Matematikai és Informatikai Intézet

Kör- és gömbalapú modellezés megvalósítása subdivision görbékkel és felületekkel

Készítette
Szabó Nóra
Programtervező Informatikus BSC

Témavezető
Troll Ede Mátyás
tanársegéd

Eger, 2022

Tartalomjegyzék

Bevezetés

Szakdolgozatom témája a kör- és gömbalapú modellezés megvalósítása subdivision görbékkel és felületekkel. Ezen probléma grafikus megoldást igényel, ami a fő oka, hogy ezt a témát választottam, mert érdekel a grafika és kihívást láttam ebben a feladatban. A grafika viszont rengeteg matematikát követel, ami eleinte meglepő és ijesztő volt, de idővel rájöttem hogy nagy segítség . Mivel teljesen ismeretlenül vágtam bele ebbe a projektbe, elsősorban az általános grafikai feladatok elvégzését tanultam meg, mint a pont illetve egyenes elhelyezése, koordinátáikkal való számítások, a kiszámolt pontok, egyenesek rajzolása, ezek díszítése különböző színekkel és méretekkel. Újdonság volt a már lerajzolt pontok elmozgatása, ami egyenesnél nem csak azt jelentette, hogy az egyik végpontja arrébb kerül, hanem az egyenlete is megváltozik, valamint az egér mozgatásával egyidőben láthatjuk, hogy milyen lesz az így manipulált egyenes új alakja. Miután elsajátítottam az alapvető lépéseket megbirkóztam a bonyolultabb elemek koordinátáinak kiszámolásával és megjelenítésével, pl kör és egyéb görbék rajzolása. Ezekhez paraméteres görbéket használtam, hisz az lehetővé teszi, hogy egy x tengelyű koordinátához több y-tengelyű koordináta is hozzátartozhasson és fordítva.

1. fejezet

Paraméteres görbék

A síkban mozgó pont egy görbét ír le. Ha minden t időpillanatban meghúzzuk az op vektort, és ezt r(t)-vel jelöljük, akkor egy I intervallumon értelmezett vektorfüggvényt kapunk. Ezt a vektorfüggvényt általában koordinátafüggvényeivel adjuk meg:
$x(t) = (x(t), y(t))$
A koordinátafüggvények általában valós változós, valós értékű függvények. Ha úgy adjuk meg a görbét definiáló egyenletrendszert, hogy abból a görbe koordinátáit segédváltozókkal kaphatjuk meg, paraméteres görbéről beszélhetünk. Ilyen paraméteres egyenletrendszert létrehozhatunk a parabola egyenletéből is. Az egyszerű egyenlete a következő:
$y = x^2$
Ha egy kis változtatást végrehajtunk, máris egyenletrendszert kaphatunk belőle:
$x = t$
$y = t^2$
Több dimenziós görbe leírására is kényelmesebb opció a paraméteres egyenletrendszert használnunk:
$x = a\cos(t)$
$y = a\sin(t)$
$z = bt$
Számítógépen az r(t) görbét töröttvonallal közelítjük. Kiszámoljuk a
$p_0 = r(t_0), p_1 = r(t_1) \dots p_{n-1} = r(t_{n-1}), p_n = r(t_n)$
pontokat, ahol $t_0 = a$ és $t_n = b$, majd a szakaszokat megjelenítjük.

1.1. Egyenes

Az egyenes egyenlete olyan paraméteres egyenlőség, ahol a paraméterek minden valós értékére valamely egyenes egyenlete megkapható, és fordítva. Egy egyenest meghatároz két pontja, amit koordinátákkal adunk meg. Ezen két ponton átmenő egyenes egyenletét felhasználhatjuk geometriai számításokra.

1.2. Kör

1.3. Kontroll alakzatokkal definiált görbék

Kontroll alaktatoknak nevezzük azokat az alakzatokat, melyek befolyásolják egy másik alakzat vagy görbe pontjait.

1.3.1. Hermite ív

1.3.2. Bézier görbe

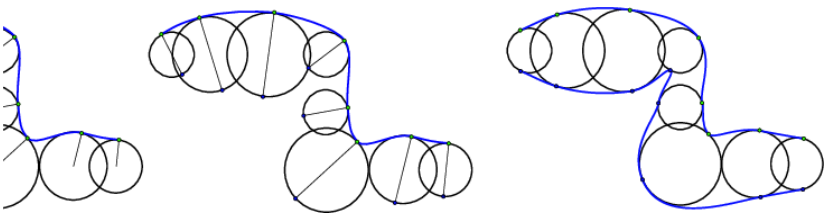
1.3.3. B-Spline görbe

2. fejezet

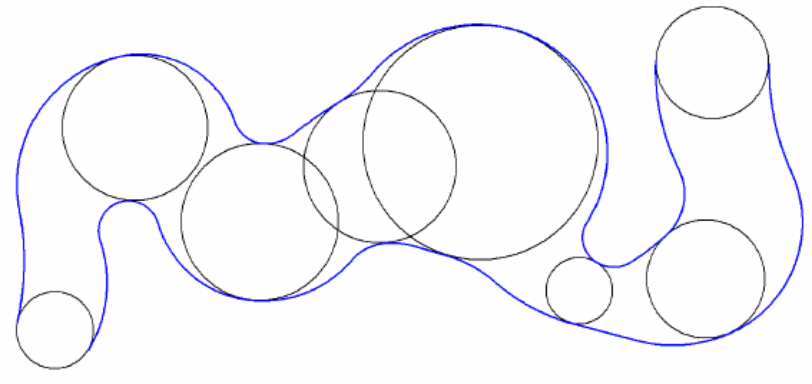
Gömbalapú modellezés

2.1. A feladat

A feladat az, hogy a rajzoló felületre köröket létrehozva megjelenjenek a hozzájuk tartozó görbék belső és külső íven, azok esztétikusan folyamatosak legyenek. Ehhez ki kell számolni 3 egymás utáni körhöz tartozó Apollonius köröket, szelektálni azokat megfelelően, majd az Apollonius körök és az eredetileg rajzolt közök metszéspontját kiszámolni és eltárolni. Ezekbe a metszéspontokba az eredeti körökre érintő egyenest kell rajzolni, melynek a hosszát és irányát szintén ki kell számolni. Ez lesz az Hermite íveink kontroll pontja. Az érintőegyenes hosszát a körök közti hatványvonalak határozzák meg. Az Hermite íveket a megfelelő belső illetve külső érintési pontokból indítjuk a következő érintési pontra úgy, hogy egy kört csak egyszer érintsen a belső, és egyszer a külső görbe. Ez után, mindezt implementálni kell 3 dimenziós programba is, ahol a körökből gömbök, az Hermite ív párokból pedig cső lesz. A célom tehát az volt, hogy egy jól működő programot hozzak létre, amelyben nem meghatározott mennyiségű kört rajzolhatok, amik köré görbék jelennek meg úgy, hogy azok úgymond tükörképei legyenek a másik egyenesnek az alábbi módon:



2.1. ábra. Program eredményének elképzelése



2.2. ábra. Program eredményének elképzelése másik verzió

2.2. A megoldás és annak implementációja

A megoldáshoz fel kell építeni az alapokat mind fejben mind az alkalmazásban. Visual Studio-ban létrehoztam egy C# programozási nyelvű Windows Forms Application alkalmazást, a Design tervezőben raktam egy vásznat. Lehetőség van események kezelésére, mint az kurzorral történő kattintás, arrébb húzás, elengedés. Ezeket később beállítottam. Elsősorban létrehoztam pár segédváltozót az egér kezeléséhez, valamint a listákat amiket már tudtam előre hogy használni fogok, pl körök listáját, belső és külső érintő pontokat, valamint a rajzoláshoz egy vastagságot jelző változót, a Graphics osztályból egy példányt, amit rögtön inicializálok a program elején. Mivel köröket kell kirajzolni, ezért létrehoztam egy „Circles" osztályt a szükséges mezőkkel mint a középpontja koordinátája, a sugarának végpontja azaz a körív egy pontja, a színe. A kör kirajzolásához létrehoztam egy metódust, mely paraméterben megkapja a szükséges adatokat, majd kirajzolja a körünket. Az itt paraméterben látható kör akkor jön létre, mikor az egerünkkel kattintunk a felületre, lenyomva tartjuk a gombot majd a megfelelő méret beállítása után elengedjük, így a kör osztály minden kötelező információjának eleget teszünk, azaz a kör középpontja az lesz, ahol elkezdtük a rajzolást, a P koordinátája ahol befejeztük, ebből pedig már ki tudja számolni a sugár hosszát. Az egérfigyelő esemény segítségével hozok létre köröket, úgy, hogy az egérgomb lenyomásakor megvizsgálom, üres felületre kattintottam-e. Ha igen, akkor létrehozok egy új kört, a segédváltozót úgy állítom be, hogy az aktuális körünk sugarát méretezze, illetve itt már látszódik a körív, így ki tudom választani a számomra ideális méretű kört. Miután felengedem a kurzort, az előbb lerajzolt kör megtartja méretét. Viszont ha az egérgomb lenyomásakor nem üres felületet találtam el, hanem ott egy létező pont van, akkor azt kell megvizsgálnom, hogy középpont, vagy sugár végpontja-e, majd az szerint kell változtatni a körön; mozgatni vagy méretezni. Minden esetben, miután felengedtem a gombot, nem tudom mozgatni a köröket a gomb újbóli lenyomásáig, valamint a segédváltozót vissza kell állítani egy semleges állapotra.

```
1 reference
public static void DrawCircle(this Graphics g, Circle circle)
{
    float t = 0.0f;
    float b = 2 * (float)Math.PI;
    float h = b / 200.0f;
    Pen p = new Pen(circle.Color, circle.LineWidth);
    PointF p1;
    PointF p0 = new PointF(circle.R * (float)Math.Cos(t) + circle.X,
                           circle.R * (float)Math.Sin(t) + circle.Y);
    while (t <= b)
    {
        t += h;
        p1 = new PointF(circle.R * (float)Math.Cos(t) + circle.X,
                        circle.R * (float)Math.Sin(t) + circle.Y);
        g.DrawLine(p, p0, p1);
        p0 = p1;
    }
}
```

2.3. ábra. Körrajzoló algoritmus kódja

2.2.1. Érintési pontok meghatározása

Az érintési pontok kiszámításához, meghatározásához először meg kell értenünk az Apollonius problémát. Három kör kirajzolása után megjelenik az első két Apollonius kör. Az Apollonius körök azon körök, melyek 3 körnél számíthatóak és egy-egy pontban érintik mindhárom kört. 8 eset van belőle: mindhárom kör az Apollonius körön belül van, csak 2 kör van az Apollonius körön belül, ez 3 kombinációban tud előfordulni, csak 1 kör van az Apollonius körön belül, ez is 3 kombinációban fordulhat elő, illetve egyik kör sincs belül, de minden esetben érinti mindhárom kört csak és csakis egy pontban, ez az az érintési pont amit eltárolunk. Nekünk a két szélsőséges eset kell, mikor egy kör sincs az Apollonius körünkön belül, ez fogja meghatározni a belső érintési pontokat, illetve mikor mindhárom kör belül van, ez pedig a külső érintési pontokat adja meg. Az Apollonius problémához külön osztályt hozok létre, amiben tárolom a 3 kört, amivel számolni kell, illetve 4 metódust. Ezek között van 2 transzformálás, amiket egy külön metódusban használok az Apollonius kör kiszámítására, utolsóként pedig az érintő kör és a problémában szereplő három kör közös pontjait számolom ki. Ezen a képen láthatóak piros körívvel a rajzolt körök, zöld körívvel az Apollonius körök, piros négyzettel pedig az érintő pontokat jelöltem. A későbbiekben az Apollonius köröket nem jelenítem meg, hogy ne kavarodjunk bele a sok színes jelölésbe. Ezeket az érintési pontokat eltároltam két külön listába, egy „innertanpoints” és egy „outertanpoints” nevűbe, hogy könnyen meg lehessen különböztetni.

2.2.2. Hermite-ívek érintőinek meghatározása

Az Hermite-ívek eléréséhez sok dolgot kell ismernünk. Az eddigieken felül szükségünk van a rajzolt körök érintőire, melyek az Apollonius körök érintési pontjaiból húzódnak. Hosszuk és irányuk kiszámításához a hatványvonalak és a kör középpontja közötti

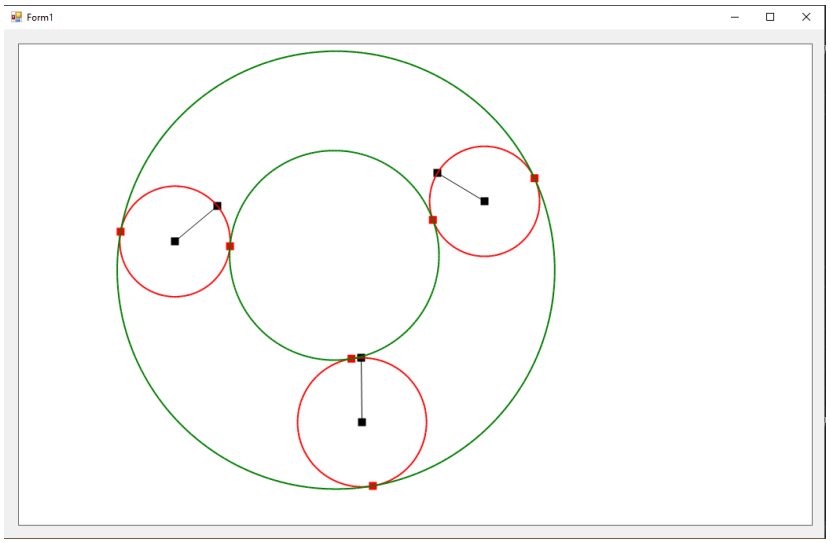
távolság kétszeresét kell használni. A hatványvonal két kör középpontjait összekötő egyenesre merőlegesen elhelyezkedő egyenes. Erre a feladatra több metódust is létrehoztam, hogy jobban értelmezni lehessen. Az egyik kiszámolja, milyen távol helyezkedik el a hatványvonal és a középpontokat összekötő vonal metszete az első kör középpontjától, a következő a hatványvonal egy másik pontját adja meg, hogy egyszerűbb legyen a rajzolás valamint a következő lépéshez a számolás. A hatványvonal hosszát a két kör középpontját összekötő szakasztól egyenlő hosszúságúra állítottam be, ennek a hossznak nincs lényege.

A narancssárga vonalak jelölik a hatványvonalakat.

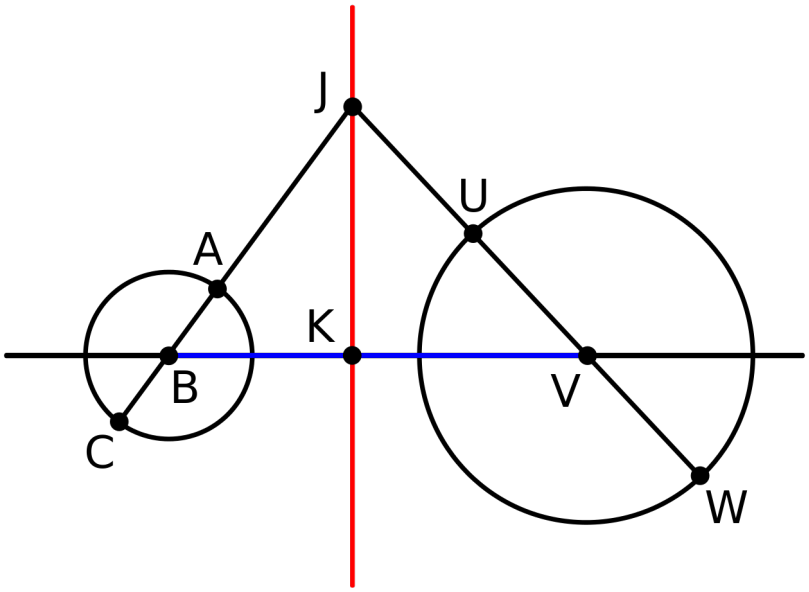
Ez után az érintőpontokba való érintővonal húzása következik. Ehhez használnunk kell a hatványvonal és az adott kör középpontjának távolságának kétszeresét. Pont és egyenes távolsága a pontból az egyenesre húzott merőleges szakasz hossza. Ennek kiszámítását szintén külön metódusba foglaltam. Az érintő végpontját vektorok segítségével számoltam ki. Az érintőpontot nevezzük t-nek, a sugár vektorát pedig r_v-nek. Létrehozok egy új vektort t_v néven, ami a t pontos helyére fog mutatni. Ebből a két vektorból már ki tudom számolni a sugárra merőleges érintőjét a körnek úgy, hogy a t_v ponthoz hozzáadom a r_v sugarat, az x és y koordinátáját felcserélve és az egyiket(pl x-et) megszorozva -1-el, majd ezt egységvektorra redukálom és megszorzom az egészet az előzőekben kiszámolt hatványvonal és középpont távolságának kétszeresével. így megkapom az érintővonal végének pontos koordinátáit.

Az Hermite-ívek így már rajzolhatóak. A rajzoló metódusba 4 pontot kell megadni, ebből az első az ív kezdőpontja, a második a végpontja. Az első ívnél a kezdőpont 3 kör esetében az első érintőpont, a végpont pedig a második érintőpont. A maradék két pont pedig a kontrollpontok, azaz az érintővonalak végpontjai. Ezek segítenek formálni az ívet, hogy a megfelelő irányba álljanak és megfelelő ívvel görbüljenek. Ugyanennyi körnél a második ív első pontja értelemszerűen a második érintőpont, utolsó a harmadik érintőpont, és a hozzájuk tartozó érintővonalak végpontjai sorrendben.

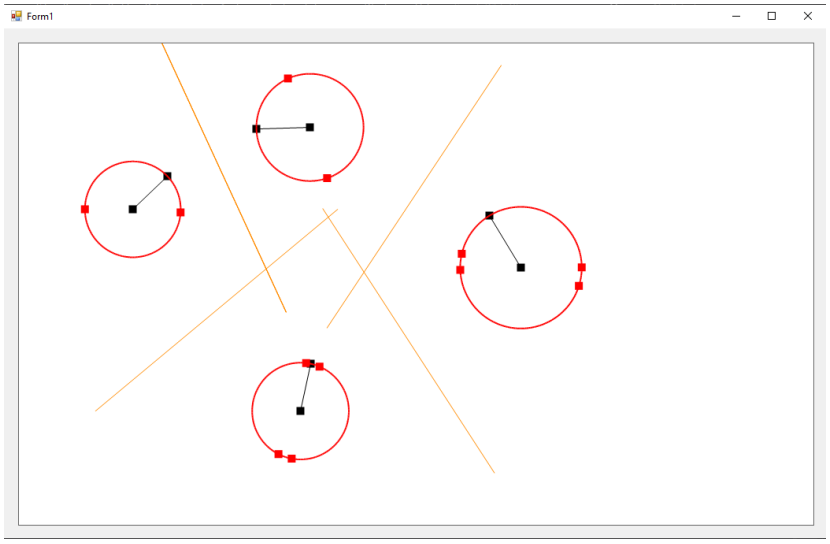
Több kör rajzolása esetén bonyolultabb ez a helyzet, mert nem az összes érintési pontból kell Hermite-ívet húzni, ugyanis úgy mindenfele görbék lennének. Így a már korábban bevezetett segédváltozót vettem használatba úgy, hogy a ciklusban, ahol rajzoltam az íveket beállítottam, hogy több mint 3 kör esetén az egyik legyen az aktuális szám 3-szorosa, mivel egy Apollonius kör 3 érintési pontot eredményez, és ezzel megfelelően operálva csak a szükséges érintési pontból indulnak az ívek. Az érintési pontok sorszámai amiket figyelembe kell venni: 1. 2. 3. 6. 9. 12. stb



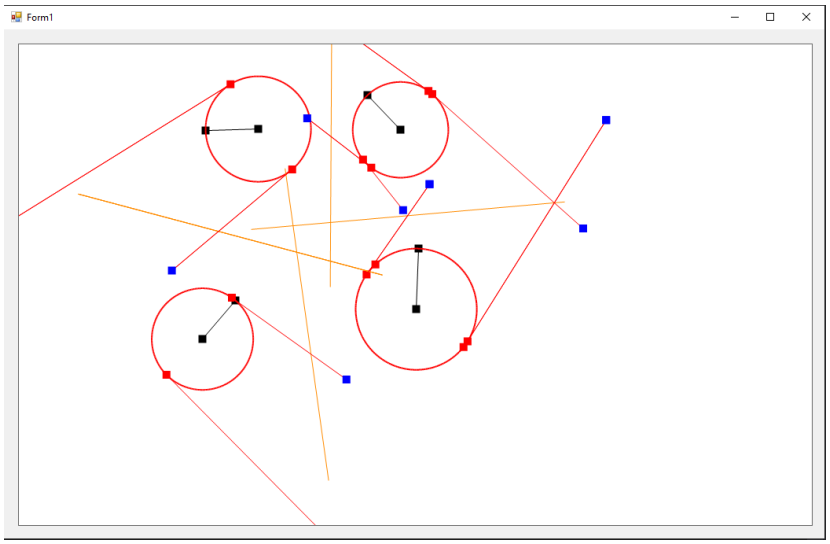
2.4. ábra. A rajzolt-és már kiszámolt Apollonius körök



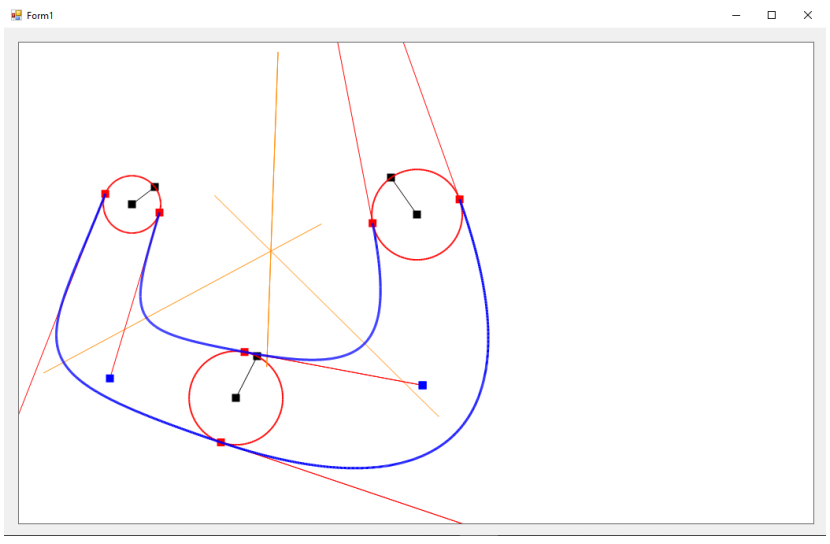
2.5. ábra. Hatványvonal kiszámításához ábra



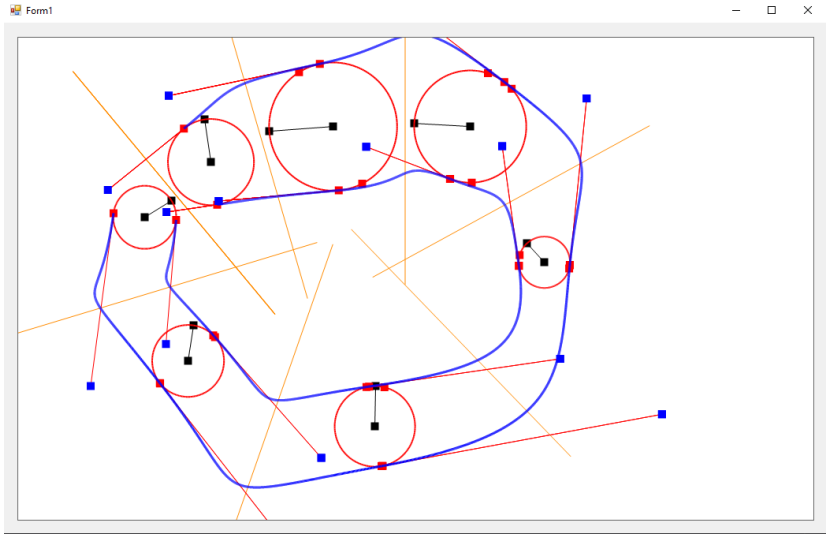
2.6. ábra. Hatványvonalak



2.7. ábra. Érintővonalak és végpontjaik



2.8. ábra. Hermite-ívek 3 kör esetében

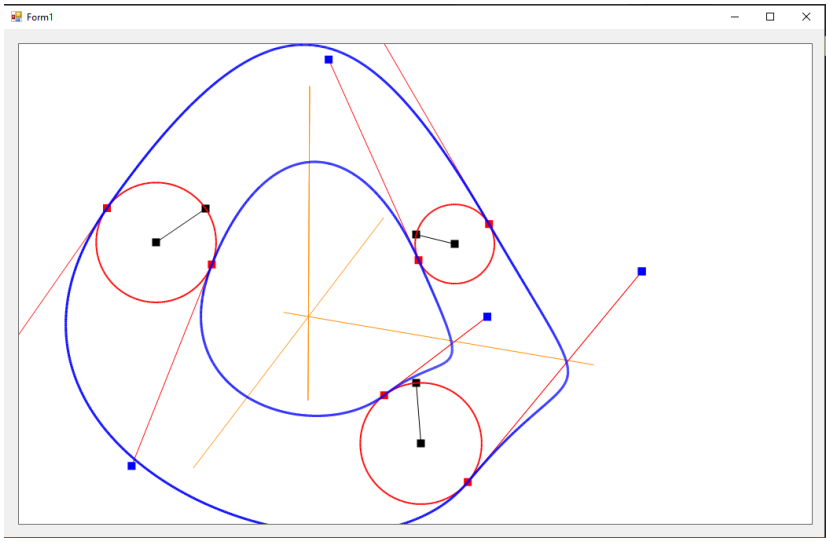


2.9. ábra. Hermite-ívek sok körnél

3. fejezet

Összegzés, további feladatok

Gyakorlott lettem a grafikai alap elemek használásában, mint a pontok, egyenesek rajzolása, külalakjuk változtatása, mozgatásuk, vektorokkal való számítások elvégzése pl összeadás, kivonás, szorzások, eltolás, merőleges előállítása stb. Bonyolult formák számításának implementálása pl Apollonius körök, Hermite-ív, pont és egyenes távolságának kiszámítása, ezek kirajzolása, manipulálása. Módosíthatom a programomat úgy, hogy az ívek eleje és vége összekapcsolódjanak, így zárt alakzatokat kapunk.



3.1. ábra. Zárt Hermite-ívek

3.1. További feladatok

Hermite-ívek görbüléseinek finomítása, irány megfelelő kiszámítása 100%-os sikerességgel, 2 dimenziós programból 3 dimenzióssá alakítás

Ábrák jegyzéke

Irodalomjegyzék

[1] Apollonius probléma: https://mathworld.wolfram.com/ApolloniusProblem.html

[2] Apollonius kör: https://mathworld.wolfram.com/ApolloniusCircle.html

[3] Apollonius probléma wikipedia oldala: https://en.wikipedia.org/wiki/Problem_of_Apollonius

[4] Apollonius körrajzoló program: https://www.walter-fendt.de/html5/men/apolloniosproblemccc_en.htm

[5] Apollonius probléma algebrai megoldása: http://claus-jo.dk/Apollonius_Uk.html#t4

[6] hatványvonal https://en.wikipedia.org/wiki/Radical_axis

[7] hatványvonal2 https://mathworld.wolfram.com/RadicalLine.html

[8] hatványvonal3 http://www.nabla.hr/PC-CircleAndLineCnt3.htm

[9] hatványvonal4 http://lexikon.fazekas.hu/184.html

[10] Hoffmnann Miklós: Topológia és differenciálgeometria

[11] Hajós György: Bevezetés a geometriába

[12] M. Hoffmann, R. Kunkli: Skinning of circles and spheres

[13] M. Hoffmann, J. Monterde, E. Troll: Blending of spheres by rotation-minimizing surfaces