Szakdolgozat

Miskolci Egyetem

Webalapú SVG szerkesztő alkalmazás

Készítette:

Czibik Lóránt Patrik

Programtervező informatikus

Témavezető:

Dr. Vadon Viktória

Miskolc, 2026

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

Szakdolgozat Feladat

Czibik Lóránt Patrik (K1IFAB) BSc Programtervező informatikus jelölt részére.

A szakdolgozat tárgyköre: Webfejlesztés, vektorgrafika, SVG

A szakdolgozat címe: Webalapú SVG szerkesztő alkalmazás

A feladat részletezése:

Az SVG (Scalable Vector Graphics) egy XML formátumú vektorgrafikus leíró nyelv, amely széles körben használatos illusztrációk, diagramok és ikonok készítésére. A formátum jellegzetessége, hogy a grafikus elemeket matematikai leírással tárolja, így az ábrák korlátlanul nagyíthatók minőségromlás nélkül.

A dolgozat célja egy olyan webalapú szerkesztőfelület tervezése és elkészítése, amely lehetővé teszi az SVG ábrák létrehozását és szerkesztését grafikus eszközökkel, a kimenet pedig szerkeszthető SVG fájlként vagy képként menthető el.

A dolgozat tartalmazza a hasonló célú alkalmazások funkcióinak áttekintését és összehasonlítását, a szerkesztőfelület specifikációját, valamint annak vizsgálatát, hogy a bemeneti SVG fájlok milyen esetekben és milyen korlátozásokkal módosíthatók a szerkesztőfelületen.

Témavezető: Dr. Vadon Viktória, egyetemi adjunktus

A feladat kiadásának ideje: 2025. 09. 18.

. . . . . .
szakfelelős

Eredetiségi Nyilatkozat

Alulírott **Czibik Lóránt Patrik**; Neptun-kód: `K1IFAB` a Miskolci Egyetem Gépészmérnöki és Informatikai Karának végzős Programtervező informatikus szakos hallgatója ezennel büntetőjogi és fegyelmi felelősségem tudatában nyilatkozom és aláírásommal igazolom, hogy a *Webalapú SVG szerkesztő alkalmazás* 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 Scalable Vector Graphics (SVG) [17] egy XML-alapú leírónyelv kétdimenziós vektorgrafikák készítéséhez, mely lehetővé teszi az interaktivitást és az animációt. Az internet korai napjaiban jött létre, mivel szükség volt egy szabványosított vektorgrafikus formátumra a weben. Az SVG háromféle grafikus objektumot támogat. Ezek a vektorgrafikus alakzatok, melyek egyenesekből és görbékből álló útvonalakat alkotnak. Emellett támogat képeket és szöveget is. A grafikus objektumok csoportosíthatók, stílusozhatók, átalakíthatók és összetett képekké egyesíthetők. Az SVG formátum különlegessége, hogy grafikai elemeit matematikai leírásként tárolja, így a képek tetszőleges mértékben nagyíthatók minőségromlás nélkül. Ezért széles körben használt ikonok, logók, diagramok készítésére, ahol a megjelenés élessége különösen fontos. Ezen tulajdonságok miatt a vektoros grafikát általában egyszerű, minimalista illusztrációkhoz használják. Ezzel szemben a rasztergrafika[3] a képet pixelek, képképpontok rácsos elrendezésével, színes fokozatokkal ábrázolja, általában a fényképek és részletes képek tárolására alkalmasabb. A komplex ábrák elkészítése SVG formátumban nem egyszerű feladat és a fájlméret is jelentősen megnövekedhet.

A szakdolgozatom célja egy webalapú SVG szerkesztő alkalmazás tervezése, dokumentálása és megvalósítása. A felület a WYSIWYG (*What You See Is What You Get*) elv alapján működik. Grafikus eszközökkel teszi lehetővé az ábrák létrehozását, bemeneti SVG fájlok módosítását és exportálását. Elmenthető a kimenet szerkeszthető SVG fájlként illetve raszteres formátumú képként. A bemeneti SVG fájlok esetén szükséges a fájlokat felülvizsgálni és optimalizálni, mivel tartalmazhatnak redundáns kódrészeket vagy külső szkripteket. Kiemelten fontos egy felhasználóbarát, intuitív és logikusan elrendezett felület kialakítása, mely alkalmas a gyakori használatra.

A webes szerkesztőfelület megvalósításához szükséges egy olyan keretrendszer, mely egyaránt biztosítja a hatékonyságot, a könnyű fejleszthetőséget és a jó teljesítményt. A cél az alap HTML, CSS és JavaScript bővítése komponensekkel és állapotkezeléssel. A komponensek a szerkesztőfelület egyes részét képezik, például az eszközsávot és gombjait. Az állapotkezelés feladata a komponensek közötti adatok megosztása, valamint az alkalmazás aktuális állapotának egységes kezelése.

A fenti szempontok alapján a Svelte[14][5] frontend keretrendszer bizonyult a legjobb választásnak. A hasonló keretrendszerektől eltérően nem futásidőben értelmezi a komponenseket, hanem már a fordítási folyamat során alakítja át őket JavaScript kóddá. A szintaxisa egyszerű, így a HTML, CSS és JavaScript ismeretekkel rendelkező fejlesztők számára könnyen elsajátítható. Továbbá a Typescript[10] technológia is használt, mely a JavaScript-et típusokkal bővíti, és gyakran használt Svelte projektekhez.

A következőkben megismerkedhetünk az SVG grafikus elemeivel(2). Ezt követően elemzésre kerülnek a hasonló célú webes alkalmazások főbb funkciói és megközelítései(2.1.5). Ezek összehasonlításából pedig követelményeket és tervezési irányelveket fogalmazok meg a saját szerkesztőmmel szemben (2.2.6). Az implementáció fejezetben bemutatom a felhasznált technológiákat, könyvtárakat, definiált osztályokat és a technológiai hátteret. Ezt követően részletezem az elkészült alkalmazás felépítését, funkcionalitását, és használatát. Végül néhány példával szemléltetem az alkalmazással előállítható illusztrációkat.

2. fejezet

Az SVG és eszközkészlete

2.1. Az SVG grafikus elemei

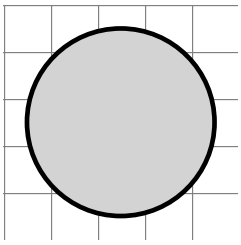
A Scalable Vector Graphics (SVG) egy XML-alapú nyelv, mely kétdimenziós vektorgrafikák leírására használatos. Hierarchikusan felépített, minden grafikus objektum egy közös gyökérelembe, az <svg> kezdő és záró címke közé ágyazódik. Ezt a szerkezetet szemlélteti az alábbi kódrészlet:

```
<svg xmlns="http://www.w3.org/2000/svg"
    width="100" height="100">
    <! - - SVG elemek - ->
</svg>
```

Az SVG koordináta-rendszerének origója (0, 0) a bal felső sarokban található. A hagyományos matematikai ábrázolástól eltérően itt az y tengely pozitív iránya lefelé mutat.

A `viewBox` attribútum A teljes skálázhatóság és a reszponzív viselkedés kulcsa az <svg> elem opcionális `viewBox` attribútuma. Ez a nézetablak meghatároz egy új, független koordináta-rendszert a tartalom számára, amely automatikusan skálázódik a megadott `width` és `height` méretekhez. Ennek hiányában a koordináták közvetlenül a képernyő pixeleiben értendők. A `viewBox` értéke négy szám: `min-x`, `min-y`, `width` és `height`. Az alábbi példában a `viewBox` egy 0-tól 100-ig terjedő koordináta-rendszert határoz meg, és az SVG tartalma teljesen kitölti a 400×400 pixeles tárolót.

```
<svg xmlns="http://www.w3.org/2000/svg"
    width="400" height="400" viewBox="0 0 100 100">
    <circle cx="50" cy="50" r="40"
        fill="lightgray" stroke="black" stroke-width="2"/>
</svg>
```



2.1. ábra. Példa: A viewBox attribútum hatása a koordinátarendszer skálázására

Ebben a fejezetben bemutatom az SVG gyakran használt elemeit, amiket a webalkalmazásban is implementálni fogok. Kezdve a megjelenési attribútumokkal(2.1), melyek a későbbiekben bemutatott elemek vizuális megjelenését befolyásolják. Folytatva az alapvető alakzatokkal(2.1.1), melyeket először az attribútumaik listázásával, majd

egy kódpéldával és ábrával demonstrálok. Ez után az SVG specifikáció különböző görbéi, az útvonalak(2.1.2) és további grafikus elemei(2.1.3) kerülnek bemutatásra, hasonló formátumban. Végül ezen elemek csoportosítását és újrafelhasználását(2.1.4) mutatom be két példán keresztül.

2.1.1. Megjelenési attribútumok

Az SVG elemek geometriai meghatározása mellett a kinézetüket a megjelenési attribútumok[6] határozzák meg. Ezek az attribútumok CSS tulajdonságként is értelmezhetők, így megadhatók közvetlenül az elemen attribútumként, vagy CSS stíluslapokon keresztül is.

Körvonal

A körvonal az alakzatok határvonalát jelenti. Az SVG szabvány számos attribútumot biztosít a vonalak végződésének, csatlakozásának és mintázatának beállítására.

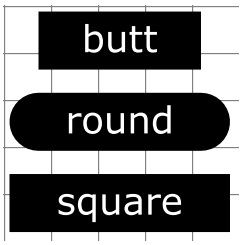
stroke: A körvonal színét határozza meg. Értéke lehet konkrét színkód, színnév, vagy hivatkozás egy színátmenetre. Alapértelmezett értéke none, vagyis a körvonal nem látható.

stroke-width: A körvonal vastagságát definiálja. Értéke egy pozitív szám, mely az aktuális koordináta-rendszer egységében értendő, alapértéke az 1.

stroke-opacity: A körvonal átlátszóságát szabályozza 0.0 (teljesen átlátszó) és 1.0 (teljesen látható) között. Ez független az elem többi részének átlátszóságától.

stroke-linecap: Nyílt útvonalak végpontjainak alakját határozza meg. Három lehetséges értéke van:

- `butt`: A vonal pontosan a végpontnál ér véget, levágott véggel. Ez az alapértelmezett érték.
- `round`: A vonal végét egy félkör zárja le, melynek sugara a vonalvastagság fele.
- `square`: A vonal vége egy négyzetes lezárást kap, mely a vonalvastagság felével túlnyúlik a végponton.

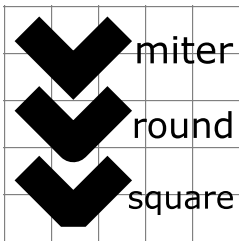


2.2. ábra. Példa: A stroke-linecap attribútum három lehetséges értéke.

stroke-linejoin: Két vonalszakasz találkozásánál, vagyis sarkánál kialakuló alakzat típusát adja meg. Három lehetséges értéke van:

- `miter`: Hegyes sarok, ahol a külső élek a metszéspontig meghosszabbodnak. Ez az alapértelmezett érték.

- `round`: A sarok lekerekített.
- `square`: Levágott sarok, mintha egy egyenes vonallal összekötnénk a két szakasz külső pontjait.



2.3. ábra. Példa: A stroke-linejoin attribútum három lehetséges értéke.

stroke-dasharray: Szaggatott vonalak létrehozására szolgál. Értéke egy vesszővel vagy szóközzel elválasztott számsorozat, mely felváltva határozza meg a vonalszakaszok és a szünetek hosszát.

Kitöltés

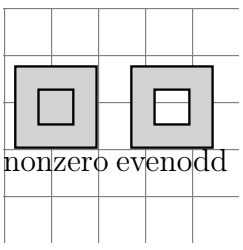
A kitöltés az alakzatok belső területének színezését jelenti. Zárt alakzatoknál, mint például köröknél, egyértelmű a belső terület. Viszont nyílt útvonalaknál a végpontok összekötésével képzett területet jelöli.

fill: A belső terület színét határozza meg. Hasonlóan a stroke-hoz, lehet szín, átmenet vagy minta. Alapértelmezett értéke a fekete. A none érték használatával az alakzat belseje átlátszó marad.

fill-opacity: A kitöltés átlátszóságát állítja be, anélkül, hogy a körvonal átlátszóságát befolyásolná.

fill-rule: Önmagát metsző vagy lyukas útvonalak esetén határozza meg, hogy mely területek számítanak belülnek és melyek kívülnek.

- `nonzero`: Ez az alapértelmezett szabály. Egy adott pontból húzott sugár és az útvonal metszéspontjainak irányát vizsgálja. Ez lehet óramutató járásával megegyező vagy ellentétes. Ha az összeg nem nulla, a pont belül van.
- `evenodd`: Azt vizsgálja, hogy a sugár hányszor metszi az útvonalat. Ha a metszéspontok száma páratlan, a pont belül van. Ha ez a szám páros, akkor kívül van. Lehetővé teszi a lyukas alakzatok egyszerű létrehozását.



2.4. ábra. Példa: A fill-rule attribútum két lehetséges értéke.

Láthatóság

Az `opacity` attribútum az elem átlátszóságát szabályozza. Értéke a 0.0 és 1.0 közötti tartományban mozoghat, ahol a nulla a teljes átlátszóságot, az egy pedig a teljes láthatóságot jelöli. Fontos különbség a kitöltési és körvonal átlátszósághoz képest, hogy csoportokra alkalmazva az attribútum a benne foglalt elemek összesített átlátszóságát határozza meg, így a megjelenítéskor az elemek egységesen halványodnak el, nem pedig egymáson áttűnve.

Transzformációk

A `transform` attribútum segítségével végezhető el az elemek geometriai manipulációja az eredeti koordináta-rendszerhez viszonyítva. A `translate(x, y)` parancs az eltolást végzi az x és y tengelyek mentén. A `rotate(a, x, y)` adott `a` szöggel és egy tetszőleges `x, y` középpont körül forgatja el az elemet. A `scale(x, y)` az elem átméretezését teszi lehetővé az `x, y` tengelyeken. Egy paraméter használatával, tehát `scale(s)` esetén `s` mennyiséggel egységesen skálázza az elemet.

Szűrők

A `filter` attribútum használata komplex vizuális effektek, például elmosás, árnyékvetés vagy színkorrekció alkalmazását teszi lehetővé. Az attribútum értéke jellemzően a definíciós szekcióban létrehozott szűrőelem egyedi azonosítójára hivatkozik, melyet az URL szintaxis segítségével kapcsolunk a grafikus elemhez.

Vágás

A `clip-path` egy specifikus SVG mechanizmus, mely a látható terület korlátozására szolgál. Egy SVG alakzat, például egy kör vagy vonal segítségével vágja le a megjelenítendő elemet. A definiált vágási területen kívül eső részek a megjelenítéskor nem láthatók.

Maszkolás

A maszkolás a vágáshoz hasonló technika, azonban itt a maszkoláshoz használt elem szürkeárnyalatos értékei határozzák meg az átlátszóság mértékét. A `mask` attribútummal használható. A fehér szín a teljes láthatóságot, a fekete a teljes átlátszóságot, a köztes szürke árnyalatok pedig félig áttetsző megjelenést eredményez.

2.1.2. Alapvető alakzatok

Az SVG szabvány számos előre definiált alakzatot[7] támogat, melyek segítségével egyszerűen hozhatunk létre grafikai elemeket.

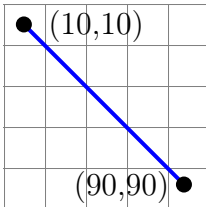
Vonalak

A vonalakat a `<line>` elem segítségével ábrázoljuk. A vonal meghatározásához négy alapvető attribútumot szükséges megadni, melyek a kezdő- és végpont koordinátái.

- `x1, y1`: A kezdőpont koordinátái

- `x2, y2`: A végpont koordinátái

```
<line x1="10" y1="10" x2="90" y2="90"
    stroke="blue" stroke-width="2"/>
```



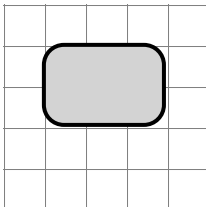
2.5. ábra. Példa: Vonal elem kódja és vizuális megjelenése

Téglalapok

A téglalapok a `<rect>` elemmel adhatók meg. Az alakzat pozícióját a bal felső sarok koordinátái, méretét pedig a szélesség és a magasság attribútumai határozzák meg. Lehetőség van továbbá a sarkok lekerekítésére is.

- `x, y`: A bal felső sarok koordinátái
- `width, height`: A bal felső sarok pontjától számított egység az x és y tengelyen
- `rx, ry`: A sarkok negyed ellipszisének horizontális és vertikális sugara

```
<rect x="20" y="20" width="60" height="40"
    rx="10" ry="10" fill="lightgray"
    stroke="black" stroke-width="1"/>
```



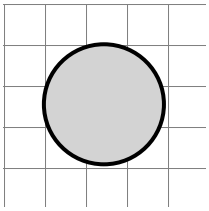
2.6. ábra. Példa: Téglalap elem, opcionálisan lekerekített sarkakkal, kód és vizuális megjelenés.

Körök

A körök a `<circle>` elemmel ábrázolhatók. Középpontjának koordinátáival és sugarával adható meg.

- `cx, cy`: A kör középpontjának koordinátái
- `r`: A kör sugara

```
<circle cx="50" cy="50" r="30"
    fill="lightgray" stroke="black"
    stroke-width="2"/>
```



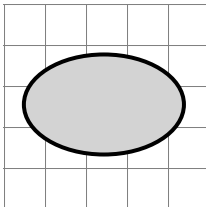
2.7. ábra. Példa: Kör elem kódja és vizuális megjelenése

Ellipszisek

Az ellipszisek az <ellipse> elemmel adhatók meg. A körhöz hasonlóan működnek, de külön x és y sugarakkal.

- `cx, cy`: Az ellipszis középpontjának koordinátái
- `r`: Az ellipszis sugara
- `rx, ry`: Az ellipszis horizontális és vertikális sugara

```
<ellipse cx="50" cy="50" rx="40" ry="25"
    fill="lightgray" stroke="black"
    stroke-width="2"/>
```

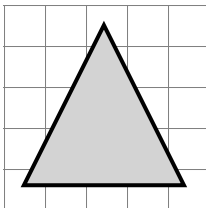


2.8. ábra. Példa: Ellipszis elem kódja és vizuális megjelenése

Poligonok

A <polygon> elem egy tetszőleges számú éllel rendelkező, zárt alakzatot definiál. A poligon leírásához a `points` attribútum szükséges. A bejárás sorrendjében, szóközzel elválasztva szükséges leírni az x,y koordináta páronkat. A leírás végeztével automatikusan összeköti a kezdő- és a végpontot, nem szükséges ismételten megadni a kiinduló pontot.

```
<polygon points="50,10 90,90 10,90"
    fill="lightgray" stroke="black"
    stroke-width="2"/>
```

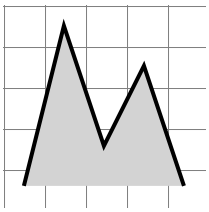


2.9. ábra. Példa: Poligon elem kódja és vizuális megjelenése

Polivonalak

A polivonalak, vagyis töröttvonalak a <polyline> elemmel definiálhatók. Hasonlóan a poligonokhoz csak a `points` attribútum szükséges a leírásához. Az eltérés csak az, hogy nem zárt alakzat.

```
<polyline points="10 90, 30 10, 50 70, 70 30, 90 90"
    fill="lightgray" stroke="black"
    stroke-width="2" />
```



2.10. ábra. Példa: Polivonal elem kódja és vizuális megjelenése

2.1.3. Útvonalak

Az SVG `<path>` eleme a legrugalmasabb és legösszetettebb módja a vektoros alakzatok létrehozásának. Az útvonalak a `d` attribútum által definiált parancsok sorozatával írják le az alakzatot. Minden parancs egy betűből és a hozzá tartozó numerikus paraméterekből áll. A betű a parancs típusát jelöli. A parancsok abszolút és relatív kategóriákba sorolhatók. Az abszolút parancsok nagybetűvel jelöltek, a koordináta-rendszer origójához viszonyítanak. A relatív parancsok kisbetűsek, az aktuális ponttól számított elmozdulást jelentik. A numerikus paraméterek lehetnek koordinátapárok, vagy például az ívek (2.1.3) esetén jelenthetnek elforgási szöget, logikai paramétereket.

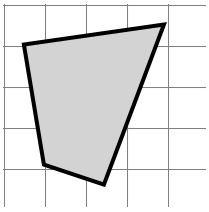
Egyenes vonalak

Az egyenes vonalak rajzolásához több alapvető parancs is rendelkezésre áll.

- `M, m` (Move To): Az "ecset" megemelése és áthelyezése a megadott pontba anélkül, hogy rajzolna
- `L, l` (Line To): Egyenes vonal rajzolása az aktuális ponttól a megadott pontig
- `H, h` (Horizontal Line To): Vízszintes vonal rajzolása
- `V, v` (Vertical Line To): Függőleges vonal rajzolása
- `Z, z` (Close Path): Egyenes vonallal visszazár az útvonal kezdőpontjába

Az alábbi útvonal egy sokszöget rajzol, mely a (20,20) pontban kezdődik, és a `Z` paranccsal zárul vissza a kezdőponthoz.

```
<path d="M 10,20 L 80,10 L 50,90 L 20,80 Z"
    fill="lightgray" stroke="black"
    stroke-width="2"/>
```



2.11. ábra. Példa: Tetszőleges sokszög rajzolása egyenes útvonalakkal, kód és megjelenés

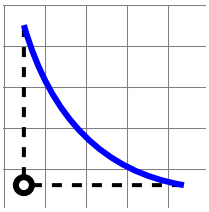
Bézier-görbék

A Bézier-görbék[4] matematikai görbék, melyek kontrollpontok segítségével definiálják a görbe alakját. Az SVG két típusú Bézier-görbét támogat.

Másodfokú Bézier-görbe

A másodfokú Bézier-görbe (Quadratic Bézier) a görbék legegyszerűbb változata. Egy kontrollpontja van, ami a görbe kezdetének és a végének az irányát befolyásolja. A `Q, q` paranccsal rajzolható.

```
<path d="M 90,90 Q 30,80 10,10"
    fill="none" stroke="blue"
    stroke-width="3"/>
```

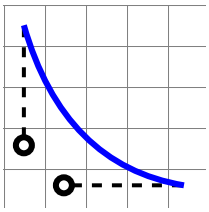


2.12. ábra. Példa: Másodfokú Bézier-görbe és kontrollpont, kód és megjelenés

Harmadfokú Bézier-görbe

A harmadfokú Bézier-görbe (Cubic Bézier) két kontrollponttal rendelkezik. Az első a görbe kezdeténél, a második a görbe végénél határozza meg az irányt. A `C, c` paranccsal rajzolható.

```
<path d="M 90,90 C 50,83 23,57 10,10"
    fill="none" stroke="blue"
    stroke-width="3"/>
```



2.13. ábra. Példa: Harmadfokú Bézier-görbe és a két kontrollpontja, kód és megjelenés

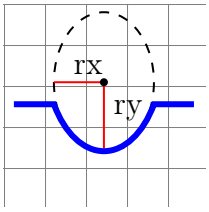
Ív

Az `A, a` (Arc) parancs elliptikus ívek rajzolására szolgál. Használata előtt szükséges egy adott kezdőpont, vagy annak meghatározása az `M, m` paranccsal. Ez az egyik legbonyolultabb útvonalparancs, mivel számos paramétert igényel:

- `rx, ry`: Az ellipszis x és y irányú sugara
- `x-axis-rotation`: Az ellipszis x tengelyének elforgatási szöge
- `large-arc-flag`: Meghatározza, hogy a rövidebb (0, alapérték) vagy hosszabb (1) ívet rajzolja-e
- `sweep-flag`: Meghatározza, hogy pozitív (1, alapérték) vagy negatív (0) irányban rajzolódik-e az ív
- `x, y`: Az ív végpontjának koordinátái

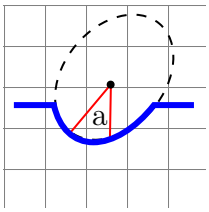
Az alábbi ábrákon szemléltetem a paraméterek hatásait. A bal és jobb oldali egyenes vonalak nem részei az ívnek:

```
<path d="M 25,50
    A 29.5,50 0 0 0 75,50"
    fill="none" stroke="blue" stroke-width="3">
</path>
```



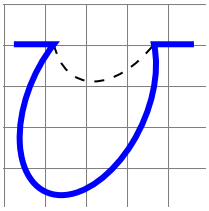
2.14. ábra. Példa: Az ív rx és ry paramétereinek hatása. A kék ív a képzeletbeli ellipszis egy szakasza. A pirossal kiemelt vonalak a sugarakat jelölik.

```
<path d="M 25,50
    A 29.5,50 25 0 0 75,50"
    fill="none" stroke="blue" stroke-width="3">
</path>
```



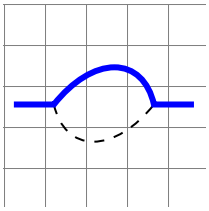
2.15. ábra. Példa: Az ív x-axis-rotation paraméterének használata, kód és vizuális megjelenés

```
<path d="M 25,50
    A 29.5,50 25 1 0 75,50"
    fill="none" stroke="blue" stroke-width="3">
</path>
```



2.16. ábra. Példa: Az ív large-arc-flag paraméterének használata, a szaggatott vonal jelzi az ívet a másik lehetséges érték esetén

```
<path d="M 25,50
    A 29.5,50 25 0 1 75,50"
    fill="none" stroke="blue" stroke-width="3">
</path>
```

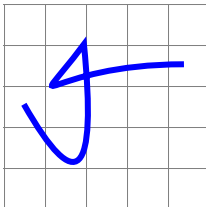


2.17. ábra. Példa: Az ív sweep-flag paraméterének használata, a szaggatott vonal jelzi az ívet a másik lehetséges érték esetén

Komplex útvonalak

Az egyenes szakaszok (`L, l`), Bézier-görbék (`Q, q, C, c`) és ívek (`A, a`) tetszőleges sorrendben kombinálhatók egyetlen <path> elemen belül. Ez teszi lehetővé a komplex, szabálytalan alakzatok leírását.

```
<path d="M10,50 L10,50 Q50,120 40,20 C10,60 20,30 90,30"
    fill="none" stroke="blue" stroke-width="3"/>
```



2.18. ábra. Példa: Tetszőleges komplex útvonal, kódja és vizuális megjelenése

2.1.4. További grafikus elemek

A teljes körű vektorgrafikus ábra nem csak geometriai formákból áll, hanem gyakran tartalmaz szöveget és beágyazott képeket is. Az SVG szabvány ezeknek a hibrid elemeknek a kezelését is lehetővé teszi.

Szövegek

Az SVG a szöveges tartalmat vektorelemként kezeli, nem pedig egyszerű raszteres képként. Ennek előnye, hogy a szöveg kijelölhető marad és nem homályosodik. A `<text>` elem használatos szöveges tartalom beszúrására.

- `x, y`: A szöveg alapvonalának kezdőpontja
- `dx, dy`: Relatív eltolás az előző karakterhez vagy pozícióhoz képest
- `rotate`: Elforgatja az egyes karakterjelek tájolását.
- `fill, stroke`: A szöveg kitöltő színe és körvonala

Viszont egyéb CSS tulajdonság is használható, például `font-family, font-size`, melyek a betűtípust és a szöveg méretét határozzák meg. Azonban a böngészőktől és rendszerektől függő betűtípus-besorolás kompatibilitási problémákat okozhat. Egy, a szerkesztőben megadott `font-family` érték más környezetben eltérő betűtípust aktiválhat, vagy akár a tartalmat megváltoztathatja. Ennélfogva az SVG szerkesztők a szövegeket általában `<path>` elemekké alakítják.

```
<text x="2" y="55"
  rotate="10" font-weight="500" font-size="13"
  fill="gray" stroke="black" stroke-width="0.3">
  Szoveg
  <tspan fill="blue">kiemeles</tspan>
</text>
```

2.19. ábra. Példa: Szöveges tartalom kódja és vizuális megjelenése

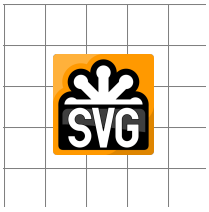
Raszteres képek

Az `<image>` elem teszi lehetővé raszteres fájlok, például PNG vagy JPEG beágyazását a vektorgrafikus ábrába. Ez a mechanizmus hibrid tartalmak létrehozásához szükséges.

- `x, y`: A kép bal felső sarkának koordinátái
- `width, height`: A kép megjelenítési mérete
- `href`: A képfájl forrásának URI-ja (elérési útja)

Az SVG szerkesztők a raszteres képeket általában `base64` formátumúvá alakítják. Majd ez a karakterlánc felhasznált a `href` attribútum értékeként.

```
<image href="svg-logo.png"
    x="25" y="25" width="50" height="50" />
```



2.20. ábra. Példa: Külső kép használata, kód és vizuális megjelenés

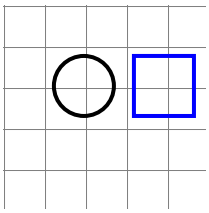
2.1.5. Logikai csoportok és definiált elemek

Csoportosítás

Az SVG dokumentumok egyik legfontosabb szervezőeleme a <g> (group)[9] tárolóelem, mely logikailag és strukturálisan összefogja a benne elhelyezett objektumokat. Használata lehetővé teszi az összetartozó komponensek, például egy ikon részleteinek egyetlen egységként való kezelését. A csoportosítás befolyásolja a transzformációkat is. Az eltolás, forgatás vagy skálázás egységesen érvényesülnek minden a csoportban lévő elemre. Hasonlóan működik a megjelenési attribútumok esetén, a kitöltés vagy a körvonal stílusa automatikusan öröklődnek. Kivéve, ha azok saját beállítással felülírják azt.

Az alábbi példában a csoport fekete körvonalat és egy eltolást definiál. A kör örökli a fekete körvonal színt, míg a téglalap felülírja azt saját kék színével, de az eltolás mindkettőre vonatkozik.

```
<g transform="translate(10,10)"
   stroke="black" stroke-width="2" fill="none">
   <circle cx="30" cy="30" r="15" />
   <rect x="55" y="15" width="30" height="30" stroke="blue" />
</g>
```



2.21. ábra. Példa: Csoportosítás, transzformáció és stílusöröklés hatása, kód és megjelenés

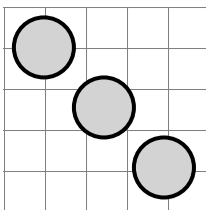
Definiálás

A <defs> elem[8] későbbi felhasználásra szánt grafikus objektumok tárolására szolgál. A tárolt objektumok nem jelennek meg közvetlenül, ehhez hivatkozni kell rájuk. Ennek egyik gyakori módja például a <use> elem használata.

A korábbi példákon látható rács is ezzel a módszerrel készült, így könnyen és egyértelműen elkülöníthető az ábra lényegi része, és egy újrahasznált elem. Az alábbi példában egy kör többszöri felhasználását mutatom be, a <defs> között egyszer meghatározva, majd a <use> elemmel háromszor megjelenítve, különböző helyeken.

```
<defs>
   <circle id="grayCircle" r="15"
   fill="lightgray" stroke="black"
   stroke-width="2"/>
</defs>

<use x="20" y="20" href="#grayCircle" />
<use x="50" y="50" href="#grayCircle" />
<use x="80" y="80" href="#grayCircle" />
```

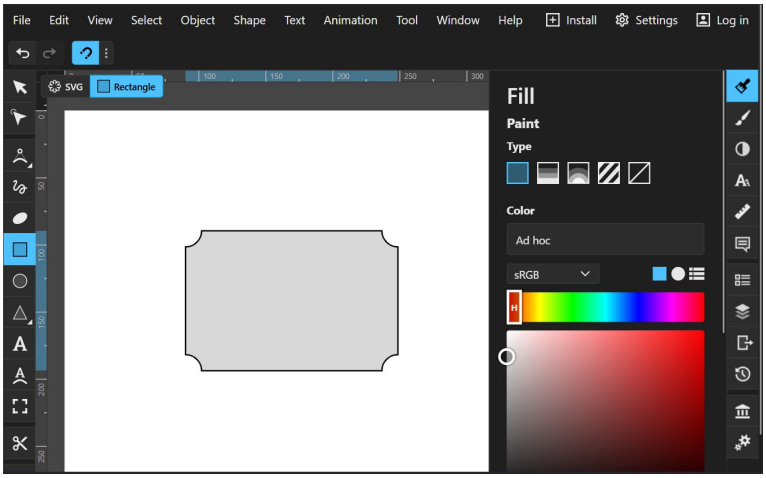


2.22. ábra. Példa: Egy elem definiálása majd többszöri használata, kód és megjelenés

2.2. Szerkesztőeszközök

Több SVG szerkesztő is elérhető, melyek különböző célközönségeknek megfelelő funkcionalitást kínálnak. Egyes esetekben nagyon bő a funkcionalitások választéka, ami magas tanulási görbét eredményezhet. Más esetekben számos előre definiált alakzat áll rendelkezésre, viszont korlátozottak a grafikus szerkesztő eszközök. Az alábbiakban megismerkedhetünk néhány népszerű webes és asztali SVG szerkesztő alkalmazással. Kifejtésre kerülnek összehasonlító szempontok, amik kritériumok lesznek a saját szerkesztőm felé.

2.2.1. Boxy SVG



2.23. ábra. A Boxy SVG kezelőfelülete[15]

A Boxy SVG (2.23. ábra) egy freemium[2] vektorgrafikus szerkesztő. A webalkalmazás alapvető grafikai szerkesztő- és exportálási funkciói elérhetők regisztráció vagy fizetés nélkül. Az asztali alkalmazások egy része és a haladó funkciók, mint az automatikus felhőszinkronizálás és a teljes verziótörténet, fizetős előfizetésekhez kötöttek. A szerkesztőfelület fejlécében egy menüsor van, amellyel elérhető a szerkesztő összes funkcionalitása. Alatta egy transzformációs sáv, mely gyors elérést ad alap csoportosítási, igazítási, forgatási és Boolean operációs eszközökhöz. Középen a rajzoló vászon jelenik meg, ezt körbevéve az úgynevezett vonalzók. Ezek pixelben mérve jelölik a vászon méretét, a kijelölt objektumok méretét és a kurzor pozicíját.

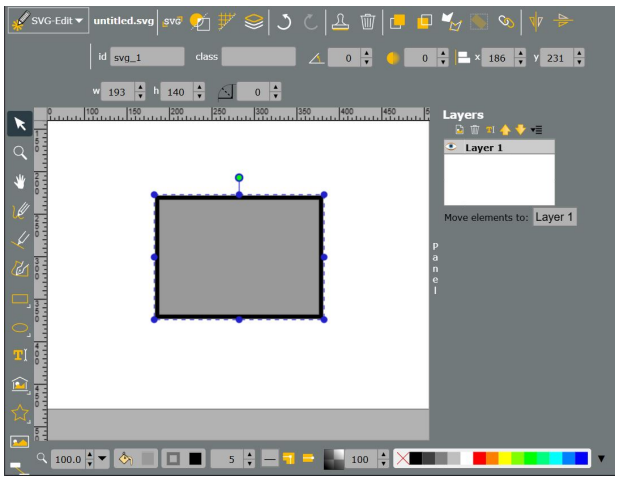
Az SVG szabványban definiált alapelemeken túl ad lehetőséget gyakran használt alakzatok, mint háromszög vagy csillag rajzolására. Ezek beállításai egy bal oldali lebegő ablakban jelenik meg. A tulajdoságok a jobb oldali sávból érhetők el, egy kattintásra beúszó ablakban állíthatók be. Az alsó két gombbal elérhető a kódszerkesztő és az animációs panel, amik alulról úsznak be. A kódszerkesztőben módosítható az SVG egész forráskódja és az adott elem stílusozása is, a változtatás azonnal megjelenik.

Teljeskörű szövegszerkesztésre ad lehetőséget, a legtöbb általános szöveg beállítás megadható. A szöveg in-place, tehát egyenesen a vásznon szerkeszthető. A színek és stílusok megadása széleskörű. Támogatja a lineáris és radiális gradiensek, valamint mintázatok létrehozását. A felhasználó előre definiált színkönyvtárakat is használhat. Az útvonalszerkesztés a szerkesztőeszközzel valósul meg, a csomópontoknál Bézier-

fogantyúkkal manipulálható. Támogatja az unió, kivonás, metszet és komplementer logikai műveleteket az alakzatok között.

Az elrendezést segédvonalak és rácshoz igazítás segíti, mely jelzi az objektumok egymáshoz viszonyított helyzetét. Van lehetőség külön réteg és csoport kezelésre, fa struktúrában látható a DOM hierarchia. Az elemek sorrendje drag-and-drop módszerrel módosítható. A kijelölés történhet kattintással vagy terület kijelöléssel, támogatva a csoportos műveleteket is. A rendszerszintű vágólap műveletek, mint a másolás és beillesztés, billentyűkombinációkkal is elérhetők. A nagyítás és navigáció az egérgörgővel és gesztusokkal intuitív módon működik. A visszavonási előzmények mélyrehatóak, számos lépés visszakereshető. A mentés automatikusan történik a felhőbe.

2.2.2. SVG-Edit



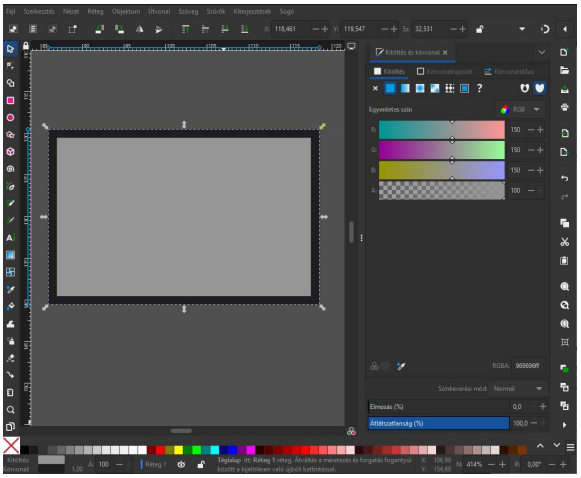
2.24. ábra. Az SVG-Edit kezelőfelülete[16]

Az SVG-Edit (2.24. ábra) egy nyílt forráskódú, ingyenes webalapú szerkesztő. Használatához nincs szükség regisztrációra vagy előfizetésre. A szerkesztő fejlécében lévő menüsor a fájl kezelésére, nézet beállítására és transzformációk beállítására szolgál.

Bal oldalon helyezkedik el az eszköztár kijelölő és rajzoló eszközökkel. Az alapelemek közül támogatja a szabványos alakzatokat, valamint rendelkezik egy beépített alakzat könyvtárral. A kiválasztott objektumok tulajdonságai, mint a kitöltés vagy körvonal, az alsó sávban módosíthatók. Egy dedikált gombbal megnyitható az SVG kódablak, ahol az XML szöveg szerkeszthető. A módosítások elfogadása után jelennek meg a változtatások a vásznon.

A szövegszerkesztés a vásznon történik, elég részletesen szabályozható a karakterek beállítása. Viszont nem támogatja a többsoros bevitelt és csak hét alap betűtípust biztosít. A színeket alap RGBA és HSV értékekkel van lehetőség megadni a bal alsó színválasztóban. Beállítható a színezés típusa is, lehet egyszínű, lineáris színátmenet vagy radiális színátmenet. Az útvonalszerkesztő eszköz lehetővé teszi a csomópontok mozgatását és törlését, de a Bézier-fogantyúk kezelése nem teljesen intuitív. Az igazításhoz bekacsolható rácsháló, egyénileg és csoportosan is rendezhetők objektumok. A rétegkezelés a jobb panelen érhető el, ahol az elemek sorrendje és láthatósága állítható. Támogatja a rendszerszintű másolás-beillesztés műveleteket, valamint rendelkezik előzmény kezeléssel is. Az automatikus mentés beállítható, alapvetően a böngésző helyi tárolójába történik.

2.2.3. Inkscape



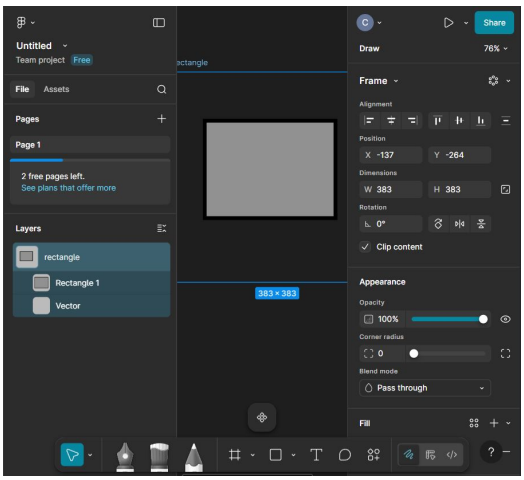
2.25. ábra. Az Inkscape kezelőfelülete[12]

Az Inkscape (2.25. ábra) egy ingyenes, nyílt forráskódú asztali vektorgrafikus szerkesztő. Felülete rendkívül összetett, bal oldalon a részletes eszköztár, felül az eszközvezérlő sáv és a menüsor, jobb oldalon dokkolható párbeszédablakok, alul pedig a színpaletta és az állapotsor található.

Az alapelemek létrehozása mellett számos speciális alakzatot is támogat. A tulajdonságok numerikus beállítását dedikált panelek teszik lehetővé. A beépített XML szerkesztő fa-struktúrában jeleníti meg a DOM-ot, lehetőséget adva bármely attribútum valós idejű módosítására. A kód nemcsak olvasható, hanem teljes mértékben írható is, a változások megerősítés után érvényesülnek.

A szövegszerkesztés nagyon részletes, támogatja a betűközök, sorközök és szöveg folyatását görbére illesztve. Dedikált szöveg szerkesztő ablak is tartozik hozzá, ha nem in-place módon szeretnénk szöveget szerkeszteni. Színkezelése teljes körű, beleértve a CMYK, HSL és RGB színtereket, valamint a komplex gradienseket és mintázatokat. Az útvonalszerkesztés a szoftver egyik legnagyobb erőssége. A csomópont eszköz teljes testreszabhatóságot ad a Bézier-görbék felett. Támogatja a csomópontok típusának váltását és a logikai műveleteket. A rétegkezelés hierarchikus, van opció csoportosításra és rétegek zárolására. Rendelkezik automatikus mentési és helyreállítási funkcióval is.

2.2.4. Figma



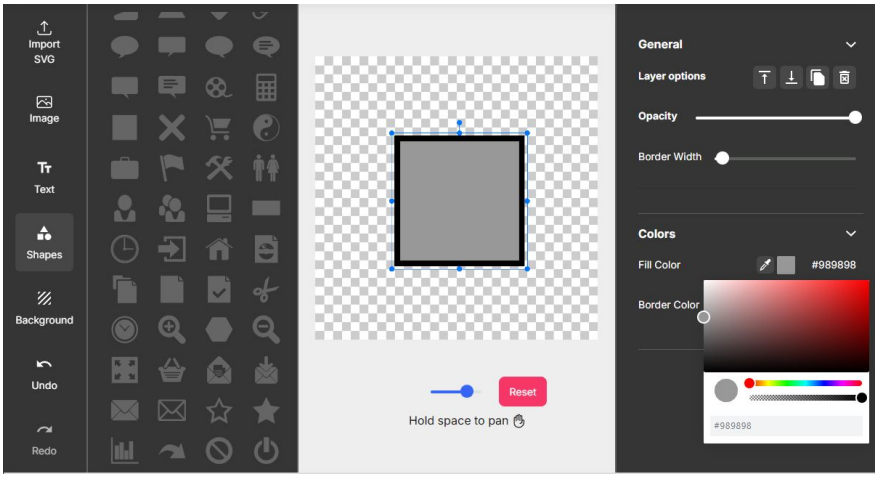
2.26. ábra. A Figma kezelőfelülete[11]

A Figma (2.26. ábra) egy freemium felülettervező eszköz, mely bár elsődlegesen UI/UX designra készült, erős vektorgrafikus képességekkel rendelkezik. Az alapvető szerkesztői funkciók ingyenesen használhatók. A csapatmunkához szükséges prémium és vállalati szintű funkciók havi előfizetéses csomagokban érhetők el. Elrendezése minimalista, a bal oldali sáv a rétegeket és az eszközöket, a jobb oldali sáv a tulajdonságokat tartalmazza, míg a középső vászon a munkafelület.

Az alapelemek egy legördülő menüből érhetők el, de nem túl bőséges a választék. A tulajdonságok szerkesztése a jobb oldali panelen történik, ahol az értékek gombokkal, csúszkákkal vagy numerikusan is megadhatók. Bár az SVG importálása és exportálása megbízhatóan működik, a felhasználó nem szerkesztheti közvetlenül az SVG struktúrát.

A szövegkezelés fejlett, támogatja a Google Fonts integrációt és a részletes tipográfiai beállításokat. A színkezelés stílus könyvtárakra épül, lehetővé téve a színek globális definiálását és újrafelhasználását. Az útvonalszerkesztést a Figma egyedi módon kezeli, a "Vector Networks" technológiával. Ez lehetővé teszi, hogy egy csomópontból kettőnél több vonal induljon ki, egyszerűbbé téve a rajzolást. Az igazítási funkciók és az automatikus elrendezés részletesek. A rétegkezelés fa-struktúrában történik, támogatva a mély kijelölést (Ctrl+Click). A mentés folyamatos és automatikus a felhőbe, teljes verziótörténettel rendelkezik.

2.2.5. Mediamodifier



2.27. ábra. A Mediamodifier kezelőfelülete[13]

A Mediamodifier (2.27. ábra) egy fizetős online marketingeszköz, mely korlátozott ingyenes szerkesztéssel rendelkezik, viszont a letöltéshez előfizetés szükséges. A felület egyszerűsített webes elrendezést követ. Bal oldalon található egy alapvető eszköz sáv, jobb oldalon a tulajdonság beállító panel, középen pedig a vászon.

A szoftver nem rajzolásra, hanem kész elemek összeállítására helyezi a hangsúlyt. Tartalmaz alapvető alakzatokat, de ezek inkább matricaként működnek. A tulajdonságok szerkesztése korlátozott. Az importálás csak alapvető alakzatok vagy görbék esetén működik megbízhatóan. A jobb oldali sávban csúszkákkal állítható az átlátszóság és körvonal méret. A forráskódhoz való hozzáférés teljes mértékben hiányzik, a felhasználó nem látja és nem módosíthatja az SVG struktúrát.

A szövegszerkesztés in-place módon történik, de a formázási lehetőségek kimerülnek a betűtípus és szín választásában. A színkezelés egyszerű színválasztókkal működnek, az átlátszósághoz külön csúszka tartozik és HEX kódként kezeltek. Nincs lehetőség görbéket rajzolni és nincs útvonalszerkesztés. A rétegkezelés csupán az elemek sorrendjének előre vagy hátra állítását teszi lehetővé. A mentés manuális vagy fiókhoz kötött.

2.2.6. Összehasonlítás

Az összehasonlítást az alábbi szempontok alapján végzem el:

Elrendezés: Az elrendezés határozza meg a felhasználó első benyomását. Fontos, hogy a felület átlátható legyen, és a gyakori eszközök kézre essenek. Két fő irányvonal figyelhető meg:

- *Klasszikus:* Bal oldali eszköztár, felső menüsor és jobb oldali tulajdonság panelek. Néhány felugró ablak vagy panel.
- *Panelek és lebegő elemek:* Lebegő panelek, alapvető eszköz sávok, ahol a vászon kapja a legnagyobb teret.

Alapelemek megjelenítése: Az SVG szabványban definiált alapelemek elérésének módja. A felhasználó szempontjából előnyös, ha ezek logikusan csoportosítva jelennek meg, és nem ömlesztve. Szempont továbbá, hogy támogat-e a szerkesztő komplexebb, előre definiált alakzatokat.

Tulajdonságok szerkesztése: A már létrehozott objektumok utólagos módosíthatósága a grafikus felületen keresztül. Ez magában foglalja a transzformációkat és a vizuális attribútumokat. Fontos szempont, hogy a vizuális manipuláció mellett ezek az értékek numerikusan is megadhatók legyenek.

Kód nézet és szerkesztés: A grafikus szerkesztők egyik legfontosabb pontja fejlesztői szempontból. Azt vizsgálom, hogy a felhasználó hozzáfér-e a generált SVG forráskódhoz szerkesztés közben:

- *Élő szerkesztés:* A kódpanelen végzett módosítás megerősítés után vagy azonnal megjelenik a vásznon. Ezek külön kategorizáltak XML, forrás és konkrét szerkesztő nézetre.
- *Nincs:* A szerkesztő teljesen elrejti a forráskódot.

Szövegek szerkesztése: Van-e lehetőség több soros, bekezdéses szöveg beszúrására, vagy csak egy sorba kerül. A szerkesztőnek biztosítania kell a betűtípus, méret, stílus és igazítás beállításait. A szöveg szerkesztése közvetlenül a vásznon *"in-place"* történik-e. Alternatívan egy felugró ablakban vagy külön beviteli mezőben.

Színek és stílusok megadása: Az SVG stílusozás és CSS kezelése.

- A kitöltés és körvonal támogatja-e az RGB, HEX, HSL[1] színmegadást, valamint az átlátszóságot.
- Van-e lehetőség lineáris és radiális színátmenetek, illetve mintázatok létrehozására és szerkesztésére.

Útvonalszerkesztés: Milyen szinten támogatott a `<path>` elem manipulációja. A szempont azt vizsgálja, hogyan kezelhetők a csomópontok:

- Hozzáadhatók és törölhetők-e pontok?
- A Bézier-görbék kontrollpontjai, fogantyúi intuitívan mozgathatók-e?
- Támogatja-e a logikai műveleteket, mint unió, különbség, metszet és komplementer az alakzatok között.

Objektumok automatikus igazítása: Rendelkezik-e a szerkesztő "mágneses" funkcióval, mely az objektumokat rácspontokhoz, más objektumok éleihez vagy középpontjához igazítja mozgatás közben.

Rétegek és csoportok kezelése: Az SVG DOM struktúrájának kezelése. Van-e lehetőség elemek csoportosítására, illetve külön rétegek létrehozására. Fontos, hogy a szerkesztő vizuálisan megjelenítse a hierarchiát, és lehetővé tegye az elemek sorrendjének drag-and-drop módosítását.

Kijelölés: Milyen módszerekkel választhatók ki az elemek. Kattintással kiválasztható egy elem, téglalap alapú kijelöléssel több is. *Shift* vagy *Ctrl* billentyűkkel több elem együttes kezelése.

Másolás és beillesztés: Működnek-e a rendszerszintű vágólap műveletek. A másolás `Ctrl+C`, a beillesztés `Ctrl+V` és a vágás `Ctrl+X` billenytűkombinációkkal. Webes környezetben ez biztonsági korlátok miatt gyakran nehézkes, ezért vizsgálom, hogy van-e belső megoldás az elemek duplikálására.

Nagyítás: Mivel az SVG skálázható, a vászon nagyításának és mozgatásának intuitívnak kell lennie. Egérgörgővel vagy gesztusokkal is vizsgáltam a funkcionalitás működését.

Undo-Redo funkciók: Az előzmények kezelése. Van-e lehetőség a műveletek visszavonására és megismétlésére. Ez mennyire mélyreható, hány lépést jegyez meg a rendszer. Elvárás a visszavonás `Ctrl+Z`, és megismétlés `Ctrl+Y / Shift+Z` billentyűkombinációk támogatása.

Automatikus mentés: Webes alkalmazásoknál kritikus, hogy hálózati hiba vagy véletlen bezárás esetén megmarad-e a munka. A szerkesztő menti-e a böngésző helyi tárolójába (Local Storage) vagy felhőbe az állapotot.

Ezeket az összehasonlítási szempontokat tartalmazza a (2.1.) táblázat.

2.1. táblázat. Webes és asztali SVG szerkesztők funkcionális összehasonlítása

	Inkscape	**SVG-Edit**	**Boxy SVG**	**Figma**	**MediaM.**
Elrendezés	Klasszikus			Panelek / Lebegő elemek	
Alapelemek	Bal oldali sáv				
Tulajdonság szerk.	Felső és Alsó sáv + Jobb panel	Felső + Alsó sáv	Felső sáv + Jobb panel	Jobb panel	
Kód nézet	Élő XML	Élő Forrás	Élő Szerkesztő	Nincs	
Szöveg	In-place + szerkesztő	In-place			
Színek	Teljes	RGBA HSV	Könyvtárak, Gradiensek		HEX kód
Útvonal-szerkesztés	Teljes körű	Alapvető	Fejlett	Vector networks	Nincs
Igazítás	Fejlett	Alap	Okos segédvonalak		Alap
Rétegek	Teljes hierarchia kezelés				Előre-hátra tolás
Kijelölés	Terület + Kattintás + Csoportos				Terület + Kattintás
Másolás és beillesztés	Gombok és billenytűkombinációk				
Nagyítás	Gombok és billenytűk.	Opciók és gombok	Gombok és billenytűkombinációk		
Automatikus mentés	Igen	Helyi tároló	Felhő alapú		Nem

3. fejezet

Tervezés

Itt kezdődik a dolgozat lényegi része, úgy értve, hogy a saját munka bemutatása. Jellemzően ebben szerepelni szoktak blokkdiagramok, a program struktúrájával foglalkozó leírások. Ehhez célszerű UML ábrákat (például osztály- és szekvenciadiagramokat) használni.

Amennyiben a dolgozat inkább kutatás jellegű, úgy itt lehet konkretizálni a kutatási módszertant, a kutatás tervezett lépéseit, az indoklást, hogy mit, miért és miért pont úgy érdemes csinálni, ahogyan az a későbbiekben majd részletezésre kerül.

Ebben a fejezetben az implementáció nem kell, hogy túl nagy szerepet kapjon. Ez még csak a tervezési fázis. (Nyilván ha olyan a téma, hogy magának az implementációnak a módjával foglalkozik, adott formális nyelvet mutat be, úgy a kódpéldákat már innen sem lehet kihagyni.)

3.1. Táblázatok

Táblázatokhoz a `table` környezetet ajánlott használni. Erre egy minta a 3.1. táblázat. A hivatkozáshoz az egyedi `label` értéke konvenció szerint `tab:` prefixszel kezdődik.

3.1. táblázat. Minta táblázat. A táblázat felirata a táblázat felett kell legyen!

a	b	c
1	2	3
4	5	6

3.2. Ábrák

Ábrákat a `figure` környezettel lehet használni. A használatára egy példa a 3.1. ábrán látható. Az `includegraphics` parancsba Az ábrák felirata az ábra alatt kell legyen. Az ábrák hivatkozásához használt nevet konvenció szerint `fig:`-el célszerű kezdeni.

3.3. További környezetek

A matematikai témájú dolgozatokban szükség lehet tételek és bizonyításaik megadására. Ehhez szintén vannak készen elérhető környezetek.

3.1. ábra. A Miskolci Egyetem címere.

3.1. definíció. Ez egy definíció

3.2. lemma. *Ez egy lemma*

3.3. tétel. *Ez egy tétel*

Bizonyítás. Ez egy bizonyítás □

3.4. következmény. *Ez egy tétel*

3.5. megjegyzés. Ez egy megjegyzés

3.6. példa. Ez egy példa

4. fejezet

Megvalósítás

Ez a fejezet mutatja be a megvalósítás lépéseit. Itt lehet az esetlegesen előforduló technikai nehézségeket említeni. Be lehet már mutatni a program elkészült részeit.

Meg lehet mutatni az elkészített programkód érdekesebb részeit. (Az érdekesebb részek bemutatására kellene szorítkozni. Többségében a szöveges leírásnak kellene benne lennie. Abból lehet kiindulni, hogy a forráskód a dolgozathoz elérhető, azt nem kell magába a dolgozatba bemásolni, elegendő csak behivatkozni.)

A dolgozatban szereplő forráskódrészletekhez külön vannak programnyelvenként stílusok. Python esetében például így néz ki egy formázott kódrészlet.

A stílusfájlok a `styles` jegyzékben találhatók. A stílusok között szerepel még C++, Java és Rust stílusfájl. Ezek használatához a `dolgozat.tex` fájl elején `usepackage` paranccsal hozzá kell adni a stílust, majd a stílusfájl nevével megegyező környezetet lehet használni. További példaként C++ forráskód esetében ez így szerepel.

Stílusfájlokból elegendő csak annyit meghagyni, amennyire a dolgozatban szükség van. Más, C szintaktikájú nyelvekhez (mint például a JavaScript és C#) a Java vagy C++ stílusfájlok átszerkesztésére van szükség. (Elegendő lehet csak a fájlnevet átírni, és a fájlban a környezet nevét.)

Nyers adatok, parancssori kimenetek megjelenítéséhez a `verbatim` környezetet lehet használni.

```
$ some commands with arguments
1 2 3 4 5
$ _
```

A kutatás jellegű témáknál ez a fejezet gyakorlatilag kimaradhat. Helyette inkább a fő vizsgálati módszerek, kutatási irányok kaphatnak külön-külön fejezeteket.

5. fejezet

Tesztelés

A fejezetben be kell mutatni, hogy az elkészült alkalmazás hogyan használható. (Az, hogy hogyan kell, hogy működjön, és hogy hogy lett elkészítve, az előző fejezetekben már megtörtént.)

Jellemzően az alábbi dolgok kerülhetnek ide.

- Tesztfuttatások. Le lehet írni a futási időket, memória és tárigényt.
- Felhasználói kézikönyv jellegű leírás. Kifejezetten a végfelhasználó szempontjából lehet azt bemutatni, hogy mit hogy lehet majd használni.
- Kutatás kapcsán ide főként táblázatok, görbék és egyéb részletes összesítések kerülhetnek.

6. fejezet

Összefoglalás

Hasonló szerepe van, mint a bevezetésnek. Itt már múltidőben lehet beszélni. A szerző saját meglátása szerint kell összegezni és értékelni a dolgozat fontosabb eredményeit. Meg lehet benne említeni, hogy mi az ami jobban, mi az ami kevésbé jobban sikerült a tervezettnél. El lehet benne mondani, hogy milyen további tervek, fejlesztési lehetőségek vannak még a témával kapcsolatban.

Irodalomjegyzék

[1] Andrew Braun. Color codes: What's the difference between hex, rgb, and hsl? https://www.maketecheasier.com/difference-between-hex-rgb-hsl/.

[2] Britannica. Freemium. https://www.britannica.com/topic/freemium.

[3] Britannica. Raster graphics. https://www.britannica.com/technology/raster-graphics.

[4] MDN Web Docs. Bézier curve. https://developer.mozilla.org/en-US/docs/Glossary/Bezier_curve.

[5] MDN Web Docs. Getting started with svelte. https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Frameworks_libraries/Svelte_getting_started.

[6] MDN Web Docs. Svg attribute reference. https://developer.mozilla.org/en-US/docs/Web/SVG/Reference/Attribute.

[7] MDN Web Docs. Svg element reference. https://developer.mozilla.org/en-US/docs/Web/SVG/Reference/Element.

[8] MDN Web Docs. Svg element reference - defs. https://developer.mozilla.org/en-US/docs/Web/SVG/Reference/Element/defs.

[9] MDN Web Docs. Svg element reference - g. https://developer.mozilla.org/en-US/docs/Web/SVG/Reference/Element/g.

[10] MDN Web Docs. Typescript. https://developer.mozilla.org/en-US/docs/Glossary/TypeScript.

[11] Figma. https://www.figma.com/.

[12] Inkscape. https://inkscape.org/about/.

[13] Mediamodifier. https://mediamodifier.com/.

[14] Svelte. https://svelte.dev/docs/svelte/overview.

[15] Boxy SVG. https://boxy-svg.com/.

[16] SVG-edit. https://svgedit.netlify.app/editor/index.html.

[17] W3C. Scalable vector graphics (svg) 2. https://www.w3.org/TR/SVG2/.

CD Használati útmutató

Ennek a címe lehet például *A mellékelt CD tartalma* vagy *Adathordozó használati útmutató* is.

Ez jellemzően csak egy fél-egy oldalas leírás. Arra szolgál, hogy ha valaki kézhez kapja a szakdolgozathoz tartozó CD-t, akkor tudja, hogy mi hol van rajta. Jellemzően elég csak felsorolni, hogy milyen jegyzékek vannak, és azokban mi található. Az elkészített programok telepítéséhez, futtatásához tartozó instrukciók kerülhetnek ide.

A CD lemezre mindenképpen rá kell tenni

- a dolgozatot egy `dolgozat.pdf` fájl formájában,
- a LaTeX forráskódját a dolgozatnak,
- az elkészített programot, fontosabb futási eredményeket (például ha kép a kimenet),
- egy útmutatót a CD használatához (ami lehet ez a fejezet külön PDF-be vagy MarkDown fájlként kimentve).