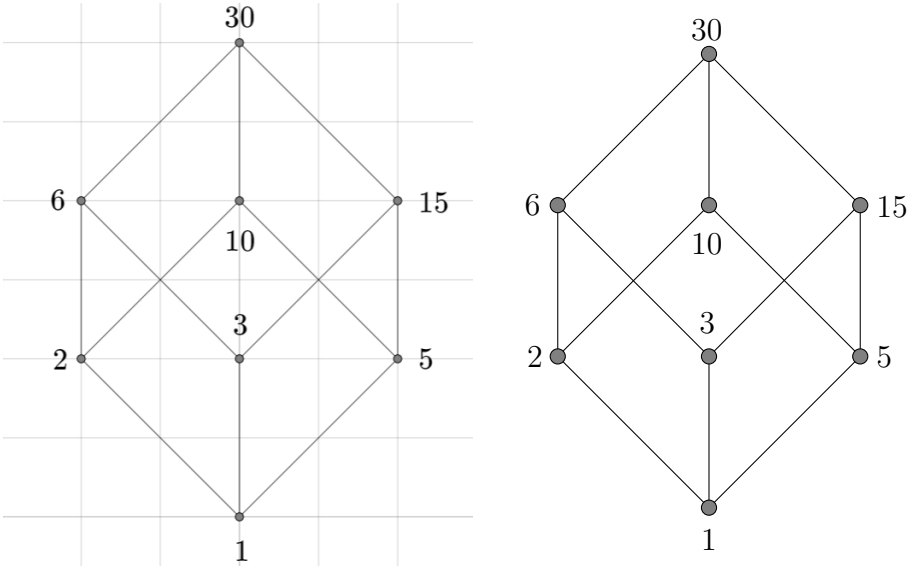
Ebben a fejezetben már az szerkesztőprogrammal elkészített ábrák kerülnek bemutatásra.

5.1. Hasse-diagram

A rendezettségelméletben a Hasse-diagram egy olyan matematikai diagramtípus, amelyet egy véges, részben rendezett halmaz ábrázolására használnak, annak tranzitív redukciójának rajza formájában. Konkrétan, egy részlegesen rendezett $(S, \leq)$ halmaz esetében az S minden elemét egy csúcsként ábrázoljuk a síkban, és egy olyan vonalszakaszt vagy görbét rajzolunk, amely x-ből y felé halad, amikor y lefedi x-et (vagyis amikor $x \leq y$ és nincs olyan z, hogy $x \leq z \leq y$). Ezek a görbék keresztezhetik egymást, de a végpontjaikon kívül nem érinthetnek más csúcsot. Egy ilyen diagram, felcímkézett csúcsokkal, egyértelműen meghatározza a részleges rendet.

Az oszthatóság egy részben rendezett halmaz. Vegyük a 30-as szám osztóit, ebben az esetben az alábbi halmaz adja meg:

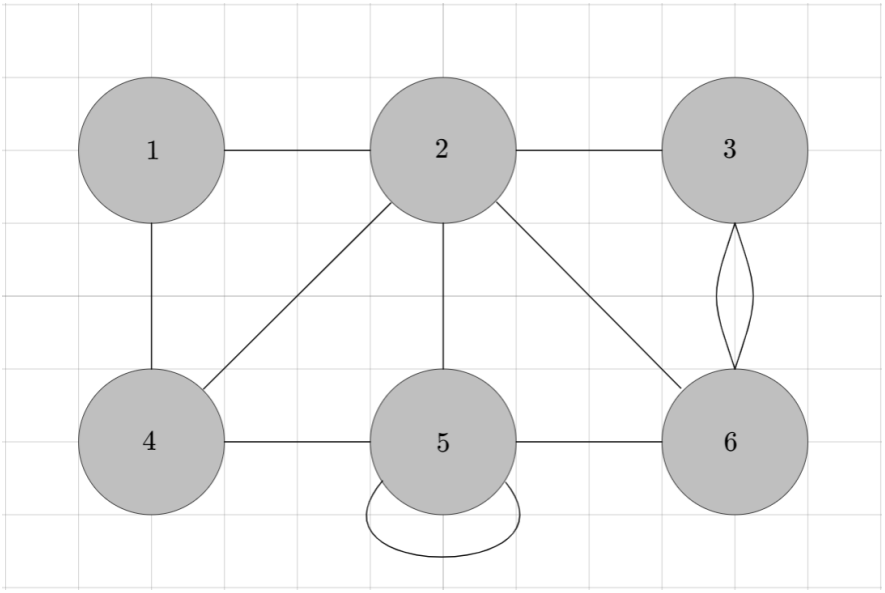
$$D_{30} = \{1, 2, 3, 5, 6, 10, 15, 30\}.$$



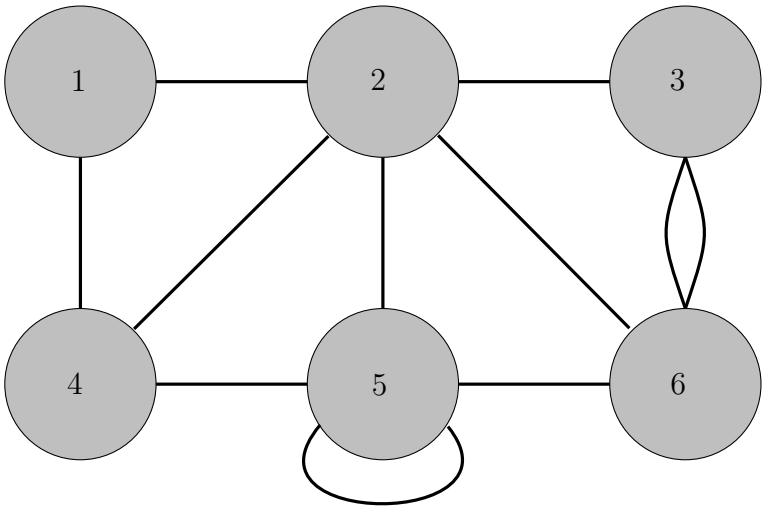
5.1. ábra. Egy Hasse-diagram az alkalmazásban és kimentve

5.2. Általános gráf

A gráf (vagy irányítatlan gráf) egy $G = (V, E)$ pár, ahol V egy olyan halmaz, amelynek elemeit csúcsoknak, E pedig a (rendezett) csúcspárok halmaza, amelynek elemeit éleknek nevezzük. Néha a gráfok tartalmazhatnak hurkokat, azaz olyan éleket, amelyek egy csúcsot önmagához kapcsolnak. Az ilyen általánosított gráfokat hurokkal rendelkező gráfoknak vagy egyszerűen gráfoknak nevezzük, ha a kontextusból egyértelmű, hogy a hurok megengedett.



5.2. ábra. Egy gráf képe az alkalmazásban

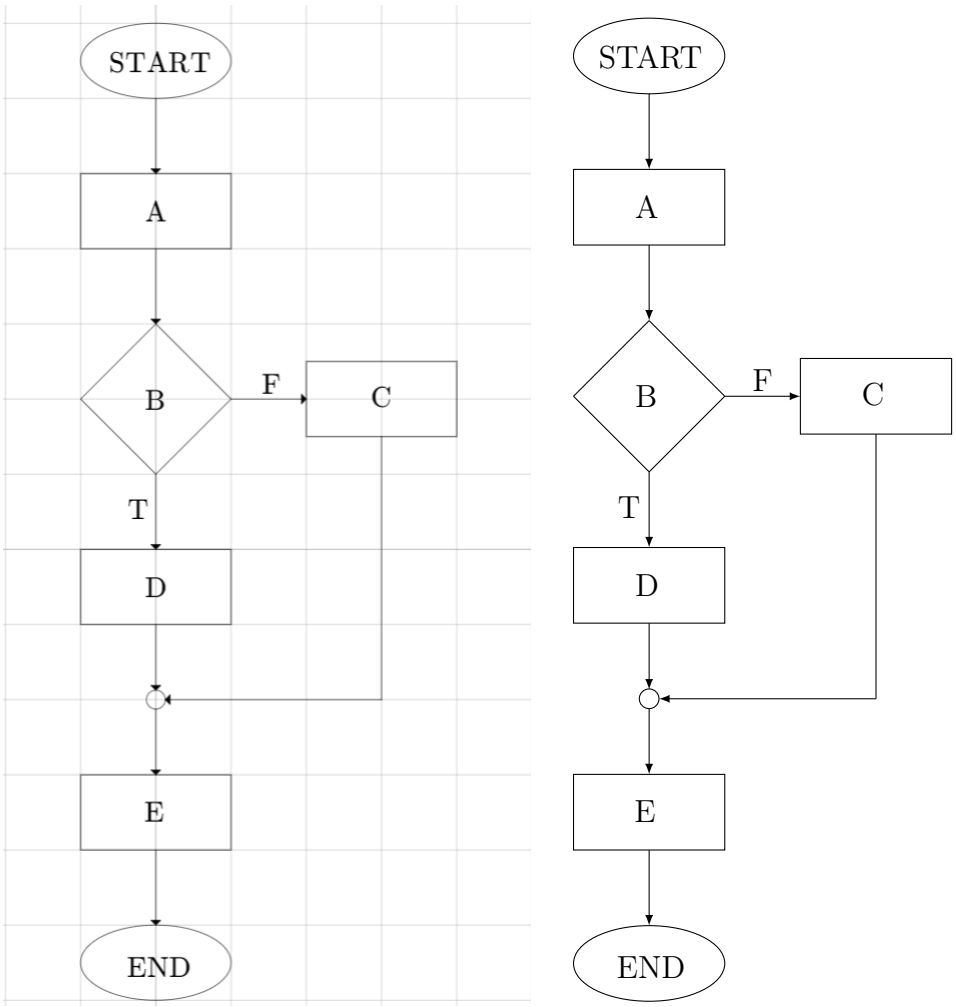


5.3. ábra. Az alkalmazásból kimentett gráf képe

5.3. Folyamatábra

A folyamatábra egy olyan diagramtípus, amely egy adott folyamatot vagy eljárást ábrázol. A folyamatábra definiálható úgy is, mint egy algoritmus, egy feladat megoldásának lépésről lépésre történő megközelítése.

Az folyamatábra a lépéseket különböző típusú dobozokként, a lépések sorrendjét pedig a dobozok nyilakkal való összekapcsolásával mutatja be. Ez a diagrammszerű ábrázolás egy adott probléma megoldási modelljét szemlélteti. A folyamatábrákat különböző területeken használják egy folyamat vagy program elemzése, tervezése, dokumentálása vagy kezelése során.



5.4. ábra. Egy folyamatábra képe az alkalmazásban és kimentve

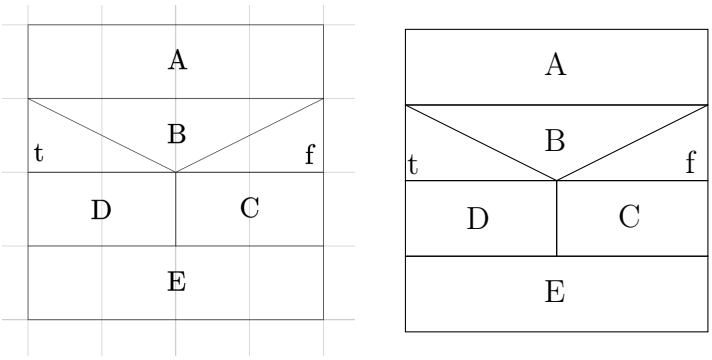
5.4. Nassi–Shneiderman-diagram

A Nassi-Shneiderman-diagram a számítógépes programozásban a strukturált programozás grafikus ábrázolása. Isaak Nassi és Ben Shneidermann fejlesztette ki 1972-ben. Ezeket az ábrákat struktogramoknak is nevezik, mivel a program struktúráit mutatják.

A struktogramok a folyamatábra szabadosságát korlátozzák és csak struktúrált szimbólumokat tartalmaznak. Alapelem a téglalap, amit különböző módokon osztunk

fel részekre. Maga a program egyetlen partícionált (felosztott) téglalap.

A Nassi-Shneiderman-diagramokat csak ritkán használják a formális programozásban. Absztrakciós szintjük közel áll a strukturált programkódhoz, és a módosítások miatt a teljes diagramot újra kell rajzolni. Az algoritmusokat és a magas szintű terveket egyértelművé teszik, ami az oktatásban is hasznossá teszi őket.



5.5. ábra. Egy struktogram képe az alkalmazásban és kimentve

5.5. Aranymetszés

A matematikában két mennyiség aranymetszésben van, ha arányuk megegyezik összegük és a két mennyiség közül a nagyobbik arányával. Algebrailag kifejezve, az a és a b esetében, ha $a > b > 0$, akkor

$$\frac{a+b}{a} = \frac{a}{b} = \varphi$$

ahol a φ az aranymetszést jelenti. Ez egy olyan irracionális szám, amely a $x^2 - x - 1 = 0$ kvadratikus egyenlet megoldása, amelynek értéke

$$\varphi = \frac{1+\sqrt{5}}{2} = 1.618033988749\ldots$$

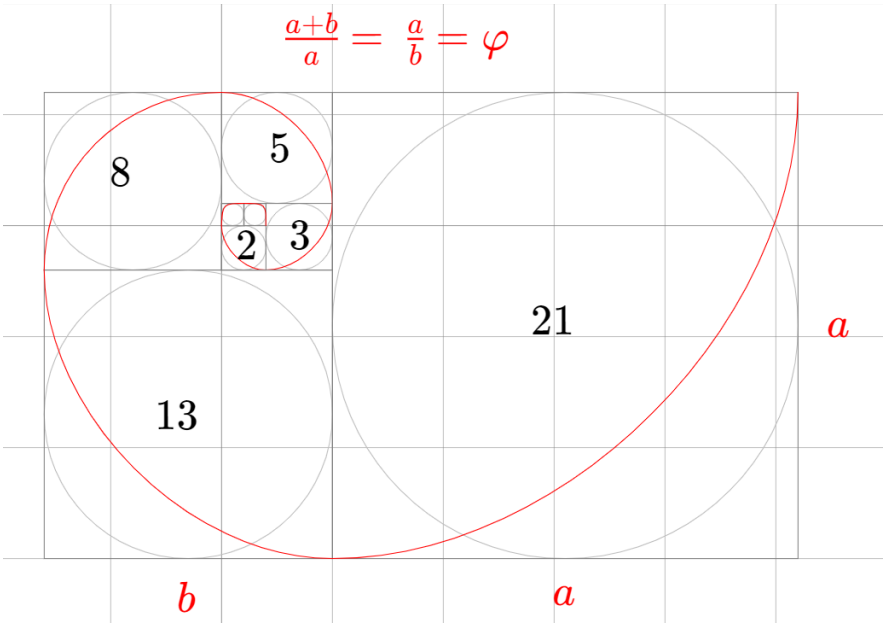
Az aranymetszés és a Fibonacci-sorozat matematikája szorosan összefügg egymással. A Fibonacci-sorozat:

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, \ldots$$

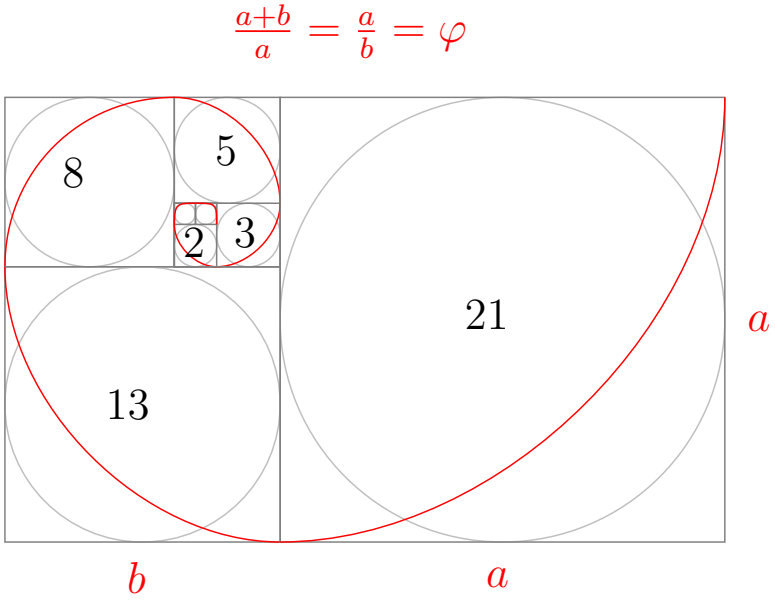
A Fibonacci-sorozat kifejezése az aranymetszést foglalja magában:

$$F(n) = \frac{\varphi^n - (1-\varphi)^n}{\sqrt{5}} = \frac{\varphi^n - (-\varphi)^{-n}}{\sqrt{5}}$$

Az aranymetszés alkalmazásával a webfejlesztésben a weloldal vonzóvá válik a felhasználó számára, és az elrendezése tudat alatt tökéletesnek és ismerősnek tűnik számukra. Ezáltal sokkal valószínűbbé válik, hogy továbbolvassák és alaposan megnézik a webhelyet. Az aranymetszést a webtervezésben arra is használják, hogy kiegyensúlyozzák a szöveges tartalmakat, hierarchiát hozzanak létre, és a felhasználók szemét bizonyos területekre irányítsák.



5.6. ábra. Az aranymetszés képe az alkalmazásban



5.7. ábra. Az alkalmazásból kimentett aranymetszés képe

6. fejezet

Összefoglalás

A szakdolgozatom egy olyan online grafikus szerkesztő létrehozása volt, amellyel LaTeX-be visszailleszthető kód hozható létre.

A dolgozat elkészítése közben alaposabban megismerhettem a Bootstrap 5[1] eszköztárral, a KaTeX[6], és a p5.js[14] függvénykönyvtárak működését, előnyeit, hátrányait, és hiányosságait. Ha újra abban a helyzetben lennék, hogy válogatni kellene a könyvtárak közül, biztos vagyok benne, hogy ugyanúgy ezeket választanám. A különböző könyvtárak dokumentációja az online felületükön is elérhetők, interaktív példákkal illusztrálva a működésüket.

Véleményem szerint sikerült egy olyan webes alkalmazást létrehoznom, amely beleillik a mai modern weboldalakba. A célom az volt, hogy egy egyszerű, könnyen használható, azonban mégis funkciókban gazdag alkalmazás készüljön el. Az alkalmazás jelen formájában kiszolgálja egy átlag felhasználó igényeit: tud másolni, beilleszteni és törölni, lehet pontokat mozgatni, valamint meglévő ábrák tulajdonságait módosítani. A már kész ábrák visszatölthetők későbbi szerkesztésre.

A szoftverfejlesztésben még mindig rengeteg lehetőség és esély rejlik. Az alkalmazásom esetében az egyik, amit kiemelnék ezek közül a LaTeX forráskód betöltése, ugyanis jelenleg minden elkészült ábrához tartozik egy kódolt karaktersorozat, amely tartalmazza az információkat, melyből az adott alakzat visszatölthető. Jelenleg a pontok mozgatása kissé körülményes tud lenni nagyobb és részletesebb ábrák esetében, illetve egy teljes alakzat mozgatásához az összes pontját ki kell jelölni. A pontok mozgatásánál jelenleg minden lerakott ábra összes pontja megjelenik, a jövőben célszerű lenne először egy kijelölésre leszűkítve megjeleníteni ezeket. Az alakzatok mozgatása esetében pedig célravezető, ha egy pont kijelölésével az elem összes pontja kiválasztásra kerül, amennyiben például *ALT* billentyű lenyomása közben történik.

Abban egészen biztos vagyok, hogy a szakdolgozat elkészítése során szerzett rengeteg hasznos ismeretet alkalmazni tudom majd a jövőben is.

Irodalomjegyzék

[1] Bootstrap 5. https://getbootstrap.com/docs/5.1/getting-started/introduction/. [2021. 10. 02.].

[2] William Slough Andrew Mertz. Graphics with tikz. http://mirror.tug.org/pracjourn/2007-1/mertz/mertz.pdf. [2021. 10. 06.].

[3] MDN Web Docs. Css/@media. https://developer.mozilla.org/en-US/docs/Web/CSS/@media. [2021. 10. 02.].

[4] MDN Web Docs. Html5. https://developer.mozilla.org/en-US/docs/Glossary/HTML5. [2021. 10. 02.].

[5] MDN Web Docs. Javascript. https://developer.mozilla.org/en-US/docs/Web/JavaScript. [2021. 10. 02.].

[6] Sophie Alpert Emily Eisenberg. Katex. https://katex.org/. [2021. 10. 05.].

[7] John D. Hobby. Metapost - a user's manual. https://www.tug.org/docs/metapost/mpman.pdf. [2021. 11. 13.].

[8] Ecma International. Ecmascript® 2015 language specification. https://262.ecma-international.org/6.0/. [2021. 10. 05.].

[9] Ecma International. Ecmascript® 2021 language specification. https://www.ecma-international.org/publications-and-standards/standards/ecma-262/. [2021. 10. 05.].

[10] Ecma International. The json data interchange syntax. https://www.ecma-international.org/publications-and-standards/standards/ecma-404/. [2021. 10. 05.].

[11] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley Professional, 1994.

[12] JGraph Ltd. https://app.diagrams.net/. [2021. 10. 07.].

[13] The LaTeX Project. https://www.latex-project.org/. [2021. 10. 02.].

[14] Evelyn Masso Qianqian Ye. p5.js. https://p5js.org/. [2021. 10. 24.].

[15] Yichuan Shen. tikzcd-editor. https://tikzcd.yichuanshen.de/. [2021. 10. 07.].

[16] Daniel Shiffman. Coding train. https://thecodingtrain.com/. [2021. 11. 07.].

[17] Daniel Shiffman. *Nature of Code: Simulating Natural Systems with Processing.* 2012.

[18] Till Tantau. Tikz and pgf 1.18. `https://www.bu.edu/math/files/2013/08/tikzpgfmanual.pdf`. [2021. 10. 03.].

[19] Till Tantau. Tikz and pgf 3.1.9a. `http://mirrors.ctan.org/graphics/pgf/base/doc/pgfmanual.pdf`. [2021. 10. 05.].

[20] TikzEdt. `http://www.tikzedt.org/`. [2021. 10. 07.].

[21] TikZiT. `https://tikzit.github.io/`. [2021. 10. 07.].

[22] Timothy Van Zandt. Pstricks: Postscript macros for generic tex. `http://mirror.ctan.org/graphics/pstricks/base/doc/pstricks-doc.pdf`. [2021. 11. 13.].

A mellékelt CD tartalma

A dolgozathoz tartozó melléklet a következőket tartalmazza:

- `thesis.pdf`: a dolgozat PDF formátumban
- `manual.pdf`: a használati útmutató PDF formátumban
- `thesis/`: a dolgozat LaTeX forráskódját tartalmazó jegyzék
- `editor/`: az elkészített szerkesztő jegyzéke

A szerkesztő futtatásához egy webszerver szükséges. Amennyiben nem áll rendelkezésre akkor egy *Node.js* telepítés a legegyszerűbb megoldás:

Letölthető az alábbi linken:
`https://nodejs.org/en/download/`

A *http-server* telepítése:

```
> npm install --global http-server
```

Az alkalmazás elindításához ezenkívül egy parancssorra lesz szükség:

```
> http-server ./editor/
```

A parancs kiadása után már el is érhető al alkalmazás, jelen esetben a *8080*-as port alatt:

```
> http-server ./editor/
Starting up http-server, serving ./editor/

http-server version: 14.0.0

http-server settings:
CORS: disabled
Cache: 3600 seconds
Connection Timeout: 120 seconds
Directory Listings: visible
AutoIndex: visible
Serve GZIP Files: false
Serve Brotli Files: false
Default File Extension: none

Available on:
http://192.168.0.2:8080
http://127.0.0.1:8080

Hit CTRL-C to stop the server
```