

TDK-dolgozat

Horváth Anna Lili

Általános geometriai mezők és szűrésük

EÖTVÖS LORÁND TUDOMÁNYEGYETEM

INFORMATIKAI KAR

ALGORITMUSOK ÉS ALKALMAZÁSAIK TANSZÉK



Szerző:

Horváth Anna Lili

programtervező informatikus MSc

1. évfolyam

Témavezetők:

Bán Róbert

doktorandusz

Valasek Gábor

egyetemi adjunktus

Budapest, 2024

Tartalomjegyzék

1. Bevezetés	3
1.1. Geometriai távolságmezők	3
1.2. Szakirodalmi áttekintés	4
1.3. Elsőrendű mezők szűrése	4
1.4. Általános felületek távolságmezője	5
2. Elsőrendű távolságmező szűrése	6
2.1. Távolságfüggvények	6
2.2. A távolságmező generálása	7
2.2.1. Távolságmező reprezentálása	7
2.2.2. Proxi geometria	8
2.3. A mező kiértékelése	11
2.3.1. Eredmények	13
2.4. CSG szűrés 2D-ben	16
2.4.1. CSG művelet két geometriára	16
2.4.2. CSG fa meghatározása	17
2.4.3. CSG fák generálása	20
2.4.4. Eredmények	21
2.5. CSG szűrés 3D-ben	25
2.5.1. A CSG fa meghatározása	26
2.5.2. Eredmények	27
3. Általános felületek geometriai távolságmezője	31
3.1. Felületi pontok választása	32
3.2. Parametrikus felület	33
3.2.1. Talppontkeresés	34
3.2.2. Proxiillesztés	34
3.3. Implicit felület	36

3.3.1. Talppontkeresés	37
3.3.2. Proxiillesztés	37
3.4. A mező kiértékelése	37
3.5. Eredmények	37
4. Összefoglalás	42
Köszönetnyilvánítás	43
Irodalomjegyzék	43

1. fejezet

Bevezetés

A számítógépes grafikában a térfogatok reprezentációjára és az ezeknek megfelelő felületek megjelenítésére számos módszert szoktak alkalmazni. Ezek közül az egyik legelterjedtebb implicit megoldás az előjeles távolságfüggvény (SDF) használata. Az SDF értéke a tér minden pontjában a felülettől vett távolság, kívül pozitív belül pedig negatív előjellel. Az SDF értékeket felhasználva sphere tracing algoritmussal megjeleníthető a geometria.

Összetett felületek esetén az SDF meghatározása bonyolult, zárt alakban pedig sokszor lehetetlen. Komplex geometriák esetén a már ismert SDF kiértékelése is nagy számításigényű, általános felülethez (például parametrikus, vagy implicit) pedig bonyolult kiszámítani a helyes távolságfüggvényt, ezért érdemes valamilyen közelítést használni. Erre alkalmazhatóak a geometriai távolságmezők, amelyek egyszerű geometriákkal közelítenek összetett felületeket.

1.1. Geometriai távolságmezők

A geometriai távolságmezők összetett alakzatot közelítenek olyan egyszerű geometriákkal amelyeknek ismerjük a távolságfüggvényét.

A mező geometriák halmazát tárolja amelyek adott felületi pontokban G^n geometriai folytonossággal csatlakoznak az eredeti felülethez.

Ez lényegében azt jelenti, hogy a felület egyes darabjait egyszerűbb geometriákkal helyettesítjük. A mezőt így a közelítési pontok pozíciói és az ezekhez meghatározott geometriák definiálják.

A legegyszerűbb megközelítés ha a mezőt egy ráccsal reprezentáljuk és minden cella középpontjához meghatározunk a legközelebbi felületi pontot a talppontot, valamint egy geometriát amely az adott cellában ad jó közelítést.

Ekkor a mező úgy értékelhető ki, hogy az adott cellában tárolt alakzat távolságfüggvényét kiszámítjuk, vagy több közeli cellából vett eredményen interpolációt végezhetünk.

Az egyik legegyszerűbb mező a nulladrendű mező, ahol a felületi mintaponthalmaz maga közelítő geometria és minden cellában a talppontot tároljuk.

1.2. Szakirodalmi áttekintés

A geometriai távolságmezők kérdésével már foglalkoztak korábbi kutatásokban, ahol bonyolult SDF-ek egyszerűbb közelítésére használták őket.

A [1] 2D-ben vizsgálta a problémát első és másodrendű mező esetében. Közelítő geometriaként első rendben az érintőegyenest, másodrendben a simulókört használták.

Tavalyi TDK dolgozatomban ezt az eredményt fejlesztettem tovább háromdimenziós alakzatokra, ahol síkot és tóruszt használtam közelítésre.

A háromdimenziós mezőket egy 3D textúrával reprezentáltam, ahol minden cella egy-egy közelítő geometriát tárolt, ami a cellához tartozó talppontban csatlakozott a felülethez.

A mezők szűréséhez egyszerű legközelebbi mintavételezést valamint trilineáris interpolációt alkalmaztam, ám a mérési eredményekből kiderült, hogy ez a megoldás nem illeszkedik a mezőhöz, mivel nem javított az approximáción. Ennek oka az, hogy az interpoláció nem veszi figyelembe az adat geometriai jelentését, így elrontja a magasabb rendű approximációt.

1.3. Elsőrendű mezők szűrése

A fent említett probléma megoldására elkezdtem kidolgozni egy olyan módszert amely megőrzi a mező geometriai jelentését. Ezt az elsőrendű mezőkre elkészített úgynevezett CSG szűréssel érem el, amely a tárolt geometriákat egyszerű halmazműveletekkel kombinálja.

Dolgozatomban a korábbi eredményekre támaszkodva bemutatom hogyan lehet elsőrendű távolságmezőt generálni 2 és 3 dimenzióban, valamint ismertetem a korábban használt szűrési módszerek hátrányait.

Bemutatom az általam kidolgozott szűrési módszert 2D-ben, amely a pontos approximáció mellett egy mintapontban képes egzaktul rekonstruálni 4 síkot ezáltal megtartani a hegyes sarkokat és éles éleket. Ezután megmutatom a módszer három dimenziós általánosítását.

1.4. Általános felületek távolságmezője

Az SDF-ekre triviális volt a távolságmező generálása, hiszen csak pontonkénti mintavételezést kellett használni. Az algoritmus kis módosításával, viszont a koncepció általánosítható egyéb felülettípusokra is.

Dolgozatom második felében parametrikus és implicit felületek távolságmezőjének generálását mutatom be. Kitérek a felületi mintapontok választásának különböző lehetőségeire a rács mellett, valamint bemutatom hogyan lehet a geometriákat definiálni különböző bemeneti adatokból.

Ezzel megadok egy általános felületreprezentációt amely átlános bementre egy egy-séges leírást szolgáltat, amelyet bemenettől függetlenül lehet kiértékelni és megfelelő szű-réssel jó approximációját adja az SDF-nek.

2. fejezet

Elsőrendű távolságmező szűrése

Tavalyi dolgozatomban SDFből generált geometriai távolságmezőkkel foglalkoztam. Komplex alakzatok ismert távolságmezőjének kiértékelése költséges lehet, ezért ebben az esetben is érdemes diszkrét közelítést alkalmazni.

A következő fejezetben bemutatom az elsőrendű távolságmező generálását SDF inputhoz a mező koncepciójához illő szűrési módszerrel.

2.1. Távolságfüggvények

A távolságmezők egy előjeles távolságfüggvény approximációját adják meg.

1. Definíció. Az $f : \mathbb{R}^N \rightarrow \mathbb{R}$ függvény az $F \subset \mathbb{R}^n$ előjel nélküli távolságfüggvénye, ha

$$f(\mathbf{p}) = d(\mathbf{p}, \partial F), \text{ ahol } d(\mathbf{x}, G) = \inf_{\mathbf{y} \in G} \|\mathbf{x} - \mathbf{y}\|_2 \text{ és } \partial F := \overline{F} \cap \overline{(\mathbb{R}^n \setminus F)}$$

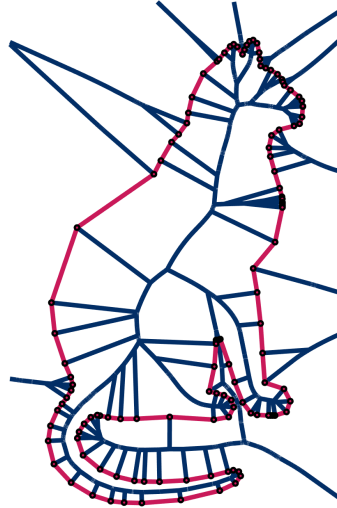
2. Definíció. Az $f : \mathbb{R}^N \rightarrow \mathbb{R}$ előjeles távolságfüggvény, ha folytonos és $|f|$ távolságfüggvény.

3. Definíció. X topologikus tér $S \subseteq X$ részhalmazának ∂S határán azokat a pontokat értjük, amelyek elemei S lezártjának, de nem belső pontjai S -nek, azaz

$$\partial S = \overline{S} \setminus \text{int}_X S$$

1. Tétel. Az f pontosan akkor SDF, ha létezik $\emptyset \neq D \subseteq \mathbb{R}^N$ halmaz amivel

$$f(\mathbf{p}) = \begin{cases} d(\mathbf{p}, \partial D) & \text{ha } \mathbf{p} \notin D \\ -d(\mathbf{p}, \partial D) & \text{ha } \mathbf{p} \in D \end{cases}$$



2.1. ábra. Síkbeli alakzat (piros) cut locus-a (kék)

Azaz az előjel meghatározza térnek egy particionálását, a negatív előjel a D alakzat belseje, a pozitív a külseje, $f(\mathbf{p}) = 0$ pedig maga a felület.

4. Definíció. $\mathbf{x} \in \mathbb{R}^n$ ponthoz tartozó talppontnak nevezzük azon felületi pontokat amelyek $f(\mathbf{x})$ távolságra vannak $\mathbf{x}$ -től, azaz a legközelebbi felületi pontokat.

5. Definíció. A cut locus azon $\mathbf{x} \in \mathbb{R}^n$ pontok halmaza amelyekhez több talppont is tartozik. Egy példa látható a 2.1. ábrán.

6. Definíció. Két felület az $\mathbf{x}$ közös pontjukban G^n geometriai folytonossággal csatlakozik, ha létezik a két felületnek olyan parametrikus megadása, amely C^n folytonossággal csatlakozik $\mathbf{x}$ -ben.

Például G1 folytonos a csatlakozás görbe és az adott pontban vett érintőegyenese között, G2 folytonos a görbe és a simulóköre között.

2. Tétel. Adott SDF n -edik deriváltja csak a $\mathbf{x}$ kiértékelési pozícióban csak az $f(\mathbf{x})$ távolságtól és az $\mathbf{x}$ -hez tartozó talppontbeli differenciálgeometriai értékektől függ.

Ebből következik hogy egy G^n geometriai folytonosságú felületnek C^n folytonos távolságmezője van.

2.2. A távolságmező generálása

2.2.1. Távolságmező reprezentálása

A távolságmező olyan geometriák halmaza amelyek adott felületi pontokban G^n folytonossággal csatlakoznak az eredeti felülethez. A pontokban lényegében ezekkel a geo-

metriai proxikkal helyettesítjük az eredeti felületet, ezért azt is elvárjuk, hogy a geometriáknak egyszerű legyen a távolságfüggvénye.

Így a két fő kérdés:

1. hogyan válasszuk a felületi pontokat?
2. milyen geometriákat használjunk?

A felületi pontok kiválasztására több lehetőség is van, amelyeket a dolgozat második felében a mező általánosításakor ismertetek részletesen. Az SDF esetén a legegyszerűbb lehetőség egy síkbeli vagy térbeli rács használata. A rács minden cellájához szeretnénk egy-egy proxit meghatározni amely az adott területen jó közelítést ad. Ehhez a rács minden cellájának középpontjához válasszuk a hozzá tartozó talppontot felületi mintapontnak és itt illesszünk egy geometriát.

A geometriai folytonosság értelmében az illesztett proxy a talppont egy kis környezetben jó közelítését fogja adni a felületnek így feltehető hogy a cellában amelyhez a talppontot számoltuk a távolságfüggvény jó approximációját kapjuk a 2. Tétel alapján.

A rácsként meghatározott mezőt a következő rendtől független adatok definiálják:

- a felbontása ($res \in \mathbb{N}$, hány cellából álljon a tömb egy dimenzió mentén)
- a befoglaló doboz oldalhossza ($box \in \mathbb{R}$, a közelítendő objektum egy origó középpontú négyzetben/ kockában helyezkedik el, ezen belül generáljuk a mezőt)

Feltehetjük, hogy minden objektum a $[-1, 1]^2$ négyzetben (3d-ben az ekvivalens kockában) helyezkedik el, ezért a box értéket alapértelmezetten 2-nek veszem, az ábrák és mérések ezzel az értékkel készültek.

Ezzel a megközelítéssel a mező könnyen tárolható egy res^2 vagy res^3 felbontású textúrában, ahol az egyes texelek a geometriák definiáló adatait tárolják. A mező kiértékelésénél a textúra hatékonyan olvasható.

2.2.2. Proxi geometria

A következő kérdés, hogy adott $N \in \{1, 2\}$ dimenzióban az elsőrendű mezőhöz milyen proxi geometriát válasszunk.

A proxitól elvárjuk, hogy elsőrendben csatlakozzon a felülethez a talppontban. Ez a gyakorlatban azt jelenti, hogy helyesen tárolja a talppont pozícióját és az ott vett normálvektort. Emellett szeretnénk hogy a távolságfüggvény egyszerűen számolható legyen.

Egy megfelelő alakzat a talppontban vett érintőegyenes, vagy érintősík. A síkot a $p \in \mathbb{R}^N$ talppont és az $n \in \mathbb{R}^N$ normális meghatározza.

Adott cellabeli geometria meghatározásához először a cella középpontját kell kiszámítani, majd ehhez megkeresni a talppontot.

A cellák középpontjai: $\forall i \in \{0 \dots res - 1\}^N$:

$$\mathbf{grid}(\mathbf{i}) = -\frac{box}{2} + (\mathbf{i} + 0.5) \cdot \frac{box}{res} \quad (2.1)$$

A felület az $f : \mathbb{R}^N \rightarrow \mathbb{R}$ SDF-el van megadva. Az

7. Definíció. $f : \mathbb{R}^N \rightarrow \mathbb{R}$ függvény gradiense:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

A gradiens azt az irányt adja meg, amerre elmozdulva függvényérték a legtöbbet változik. Ez SDF esetében párhuzamos a talppontban vett normálissal, hiszen egy adott felületi pontból indulva a merőleges irányába elmozdulva nő a legtöbbet a felülettől vett távolság. Így a talppontba eljuthatunk ha a gradienssel ellentétes irányba lépünk, az SDF értéke pedig megadja a lépés hosszát.

A derivált pontos kiértékelése sok esetben költséges lenne, de adott $\mathbf{x} \in \mathbb{R}^N$ pontbeli gradiens egyszerűen közelíthető centrális differenciákat használva:

$$\partial_x f(\mathbf{x}) = \frac{f(\mathbf{x} + \varepsilon) - f(\mathbf{x} - \varepsilon)}{2\varepsilon}$$

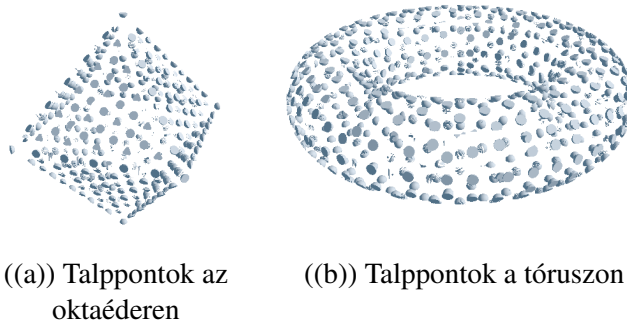
ahol $0 < \varepsilon \in \mathbb{R}$ egy kellően kicsi pozitív szám.

Ezt felhasználva a ponthoz tartozó talppont:

$$\mathbf{fp}(\mathbf{x}) = \mathbf{x} - f(\mathbf{x}) \cdot \frac{\nabla f}{\|\nabla f\|} \quad (2.2)$$

A robusztus eredmény érdekében a lépést iteráljuk néhányszor, amíg a gradiens hossza kellően nagy, így megkapjuk a talppontot. A normálvektor az itt vett gradiens. A talált talppontokra látható egy 3D-s példa a 2.2. ábrán.

A mező eltárolható tehát egy textúrában, ahol minden texel a proxit definíció talppontot és normálvektort tartalmazza. Ez 2D-ben 4, 3D-ben 5 skalár lenne. A normálvektor



2.2. ábra. A mező egy 3D rácsként van definiálva, ahol először minden cella középpontjához megkeressük a legközelebbi felületi pontot. Ezekre a pontok látható itt példa oktaéderről és tóruszról 32^3 felbontású mező esetén

hosszáról feltehetjük, hogy 1, így a normálvektor leírásánál valójában csak 2 szabadságfokunk van. Ilyen megadás például a polárkoordináta, ahol a vektor irányát 2 szöggel tudjuk megadni.

A tárigény tovább csökkenthető, ha kihasználjuk hogy a talppontból a cellaközéppontba mutató vektor párhuzamos a normálissal. A cella középpontját a mező kiértékelésekor is meg tudjuk határozni, ha tudjuk hogy kiértékelési pontunk melyik cellába esik, így elég csak a talppontot tárolni, ami 2 illetve 3 skalárt jelent.

Mivel az eredeti alakzatnak van kívül-belül felosztása, ezért a geometriai proxiknak is kell ezt közelítenie, azaz egyenes és sík helyett félsíkot és félteret kell értelmezni.

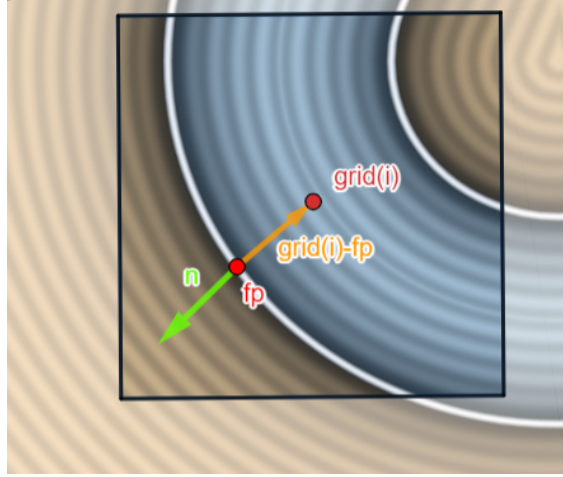
Ez egyszerűen megtehető a normálvektor alapján. SDF-ek estében feltehetjük hogy a felületi pontokbann vett gradiensek mindig kifelé azaz a pozitív oldalra mutatnak, tehát a geometria külseje az az oldal amerre a normális mutat, a másik pedig a belseje.

Így viszont, ha a cellánk középpontja az alakzaton belül volt akkor a talppontból a cellaközéppontba mutató vektor a valódi normálissal ellentétes irányú lesz, ahogy a 2.3. ábrán is látható.

Ennek megjavítására még egy s skalárt tárolnunk kell, amely 1 vagy -1 értéket vesz fel aszerint, hogy a talppontban vett gradiens iránya azonos-e a talppontból a cellaközéppontba mutató vektorával, így a kiértékelésnél a normálvektor a helyes irányban írható fel.

Így tehát a geometria az fp talpponttal és s irányskalárral leírható, amely 2Dben 3, 3Dben 4 skalárt jelent.

Összegezve az elsőrendű mező generálásának algoritmus 2 dimenzióban:



2.3. ábra. A talppontból a cella középpontjába mutató vektor ($grid(i) - fp$) párhuzamos az n normálvektorral, ám az iránya nem feltétlen azonos. Amikor a cellaközéppont az alakzat belsejébe esik a normálissal ellentétes irányú vektort kapunk.

1. Algoritmus Elsőrendű mező generálása N dimenzióban

```

for  $i_1 = -res \dots res$  do                                ▷ A mező celláiban tárolt adatok számolása
  ⋮
  for  $i_N = -res \dots res$  do
     $grid(i_1, \dots, i_N) = -box/2 + \begin{bmatrix} i_1 + 0.5 \\ \vdots \\ i_N + 0.5 \end{bmatrix} \cdot \frac{box}{res}$           ▷ A cella középpontja
     $fp = grid(i_1, \dots, i_N) - f(grid(i_1, \dots, i_N)) \cdot \frac{\nabla f(grid(i_1, \dots, i_N))}{\|\nabla f(grid(i_1, \dots, i_N))\|}$   ▷ Talppont
     $n = \frac{\nabla f(fp)}{\|\nabla f(fp)\|}$                                 ▷ Normálvektor a talppontban
     $s = \begin{cases} 1 & \text{ha } \langle n, grid(i_1, \dots, i_N) - fp \rangle > 0 \\ -1 & \text{ha } \langle n, grid(i_1, \dots, i_N) - fp \rangle < 0 \end{cases}$           ▷ Irányskalár
  end for
  ⋮
end for

```

A generálás egy előzetes lépésként futtatható, amelyet a geometria megjelenítése előtt egyszer kell elvégezni. A tárolt proxik utána futásidőben tesztölgesen kiértékelhetők.

2.3. A mező kiértékelése

A mezőt egy 2- vagy 3D textúrában tároljuk el. A kiértékeléskor az x pozíció alapján a legközelebbi cellából kiolvasom a proxit definiáló adatokat. A gyakorlatban ez egy egyszerű hardveresen támogatott textúra olvasást jelent, nearest sampling-el.

A cellából kiolvashatjuk az $\mathbf{fp} \in \mathbb{R}^N$ talppont értéket és az $s \in \{-1, 1\}$ skalárt amelyből meghatározható a normálvektor.

Ehhez a cella $\mathbf{i}$ indexéből ki kell számítani a középpontot a 2.1 képlettel. A normálvektor a talppontból a cellaközéppontba mutató vektorral párhuzamos és az s irányskalár szerint van irányítva:

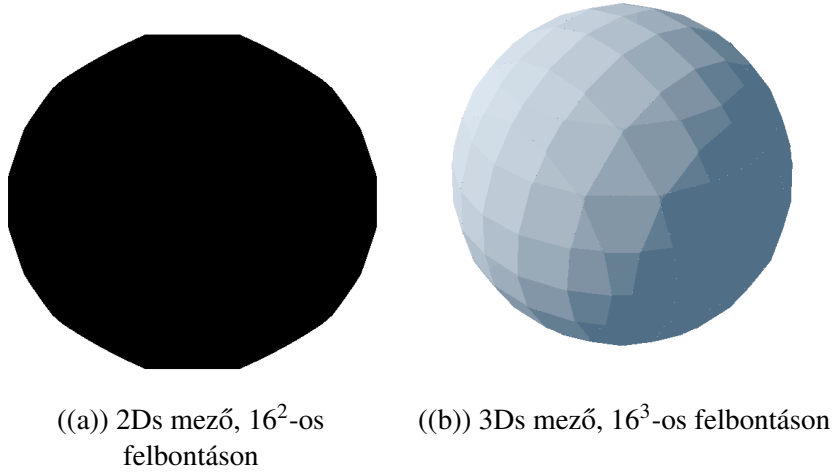
$$\mathbf{n} = s \cdot (\mathbf{grid}(\mathbf{i}) - \mathbf{fp}) \quad (2.3)$$

A kapott adatokkal a félsík vagy féltér előjeles távolságfüggvénye használható.

A távolságfüggvény lényegében azonos, az adatok dimezióját leszámítva. Adott $x \in \mathbb{R}^N$ pont előjeles távolsága a mezőből olvasott geometriától:

$$d(\mathbf{x}) = \langle \mathbf{n}, \mathbf{x} - \mathbf{fp} \rangle \quad (2.4)$$

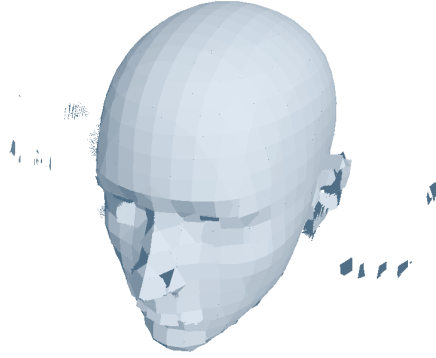
Kétdimenziós esetben a mező egyszerűen megjeleníthető a pixelek távolságfüggvény előjele szerinti színezésével. Három dimenzióban a sphere tracing algoritmus használható [2]. A megjelenített mezőre példák láthatók a 2.4. ábrán.



2.4. ábra. Példák a mező megjelenítésére

A mező koncepciójából adódóan a proxik csak a talppont közelében adnak helyes közelítést. Ha a kiértékelési pontunk messze van a hozzá tartozó talpponttól hibás távolságértéket kaphatunk. Ez inkább 3Dben okoz problémát, mivel a spheretracing algoritmus viselkedéséből adódóan hibás objektumok jelenhetnek meg, ahogy a 2.5. ábra is mutatja.

Ez javítható, ha a geometriát lekorlátozzuk a talppont kis környezetére. Erre megfelelő megoldás, ha a talppont körül egy körrel vagy gömbbel metszük a geometriát. SDF-ek



2.5. ábra. A geometriák csak a talppont közelében adnak jó közelítést, így ha a mintapontunk messze van a talpponttól, hibás lebegő objektumok jelenhetnek meg

esetében ez a két távolságfüggvény maximumát jelenti, azaz r sugarú kör/gömb esetén a korlátozott geometria:

$$d_{constrained}(\mathbf{x}) = \max(d(\mathbf{x}), \sqrt{\|\mathbf{x} - \mathbf{fp}\|_2} - r) \quad (2.5)$$

Az r sugár értékére több kísérletet is végeztünk, például kis skalárokkal, de a legjobbnak az bizonyult amikor a cella méretének, azaz $\frac{box}{res}$ -nek választottuk.

Simább eredmény érdekében bilineáris illetve trilineáris interpolációt alkalmazhatunk.

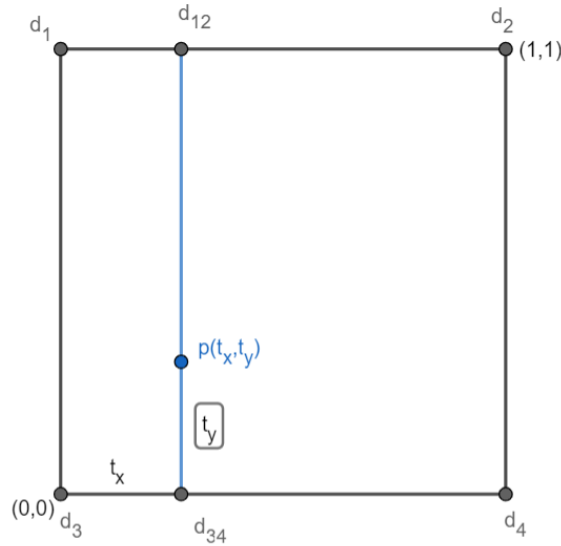
A 2D-s esetben a legközelebbi 4 texelből kiolvasott geometriát kellene interpolálni. Mivel a paraméterek nem interpolálhatók ezért először a 4 geometriától vett távolságot kell meghatározni és a kapott távolságokon interpolálni. Az interpoláció a kiértékelési pozíció alapján történik, a 2.6 ábrán mutatott módon.

3D-ben ugyanezt kell elvégezni, csak a legközelebbi 8 texelből olvasott proxikkal.

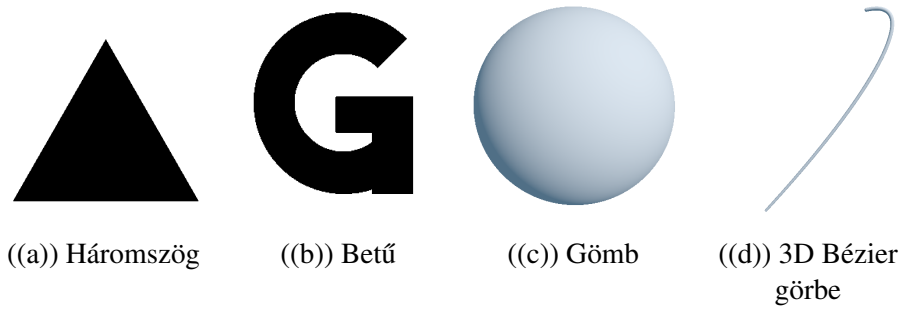
2.3.1. Eredmények

A mező helyességét mérésekkel vizsgáltam. Vettem egy nagy felbontású finom rácsot és minden rácspontban kiértékeltem az eredeti SDF-et valamint az adott felbontású elsőrendű mezőből kapott eredményt. A két érték különbsége lett a hiba. A hibavektort $\|\cdot\|_1$, $\|\cdot\|_2$ és $\|\cdot\|_\infty$ normákban értékeltem ki.

Összehasonlításként a nulladrendű mezőt (A0) is belevettem a mérésbe. Itt minden cella csak a középpontban vett SDF értéket tárolja. A mezőt bilineáris és trilineáris szűréssel lehet kiértékelni.



2.6. ábra. Bilineáris interpoláció



2.7. ábra. A méréshez használt SDF-ek. Az SDF-ek forrása a [3] és [4]

A méréseket a 2.7 ábrán látható két és három dimenziós alakzatokon végeztem. Az eredmények a 2.1 táblázatban olvashatók.

Az eredményből látható, hogy a mező jó közelítést ad már alacsony felbontáson is. Az átlaghiba és a kettes norma mindig alacsony érték, egyedül a maximum hiba lehet nagy szűretlen mezőnél. Erre az ötletem az volt, hogy a talppont numerikus hibái okozzák, amit később kísérlettel validáltam, ez a dolgozat második felében olvasható.

Az is megfigyelhető, hogy a mezők minősége nem javul, sőt néhány esetben romlik a trilineáris szűrés hatására még ezeken a viszonylag egyszerűbb alakzatokon is. Ennek oka az lehet hogy a szűrést a távolságokon végezzük így elhagyjuk az adat geometriai jelentését.

Ennek megoldására először 2D-ben kidolgoztam egy módszert, amely geometriai műveletekkel kombinálja proxikat.

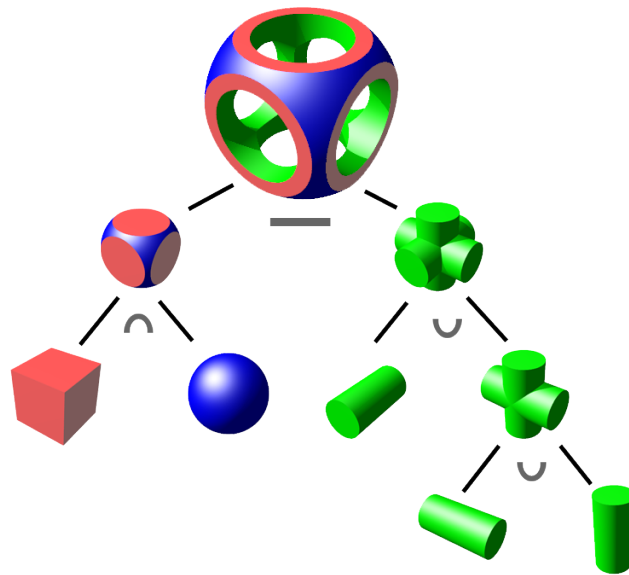
Háromszög				
Res.	Norm	A0	G1	
			nearest	bilinear
16 ²	1	0.0053	0.00097	0.0020
	2	0.013	0.0046	0.0047
	∞	0.087	0.14	0.042
32 ²	1	0.0014	0.00036	0.00061
	2	0.0045	0.0016	0.0016
	∞	0.043	0.068	0.021
64 ²	1	0.00042	0.0002	0.00027
	2	0.0016	0.00058	0.00055
	∞	0.021	0.035	0.011
Betű				
Res.	Norm	A0	G1	
			nearest	bilinear
16	1	0.0065	0.0019	0.0032
	2	0.014	0.007	0.0068
	∞	0.082	0.11	0.057
32	1	0.0017	0.00059	0.00095
	2	0.0047	0.0031	0.0026
	∞	0.038	0.073	0.029
64	1	0.00052	0.00019	0.00026
	2	0.0017	0.0011	0.00088
	∞	0.016	0.034	0.014
Gömb				
Res.	Norm	A0	G1	
			nearest	trilinear
16	1	0.0069	0.0014	0.0032
	2	0.013	0.0021	0.0037
	∞	0.11	0.051	0.032
32	1	0.0017	0.00055	0.00092
	2	0.0045	0.00066	0.001
	∞	0.054	0.023	0.015
64	1	0.00046	0.00026	0.00034
	2	0.0015	0.00033	0.00037
	∞	0.027	0.011	0.0077
Bézier görbe				
Res.	Norm	A0	G1	
			nearest	trilinear
16	1	0.017	0.002	0.0034
	2	0.054	0.0033	0.0043
	∞	0.64	0.086	0.034
32	1	0.0075	0.00088	0.0012
	2	0.037	0.0013	0.0015
	∞	0.58	0.044	0.022
64	1	0.0035	0.00059	0.00068
	2	0.025	0.00072	0.00075
	∞	0.5	0.021	0.021

2.1. táblázat. Pontosság mérés a távolságmezőkre

2.4. CSG szűrés 2D-ben

A CSG műveletek geometriákon végzett halmazműveletek, ilyen a metszet, unió és különbség. SDF-ek esetében ezek megfelelnek a minimum és maximum műveleteknek, valamint ezek kombinációjának.

A CSG fa egy olyan bináris gráf amelynek leveleiben elemi geometriák vannak, belső csúcsaiban pedig CSG műveletek. A fa kiértékelése során az elemi geometriákat az adott műveletekkel kombináljuk és a fa gyökerébe érve egy összetett alakzatot kapunk eredményül. A 2.8 ábrán látható egy példa egy általános CSG fára.



2.8. ábra. A CSG fa egy olyan bináris gráf amely a leveleiben lévő geometriákat a belső csúcsaiban lévő műveletekkel kombinálja

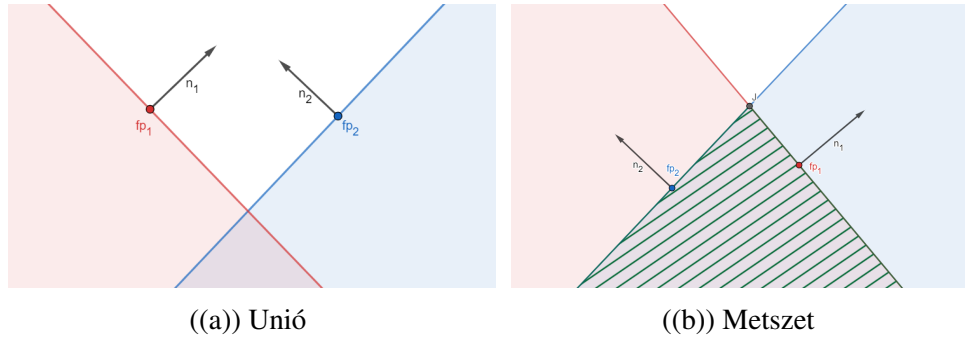
A CSG szűrés ötlete azon alapul, hogy a kiértékelési ponthoz legközelebbi geometriát CSG műveletek segítségével úgy tudjuk kombinálni hogy egy pontosabb közelítést kapjunk.

Ez azt jelenti hogy a kiértékelési hely körüli 4 félsíkot egy CSG fába szervezzük, amelyben metszet és unió műveletek találhatók.

2.4.1. CSG művelet két geometriára

A CSG fa meghatározásához először azt kell eldöntenünk hogy 2 ismert félsíkra milyen műveletet kell alkalmaznunk.

Adott A és B félsík, amelyek definiáló talppontjai $\mathbf{a} \in \mathbb{R}^2$ és $\mathbf{b} \in \mathbb{R}^2$. Feltehetjük hogy mindkét talppont az eredeti alakzat határpontja volt. Ez alapján ha a két félsík tartalmazza



2.9. ábra. Adott két félsíkról szeretnénk eldönteni milyen művelettel lehetne őket kombinálni. A talppontok biztos az eredeti felület határán vannak így ha a félsíkok tartalmazzák egymás talppontját metszetet, ha nem akkor uniót kell venni.

egymás talppontját akkor a metszet adja a helyes közelítést, ha nem tartalmazzák akkor az unió. A két eset látható a 2.9 ábrán.

Amikor az egyik sík tartalmazza a másik talppontját de ez a tartalmazás nem kölcsönös az unió műveletet választottam. Ez az eset általában akkor állt fenn amikor a mező felbontása kicsi volt az eredeti alakzat részletességéhez képest, vagy numerikus hibák léptek fel a talppont számolásakor.

A különböző esetek összefoglalása a 2.2 táblázatban látható.

Feltételek	Művelet
$a \in B \wedge b \in A$	$\cap$ metszet
$a \notin B \wedge b \notin A$	$\cup$ unió
$a \in B \wedge b \notin A$	$\cup$ unió

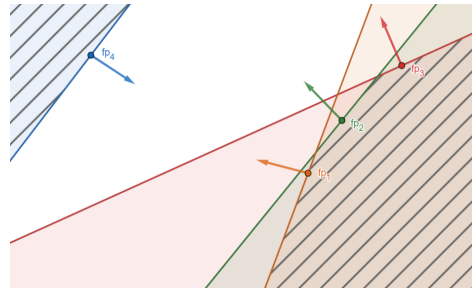
2.2. táblázat. A talppontok tartalmazása szerint választható műveletek, a és b talpponttal definiált A és B félsíkok esetén.

2.4.2. CSG fa meghatározása

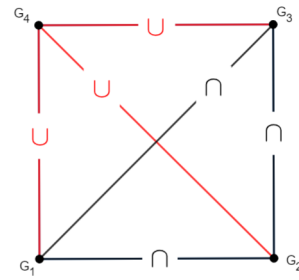
Miután két geometria között kiválasztottuk a műveletet, kombináljuk a 4 geometriát egy CSG fává. Először is minden geometria párra határozzuk meg a köztük szükséges műveletet. Ezeket a műveleteket egy 4 csúcsú 6 élű gráfban tudjuk ábrázolni, mint ami a 2.10 ábrán is látható.

Ennek a gráfnak szimmetriákat és forgatásokat leszámítva 11 lehetséges esete van, amelyekhez egyértelműen meg lehet adni a megfelelő CSG fát. A műveletek száma és pozíciója egyértelműen meghatároz egy fát.

A 2.3. táblázatban láthatóak a műveleti gráfok és a hozzájuk tartozó CSG fa, valamint egy példa, amely az adott állapotot eredményezi.



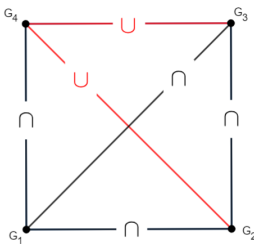
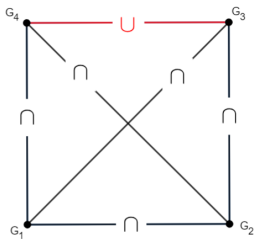
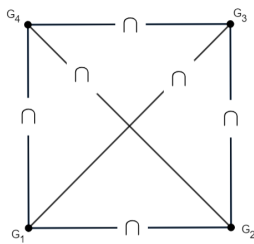
((a)) A 4 félsík elhelyezkedése



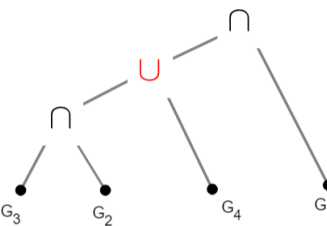
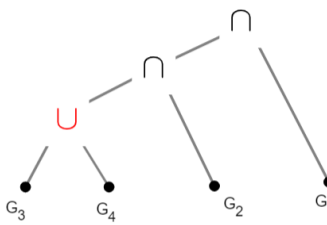
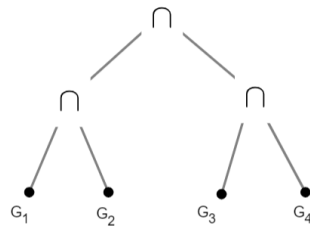
((b)) A hozzá tartozó műveleti gráf

2.10. ábra. A bal oldali ábrán a 4 félsík helyzete látható, a sátrózott rész jelöli a jobb közelítést amit kapni szeretnénk. A félsíkok között páronként megállapíthatjuk a szükséges műveletet, amit egy 4 csúcsú 6 élű gráfba tudunk felírni. Itt a csúcsok a geometriákat az élek a műveleteket jelképezik. Ebből a gráfból lehet felírni a CSG fát.

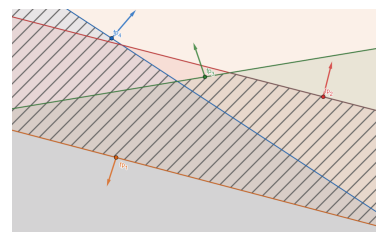
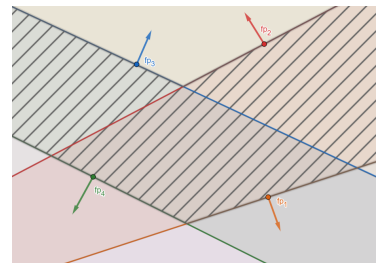
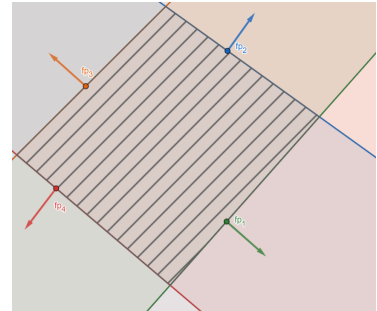
Műveletek



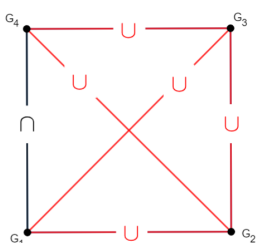
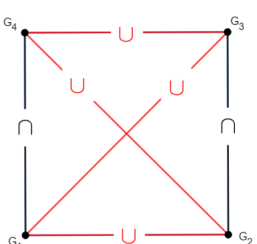
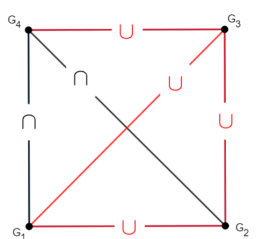
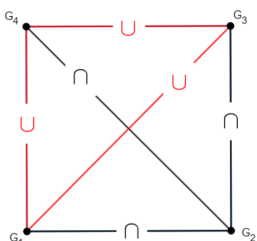
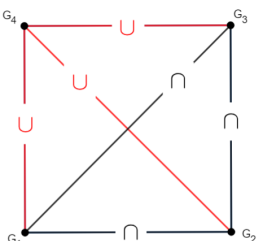
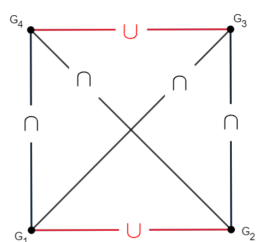
CSG fa



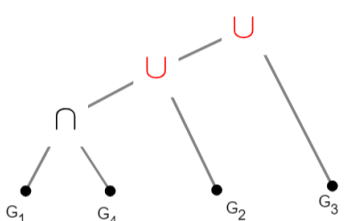
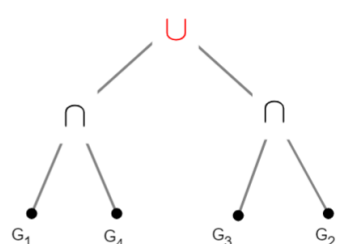
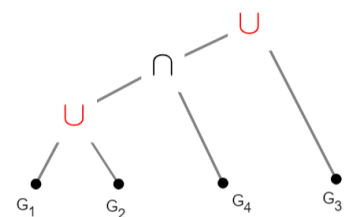
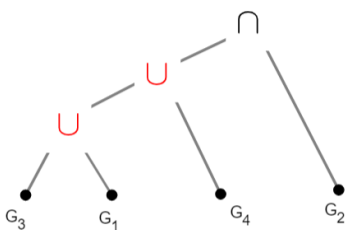
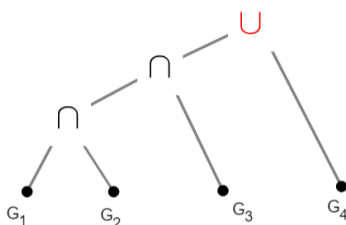
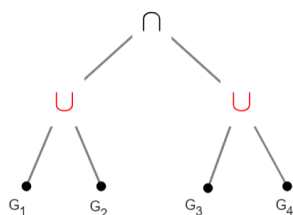
Példa



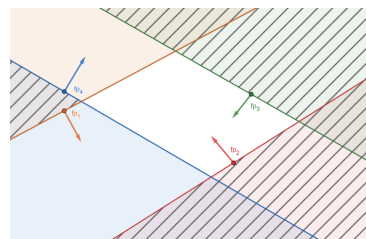
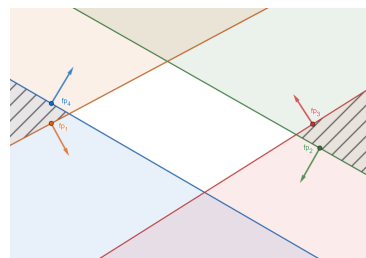
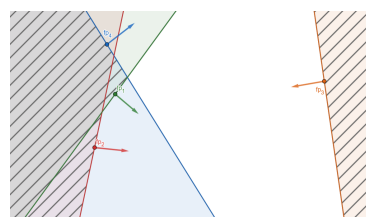
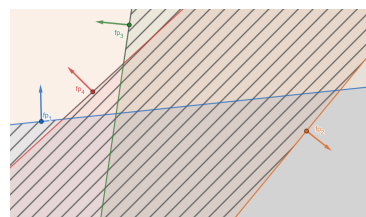
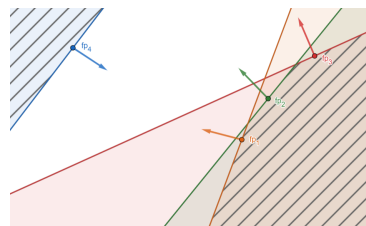
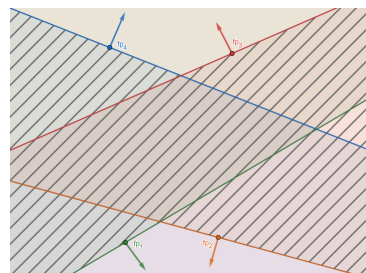
Műveletek

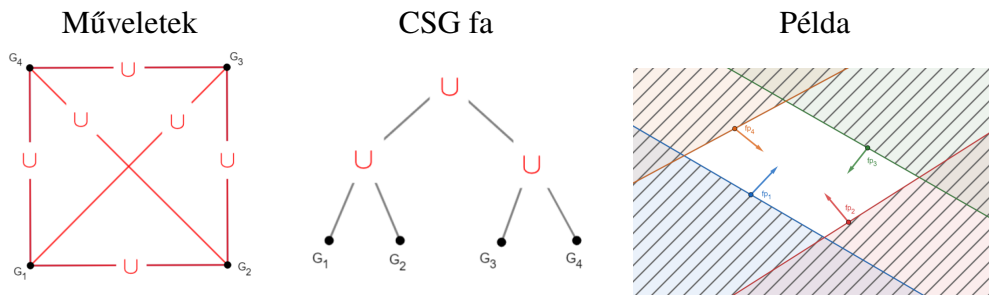


CSG fa

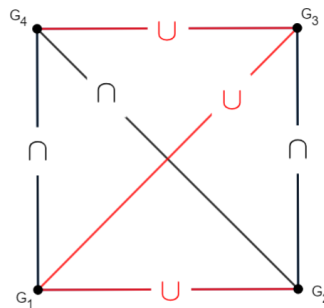


Példa





2.3. táblázat. A CSG szűrés lehetséges esetei. Az első oszlopban látható gráf a geometriák közötti páronkénti műveletekkel, majd mellette a megfelelő CSG fa. Az utolsó oszlopban egy példa látható az adott esetre, ahol a satírozott rész jelöli a jobb közelítést.



2.11. ábra. Egy szélsőséges eset, amihez nem sikerült megfelelő CSG fát felírni. A tapasztalatok szerint ritkán fordul elő és főleg apró hibák okozzák.

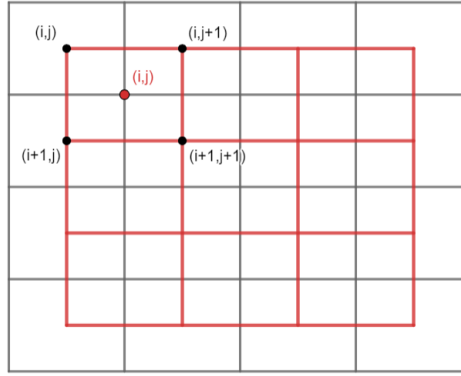
A 2.11. ábrán látható egy 11. eset, amely nem szerepelt a táblázatban, mivel ehhez nem sikerült megfelelő CSG fát felírni. Ez egy ritka eset, amely a tapasztalatok szerint numerikus hibák következtében jön elő, ezért itt a CSG műveletek helyett bilineáris szűrést alkalmaztam.

2.4.3. CSG fák generálása

A CSG fa a kiértékelés közben, futásidőben meghatározható, ám ez lassítja a programot. Ennek javítására kidolgoztam egy módszert a CSG fa előzetes generálására.

Észrevehetjük, hogy a kiértékelési pontok egy rácsot alkotnak, ahol egy cellán belüli pontokhoz ugyanaz a CSG fa tartozik, mivel ugyanaz a 4 legközelebbi tárolt geometria. Ez a rács $res - 1$ felbontású és a geometriákat tároló rácshoz képest fél cellával el van csúsztatva a 2.12 ábrán látható módon.

Ezt a rácsot szintén legenerálhatjuk előre, majd a CSG fákat egy 2D textúrában eltárolhatjuk. A mező olvasásakor csak a tárolt CSG fát kell kiértékelni.



2.12. ábra. A kiértékelési pontokból alkotható egy rács, ahol egy cellán belül azonos CSG fa tartozik a pontokhoz. Ez az ábrán pirossal jelzett rács $res - 1$ felbontású és a geometriákat tároló az ábrán fekete rácshoz képest fél cellával el van csúsztatva. Így a CSG rács (i,j) indexű cellájában tárolt CSG fához a távolságmező $(i,j), (i+1,j), (i,j+1), (i+1,j+1)$ celláiban található geometriák kellenek.

Miután legeneráltuk a mezőt, végigiterálunk a CSG fákat tartalmazó $(res - 1) \times (res - 1)$ felbontású rácson. Az (i,j) helyre számolandó CSG fa a mező $(i,j), (i+1,j), (i,j+1), (i+1,j+1)$ celláiban található geometriákból épül fel.

A négy geometriából a fent leírt módon kiszámítjuk a fát, majd eltároljuk.

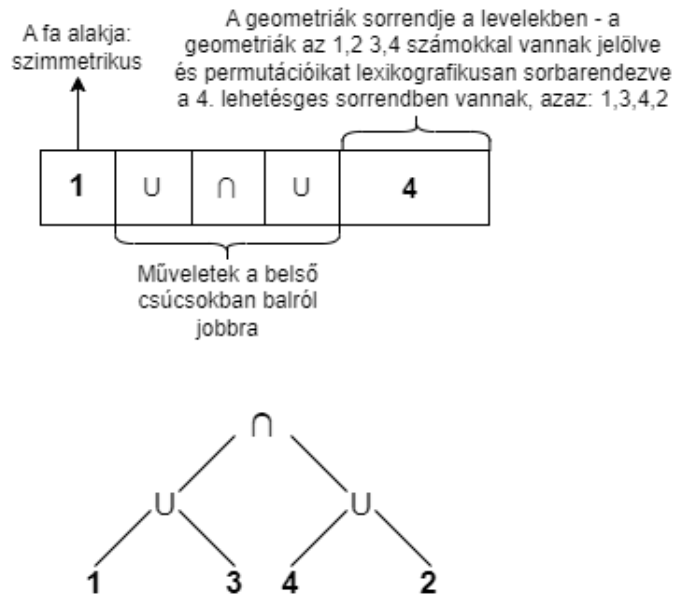
A fa tárolásához 4 skalár értéket használunk:

- a fa alakja: a 2.3. táblázatban látható hogy a fa alakja 2 féle lehet, így egy bináris értékkel tárolhatjuk hogy néz ki
- a 3 művelet a belső csúcsokban: 3 bináris értékkel balról jobbra haladva leírhatjuk a használt műveleteket
- a geometriák sorrendje a levelekben: a levelekben a 4 geometria található valamilyen sorrendben. Ez a sorrend egy értékkel leírható ha az 1,2,3,4 permutációit lexikografikusan megszámozzuk és a szükséges permutáció sorszámát tároljuk el, ami egy 1 és 24 közötti érték

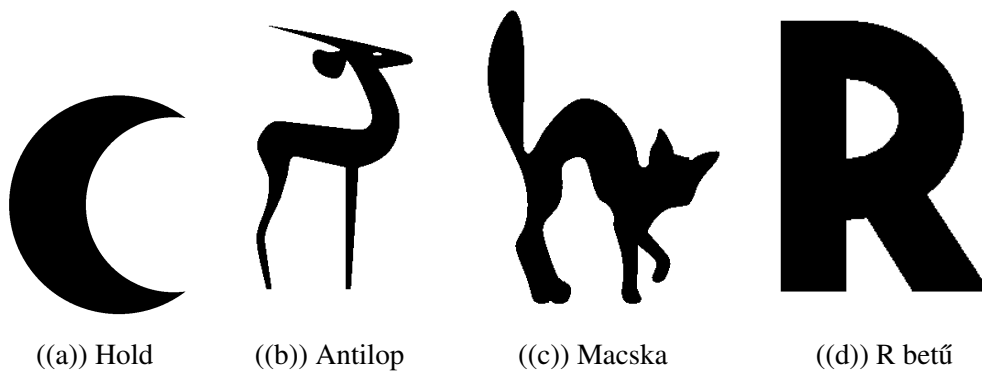
A 2.13. ábrán látható egy példa egy adott fát leíró számsorozatra és annak értelmezésére.

2.4.4. Eredmények

A CSG szűrés helyességének vizsgálatára a korábbi mérésekhez hasonló kísérleteket végeztem, ahol egy 4099^2 felbontású sűrű rácson kiértékeltem az eredeti SDF-et és a



2.13. ábra. Példa a CSG fa tárolására. A felső részen látható a tárolt adat, alul pedig a rekonstruált CSG fa.



2.14. ábra. A méréshez használt SDF-ek. Az alakzatok forrása [3]

generált mezőt 0 és 1. rendben, különböző szűrésekkel. A hiba az értékek különbsége, amelyet aztán különböző normákban vizsgáltam a fent leírt módon.

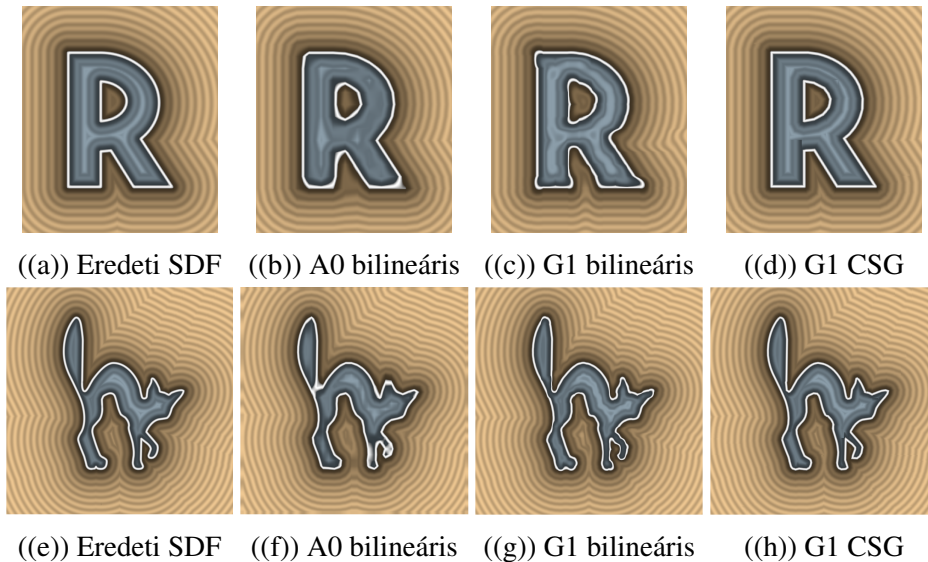
A mérésekre a 2.14. ábrán látható alakzatokat használtam, az eredmények a 2.4 táblázatban láthatók.

Az eredmények azt mutatják, hogy a CSG szűrés jelentősen javít a mező minőségén már alacsony felbontás mellett is. Emellett nagy előnye, hogy más szűrési módszerekkel ellentétben megtartja a hegyes sarkokat valamint a pontos cut locust, ahogy a 2.15. ábrán látható.

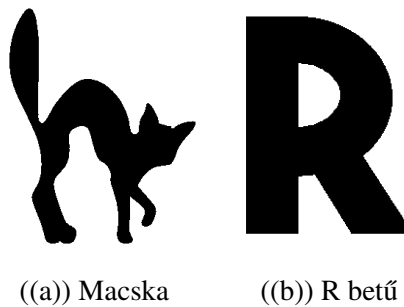
Az eredményt összehasonlítottam a [6]-ban leírt multichannel SDF-el, amely szintén képes az éles sarkok megőrzésére. A módszer az enyémtől eltérő SDF koncepciót használ, amely azon alapul, hogy a sarkok körül átalakítja az SDF értékeket majd ehhez

Hold					
Res.	Norm	A0	G1		
			nearest	bilinear	CSG
16	1	0.0079	0.0022	0.0044	0.00053
	2	0.015	0.0087	0.0083	0.00089
	∞	0.082	0.15	0.05	0.017
32	1	0.0021	0.0006	0.0011	0.00027
	2	0.0051	0.0029	0.0029	0.00039
	∞	0.04	0.072	0.025	0.0086
64	1	0.00053	0.0003	0.00037	0.00026
	2	0.0018	0.0011	0.0011	0.00031
	∞	0.019	0.03	0.013	0.0032
128	1	0.00019	0.00025	0.00025	0.00025
	2	0.0006	0.00044	0.00042	0.00027
	∞	0.0086	0.013	0.0065	0.0034
Antilop					
Res.	Norm	A0	G1		
			nearest	bilinear	CSG
16 ²	1	0.0077	0.0032	0.0048	0.002
	2	0.014	0.0097	0.0091	0.0076
	∞	0.34	0.4	0.39	0.4
32 ²	1	0.0021	0.00094	0.0015	0.00052
	2	0.0053	0.0038	0.0036	0.0016
	∞	0.37	0.4	0.4	0.4
64 ²	1	0.00057	0.00036	0.00051	0.00029
	2	0.0022	0.0017	0.0017	0.0012
	∞	0.38	0.4	0.4	0.4
128 ²	1	0.00025	0.00021	0.00026	0.00025
	2	0.0013	0.0012	0.0012	0.0012
	∞	0.39	0.4	0.4	0.4

2.4. táblázat. A mezők pontossága CSG szűréssel és egyéb módszerekkel.



2.15. ábra. A kiértékelt mező összehasonlítása különböző típusú 32^2 felbontású mezőkre, bilineáris és CSG szűréssel A színezés forrása: [5].



2.16. ábra. A CSG szűrés és a multichannel SDF [6] pontosságának összehasonlításához használt ábrák. Az alakzatok forrása [3]

generál egy többcsatornás távolságmezőt. Mivel az SDF átalakítása feltétlenül szükséges a módszer működéséhez, így nem tudtam a közvetlen számértékeket hasonlítani az én eredményeimhez. Ehelyett egy-egy fekete-fehér képet generáltam az eredeti SDFből, az én CSG szűrt megoldásomból valamint a multichannel SDF-el kapott eredményből, azonos felbontáson. Ezeken a képeken a negatív SDF értékek feketék a pozitívak fehérek.

A helyességet úgy ellenőriztem hogy az eredeti SDFből kapott képpel pixelenként összehasonlítottam a két megoldást. A 2.5. táblázatban látható a helyesen színezett pixelek aránya adott felbontáson. A használt tesztalakzatok a 2.16. ábrán láthatók.

A táblázat összehasonlításaiból kiderül, hogy a CSG szűrés pontosabb eredményt ad, és adott pontosságot alacsonyabb felbontással elér.

Lemértem a mező generálásának sebességét a CSG fa előzetes kiszámításával és anélkül. A 2.6 táblázatban az előző méréseknél használt alakzatok mezőjének generálási ideje

	R betű		Macska	
Res.	G1 CSG	MSDF	G1 CSG	MSDF
16 ²	0.13%	0.71%	0.9%	2.6%
32 ²	0.015%	0.11%	0.21%	0.98%
64 ²	0.0066%	0.022%	0.032%	0.13%
128 ²	0.0049%	0.0058%	0.018%	0.032%

2.5. táblázat. A CSG szűrt elsőrendű mező és a multichannel SDF (MSDF) potosságának összehasonlítása.

	A0	G1	G1 és CSG
Macska	0.05 ms	1.99 ms	3.5 ms
Antilop	0.04 ms	1.16 ms	2.02 ms
R betű	0.02 ms	0.42 ms	0.75 ms

2.6. táblázat. A mező generálásának sebessége 32² felbontású nulladrendű és elsőrendű mezőre. Elsőrendű mező esetén a CSG fa előre kiszámítható a mező generálása közben, ennek sebessége látható az utolsó oszlopban.

látható nulladrendű esetben, valamint elsőrendű esetben a CSG fa számításával és nélküle. A méréseket NVIDIA GTX 1650-es laptop GPU-n végeztem.

A táblázatból kiolvasható, hogy a CSG fa generálása nagyjából felére csökkenti a generálás sebességét. Ám mivel ezt csak a program futtatása kezdetén, egyszer kell elvégezni, így nem okoz problémát.

Végül a mező kiértékelésének sebességét hasonlítottam össze a különböző szűrési algoritmusokkal. Az eddigi mérésekben használt alakzatok kiértékelési idejét mértem különböző felbontásokon, és ezek átlaga lett az eredmény, amely a 2.7. táblázatban látható.

A mérésekből látható, hogy a CSG szűrés nem jelentősen lassabb a bilineáris szűrésnél, miközben nagyobb pontosságot biztosít, már alacsony felbontáson is.

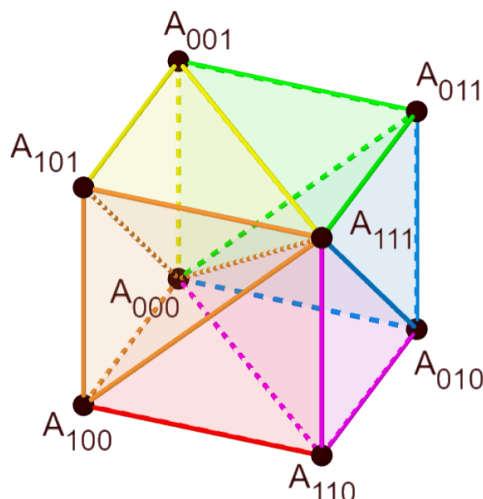
2.5. CSG szűrés 3D-ben

A 2D-s numerikus hibában és szubjektív kiértékelésben is jól teljesített, így feltehető hogy 3Dben is hatékony lenne egy hasonló módszer.

A következőkben bemutatom, hogyan általánosítottam a CSG szűrés koncepcióját a térbe.

	A0 (bilinear)	G1 (bilinear)	G1 (CSG)
Kiértékelés	0.16 ms	0.25 ms	0.38 ms

2.7. táblázat. A különböző mezők kiértékelésének ideje, a teljes képernyőn.



2.17. ábra. Adott kiértékelési ponthoz tartozó 8 legközelebbi texel elképzelhető egy kockaként, ahol a csúcsok a texeleket jelképezik. Ez felosztható 6 tetraéderre.

Három dimenzióban a legközelebbi 4 helyett a legközelebbi 8 cellában tárolt geometriát kéne kombinálni egy fába. Ezek a 2Ds esethez analóg módon félterek, amelyek talppontjukkal vannak definiálva.

Először ismét két geometriára kell meghatározni a szükséges műveletet.

Adott A és B félsík, $\mathbf{a}$ és $\mathbf{b}$ talppontokkal definiálva. A 2D-s esethez hasonlóan a talppontokról itt is feltehető hogy a felület határán helyezkednek el így ugyanazok az esetek lépnek fel, amelyek a 2.2 táblázatban voltak bemutatva.

2.5.1. A CSG fa meghatározása

A CSG fát itt 8 geometriából kellene felépíteni. Ha párosával meghatározzuk a műveleteket és ezekből felépítjük a műveleti gráfot, akkor egy 8 csúcsú és 28 élű gráfot kapunk.

A szimmetriákat és forgatásokat leszámítva ez több mint 40 esetet jelent, amelyek nagy részét nehéz vizualizálni, így bonyolult a CSG fa meghatározása. Emellett még ha sikerülne is beazonosítani a szükséges fákat, azok 8 geometriát és 7 műveletet tartalmaznának így a generálás és kiértékelés költséges lenne.

Ezért ahelyett hogy egyben kezelnénk a 8 geometriát felbontjuk őket több részre.

Egy adott kiértékelési ponthoz tartozó 8 legközelebbi texel elképzelhető egy kockaként, ahol a kocka minden csúcsa egy texelt reprezentál és tartozik hozzá egy-egy geometria. Ezt a kockát a [7]-ben bemutatott módon fel lehet osztani 6 tetraéderre. A felosztás látható a 2.17. ábrán.

Ha a kiértékelési pontról eldöntjük melyik tetraéderben van, a tetraéder csúcsai megadnak 4 geometriát. Ezekből az előző részben mutatottak alapján már fel tudunk írni egy 4 csúcsú CSG fát.

A tetraéder egyszerűen meghatározható, ha a kiértékelési pont koordinátáinak növekvő sorrendje alapján a 2.18 táblázat szerint.

Szín	Tetraéder
piros	$0 \leq z \leq y \leq x \leq 1$
narancs	$0 \leq y \leq z \leq x \leq 1$
sárga	$0 \leq y \leq x \leq z \leq 1$
zöld	$0 \leq x \leq y \leq z \leq 1$
kék	$0 \leq x \leq z \leq y \leq 1$
lila	$0 \leq z \leq x \leq y \leq 1$

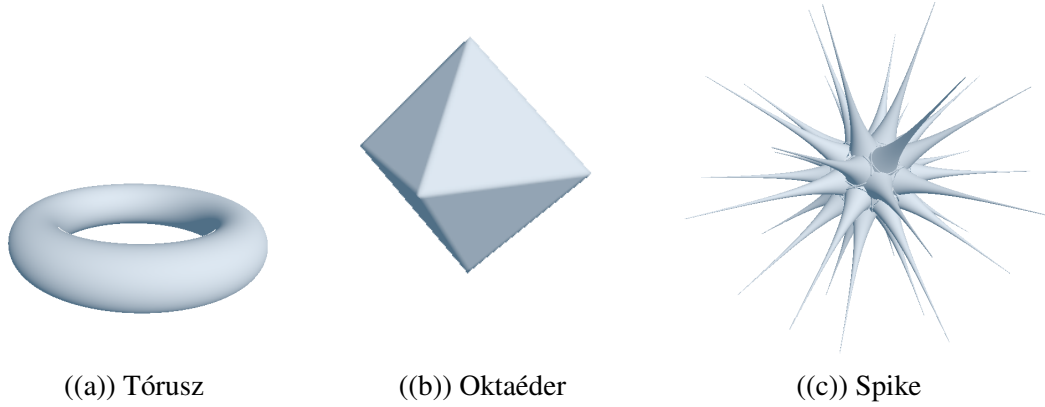
2.18. ábra. Ha a kockát tetraéderekre osztjuk a $\mathbf{p} = (x, y, z)$ pont koordinátájának sorrendje alapján eldönthető, hogy a pont melyik tetraéderbe esik. A színek a 2.17. ábrán látható színezés szerint vannak leírva.

Miután megvan a 4 csúcs, kiolvassuk a megfelelő 4 geometriát és innentől a feladat azonos a 2D-s esettel.

A fa generálása itt is megoldható előzetesen ám itt egy 3D rácsban tudnánk tárolni a fákat és minden cellában 6 fát kéne tárolni a 6 lehetséges tetraéderhez. Ez a megoldás hely és számításigényes így nem implementáltam, a megoldásomban a fát futásidőben számítom, de a jövőben szeretnék egy hatékony megoldást találni a fák tárolására.

2.5.2. Eredmények

A mérést ismét ugyanúgy végeztem mint az előzőekben egy 3D rács segítségével. A felbontás 217^3 volt, a méréshez használt alakzatok a 2.19 ábrán láthatók.



2.19. ábra. A méréshez használt SDF-ek. Az SDF-ek forrása: [8] és

Tórusz					
Res.	Norm	A0	nearest	G1 trilinear	CSG
16	1	0.0071	0.0019	0.0035	0.0014
	2	0.013	0.003	0.0044	0.0022
	∞	0.1	0.068	0.032	0.07
32	1	0.0018	0.00069	0.0011	0.00053
	2	0.0045	0.0010	0.0013	0.0007
	∞	0.046	0.037	0.017	0.033
64	1	0.00048	0.00037	0.00047	0.00027
	2	0.0016	0.00046	0.00053	0.00034
	∞	0.02	0.014	0.0082	0.019
128	1	0.00025	0.0003	0.00032	0.00018
	2	0.00053	0.00038	0.00037	0.00024
	∞	0.0061	0.0056	0.0035	0.0064

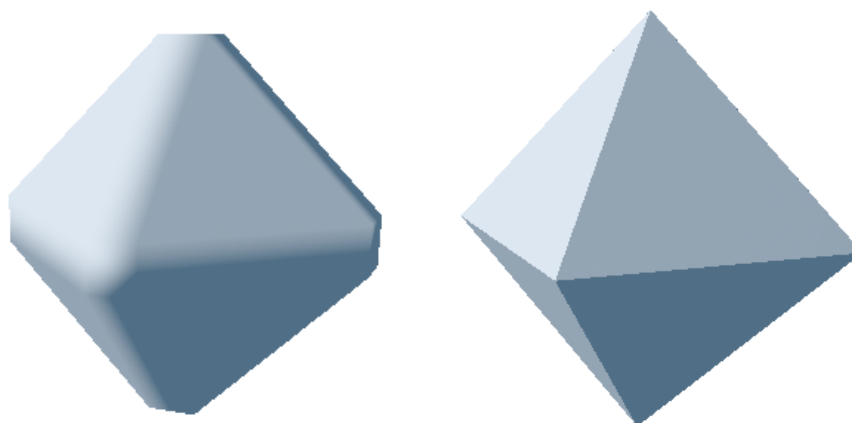
Oktaéder					
Res.	Norm	A0	nearest	G1 trilinear	CSG
16	1	0.0062	0.0017	0.0031	0.0015
	2	0.012	0.0024	0.0038	0.0025
	∞	0.1	0.03	0.027	0.078
32	1	0.0016	0.00078	0.0011	0.0006
	2	0.0042	0.00095	0.0013	0.00091
	∞	0.047	0.018	0.014	0.059
64	1	0.0005	0.00053	0.00062	0.00037
	2	0.0015	0.00065	0.00069	0.00049
	∞	0.02	0.0045	0.0057	0.027
128	1	0.00031	0.00047	0.00048	0.00027
	2	0.00055	0.0006	0.00057	0.00038
	∞	0.0059	0.0022	0.0031	0.0072

Spike					
Res.	Norm	A0	nearest	G1 trilinear	CSG
16	1	0.014	0.0083	0.1	0.0069
	2	0.019	0.02	0.016	0.015
	∞	0.22	0.28	0.26	0.33
32	1	0.0042	0.0022	0.0034	0.0016
	2	0.0068	0.0067	0.0055	0.0032
	∞	0.13	0.16	0.15	0.19
64	1	0.0011	0.00062	0.00099	0.00043
	2	0.0027	0.0029	0.0023	0.0020
	∞	0.3	0.29	0.28	0.29
128	1	0.00036	0.00022	0.00031	0.00018
	2	0.0015	0.0017	0.0014	0.0015
	∞	0.29	0.29	0.29	0.29

2.8. táblázat. A 3D-s elsőrendű mezők pontossága különböző szűrésekkel.

A 2.8. táblázatból látszik hogy a CSG szűrés itt is jelentősen javította a minőséget. A maximum hiba néhány esetben megnőhet, ennek egyik oka a talppontok pontatlansága, mivel a CSG szűrés a talppontok pozícióján alapul.

A 2D-hez hasonlóan itt is nagy előny, hogy az éles élek és sarkok megmaradnak, valamint már kis felbontással nagyon pontos közelítést érhetünk el.



((a)) trilineárisan szűrt A0, 32^3 felbontás (32768 skalár)

((b)) CSG szűrt G1, 2^3 felbontás (32 skalár)

2.20. ábra. Az oktaéder nulladrendű és elsőrendű mezővel közelítve. Látható hogy a CSG szűrt elsőrendű mező, már nagyon kis felbontáson pontos.

A 2.20 ábrán látható hogy a CSG szűrt elsőrendű mező már akár 2^3 -es felbontáson tökéletesen reprodukál egy oktaédert, míg nulladrendű mezővel még magas felbontás mel-

lett is lekerekített éleket kapunk.

A dolgozat következő részében az SDF távolságmezőjével elért eredményeket fogom kiterjeszteni egyéb bemenetekre és megmutatom hogy kis változtatással létrehozható egy általános felületreprezentáció, amely a CSG szűréssel kombinálva egyszerű és pontos közelítést adja különböző típusú bemeneti felületnek.

3. fejezet

Általános felületek geometriai távolságmezője

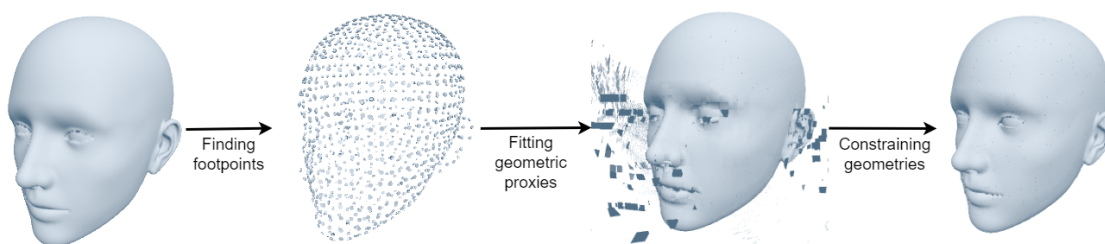
Láttuk, hogy SDF bemenethez a generált távolságmező megfelelő szűréssel hatékony approximációt biztosít.

A használt algoritmus 4 lépésből állt, ennek körvonala a 3.1 ábrán látható. Ezen lépések:

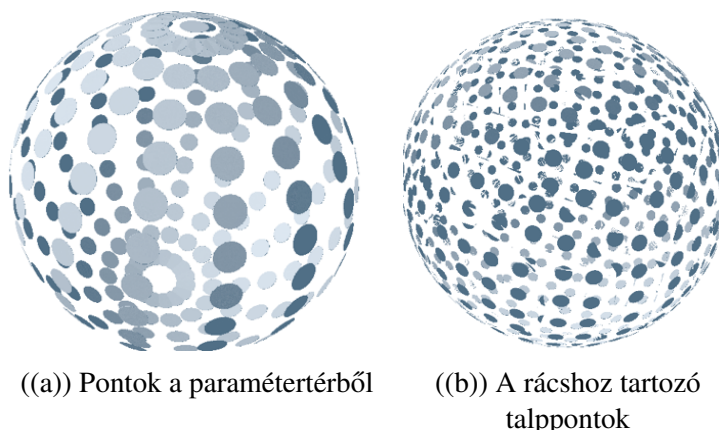
1. felületi mintapontok
2. geometriák illesztése
3. geometriák távolságfüggvényének felírása és korlátozása
4. mező szűrése

A lépések közül csak az első kettő függ a bemeneti felület megadásától, hiszen csak a generálásban használtuk ki az input SDF voltát, a megjelítésnél és szűrésnél csak a mezőből olvastunk.

Ez alapján a távolságmező általánosítható más bemeneti adatra is, csak a talppontkereső és geometriaillesztő algoritmust kell módosítani. Az illesztett geometria is bemenet-



3.1. ábra. A távolságmező generálásának és kiértékelésének összefoglalása



3.2. ábra. A felületi pontok választásának két lehetséges megoldása

független, hiszen azt matematikai elvárások alapján határoztuk meg és a lehető legegyszerűbb és általánosabb módon definiáltuk.

Így a távolságmező felfogható egy általános felületreprezentációként, amellyel tetszőleges felület felírható, majd egységes módszerrel kiértékelhető.

A következő részben bemutatom az 3D-s elsőrendű távolságmezők általánosítását. Először kitérek a felületi mintapontok választásának egyéb lehetőségeire az eddig használt rács és talppontkeresés mellett. Ezután bemutatom a távolságmező generálását parametrikus és implicit esetben, majd mérésekkel vizsgálom a módszer viselkedését.

3.1. Felületi pontok választása

A geometriai távolságmező lényege hogy adott felületi pontokban illesztett geometriákkal közelíti az eredeti alakzatot. Ezen felületi pontok pozíciója tetszőlegesen megválasztható, akár a felület lokális tulajdonságai alapján.

A dolgozat első felében a mezőt egy 3D ráccsal reprezentáltam, ahol minden cellához tartozott egy geometria melynek SDFje az adott cellában jó közelítése volt a valódi SDFnek. Ennek megfelelően a felületi mintapontok a cellák középpontjaihoz tartozó talppontok voltak.

Ezen kívül számos lehetőség van a mintapontok választására. Parametrikus felületek esetén például vehetünk egy adott felbontású rácsot a paraméterterben, majd ezekhez kiszámolhatjuk a megfelelő felületi pontokat. A 3.2. ábrán ezen megközelítéssel kapott pontok összehasonlítása látható a rácsos módszerrel.

A paraméterterben vett pontok előnye, hogy kevesebb duplikált pont lehet, míg a rácsos esetben több talppont egybeeshet. Akár olyan megoldást is lehetne készíteni ami

alkalmazkodik a felületi viszonyokhoz, és például a nagyobb görbületű részen sűrűbben vesz mintákat.

Ami viszont nagy hátránya ennek a módszernek, hogy a kapott adat strukturálatlan lesz, hiszen csak egy ponthalmazt kapunk, míg a rácsnál minden felületi pontról ismerjük melyik cellához tartozik.

Ez azért probléma, mert a geometriai folytonos csatlakozás definíciója alapján az illesztett geometriák csak a talppont egy kis környezetében adnak jó közelítést. A mező kiértékelésekor ez azt jelenti, hogy tetszőleges $\mathbf{x} \in \mathbb{R}^3$ mintapont körül annak a proxinak az SDFje közelít pontosan, amelynek mintapontja $f\mathbf{p}(\mathbf{x})$ -hez a legközelebb esik, mivel az $\mathbf{x}$ -ben vett SDF érték lényegében a talpponttól vett távolság.

Ez azt jelenti hogy ha az illesztett geometriáink strukturálatlanul helyezkednek el, muszáj közülük kiválasztani azt amelynek a mintapontja a legközelebb van $\mathbf{x}$ -hez. Ez számításigényes feladat lehet, hiszen az összes geometrián végig kell iterálni, ami a felbontástól függően nagyon sok is lehet. A módszer gyorsítható valamilyen térbeli adatstruktúra használatával, ám ennek létrehozása és bejárása is lassíthatja a kiértékelést.

Emiatt általánosságban, ha a lehető leghatékonyabb kiértékelést szeretnénk elérni érdemes a 3D rácson módszert használni, hiszen ilyenkor szélsőséges esetektől eltekintve feltehető, hogy egy cella belső pontjaihoz az adott cella középpontjához tartozó talppont van a legközelebb az összes talppont közül. Tehát tényleg elég csak a cellából olvasni kiértékeléskor, ami sokkal egyszerűbb, mint az összes geometriát vizsgálni.

Így általános esetben is az eddig használt megközelítésnél maradunk. Minden bemeneti felülettípusra definiáljuk a rácsot, majd megkeressük a középponthez tartozó talppontokat.

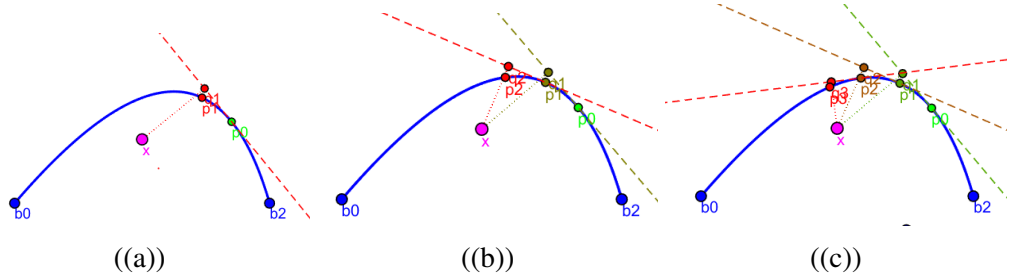
Az elsőrendű mező generálásának tehát két fő kérdése lesz általános esetben:

- hogyan találjuk meg adott ponthoz a talppontot?
- hogyan számítsuk ki az érintősíkot definiáló adatokat?

A következő részben ezeket a kérdéseket fogom körüljárni parametrikus és implicit felület esetében.

3.2. Parametrikus felület

Adott $\mathbf{s} : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ parametrikus felület. Az SDF bemenethez hasonlóan a 3D rács felírása után a cellákon iterálva, meghatározzuk a megfelelő proxit.



3.3. ábra. A geometriai Newton-Rhapon módszer működése 2D-ben. Adott görbén keressük az $\mathbf{x}$ -hez legközelebbi pontot. $\mathbf{p}_i$ pontban vesszük az érintőegyenest és ezen megkeressük $\mathbf{q}_i$ $\mathbf{x}$ -hez legközelebbi pontot, majd ezt visszakonvertáljuk a görbére $\mathbf{p}_{i+1}$ pontba. Ezt iterálva $\mathbf{p}_i$ a talpponthoz konvergál.

3.2.1. Talppontkeresés

Adott $\mathbf{x} \in \mathbb{R}^3$ térbeli ponthoz a $\mathbf{fp}$ legközelebbi pontot és a talppont $u, v \in \mathbb{R}$ paramétereit is meg kell határozni, mivel később ebből tudjuk az illesztett geometriát kiszámolni.

Ezek megkeresésére a geometriai Newton-Rhapon módszert használtam [9]. Ez egy iteratív módszer, mely a geometriai mezőkhöz hasonló ötleten alapul.

A 3.3 ábrán látható a módszer példája 2D-ben és az egyszerűség kedvéért először síkban írom le a módszert. Itt először választunk egy kiindulási $t_0 \in \mathbb{R}$ paramétert és hozzá tartozó $p_0 \in \mathbb{R}^2$ kiindulási pontot a görbén, majd ebben a pontban kiszámítjuk az érintőegyenest. Ez is lényegében egy proxiilesztés, amelyen megoldjuk az alapeladat egy egyszerűbb változatát, majd az eredményt visszavezetjük az eredeti feladatra.

Ez alapján az illesztett egyenesen megkeressük az adott $\mathbf{x}$ -hez legközelebbi q_1 pontot és a t'_1 paramétert, majd a deriválttal leosztva kiszámítjuk a t_1 paramétert az eredeti görbére. Ez egy jobb közelítés lesz mint az előző t_0 paraméter volt. Ezt iterálva a módszer a valódi talpponthez konvergál.

A 3D-s megoldás ehhez nagyon hasonló, csak egyenes helyett síkot illesztünk és két irányban változtatjuk a paramétereket. A konvergenciához érdemes a t_0 paramétert úgy választani, hogy minél közelebb legyen a talpponthez. Ezt úgy oldottam meg, hogy a paraméterterben vettem egy rácsot és a rácpontok közül azt az (u, v) paramétert választottam, amelyhez tartozó $\mathbf{s}(u, v)$ pont $\mathbf{x} \in \mathbb{R}^3$ ponthoz a legközelebb van.

A használt algoritmusunk leírása olvasható a 2. algoritmusban.

3.2.2. Proxiilestés

Miután megvan $\mathbf{fp} = \mathbf{s}(u, v)$ talppont és a pont $u, v \in \mathbb{R}$ paraméterei már meg tudjuk határozni az érintősíkot.

2. Algoritmus Geometriai Newton-Rhapson módszer a talppont meghatározásához

```

1: Input:  $f : [0, 1]^2 \rightarrow \mathbb{R}^3$  parametrikus felület;  $p \in \mathbb{R}^3$  pont;  $t_0 \in [0, 1]^2$  kiindulási pa-
   raméter
2: for  $k = 0 \dots n$  do ▷ rács a paraméterterben
3:   for  $j = 0 \dots n$  do
4:      $t = \begin{bmatrix} k/n \\ j/n \end{bmatrix}$ 
5:     if  $\|f(t) - p\| < \|f(t_0) - p\|$  then
6:        $t_0 = t$  ▷ a  $p$ -hez legközelebbi felületi pontot választjuk kiindulópontnak
7:     end if
8:   end for
9: end for
10:  $i = 0$ 
11: repeat
12:   while  $i < 2 \vee \|p_i - p_{i-1}\| > \varepsilon$  do
13:      $p_i = f(t_i)$ 
14:      $\partial_u = \partial_u f(t_i)$ 
15:      $\partial_v = \partial_v f(t_i)$ 
16:      $b = \begin{bmatrix} (p - p_i) \cdot \partial_u \\ (p - p_i) \cdot \partial_v \end{bmatrix}; \quad A = \begin{bmatrix} \partial_u \cdot \partial_u & \partial_u \cdot \partial_v \\ \partial_v \cdot \partial_u & \partial_v \cdot \partial_v \end{bmatrix}$ 
17:      $c = A^{-1} \cdot b$  ▷ lépés az érintősíkon
18:      $q_i = p_i + c_u \cdot \partial_u + c_v \cdot \partial_v$  ▷ legközelebbi pont az érintősíkon
19:      $p_{i+1} = f(t_i + c)$  ▷ legközelebbi pont becslése
20:      $f_1 = q_i - p_i; \quad f_2 = p_{i+1} - q_i$ 
21:      $a_0 = (p - p_i) \cdot f_1; \quad a_1 = (p - p_i) \cdot 2f_2 - f_1 \cdot f_1; \quad a_2 = -3f_1 \cdot f_2; \quad a_3 =$ 
        $-2f_2 \cdot f_2$ 
22:      $\Delta t = \left(1 - \frac{a_0 + a_1 + a_2 + a_3}{a_1 + 2 \cdot a_2 + 3 \cdot a_3}\right) c$  ▷ lépés az eredeti paraméterrel
23:      $t_{i+1} = t_i + \Delta t$ 
24:      $i = i + 1$ 
25:   end while
26: until  $\|p_i - p_{i-1}\| < \varepsilon$ 
27: Output:  $t_i$  ▷ a legközelebbi felületi pont paramétere

```

Ehhez a talppontban vett normálvektorra van szükség. Ez a vektor a parametrikus felület ∂_u és ∂_v szerinti iránymenti deriváltjaira merőleges, így $\mathbf{n} = \partial_u \mathbf{s}(u, v) \times \partial_v \mathbf{s}(u, v)$.

A deriváltakat meghatározhatjuk pontosan például Bézier felületek esetében, de ha erre nincs lehetőség közelíthetjük centrális differenciával:

$$\partial_u \mathbf{s}(u, v) = \frac{\mathbf{s}(u + \varepsilon, v) - \mathbf{s}(u - \varepsilon, v)}{2\varepsilon}$$

ahol $\varepsilon \in \mathbb{R}$ egy kellően kicsi szám.

A sík tárolását az SDF-nél bemutatott módon végezzük, az egységes kiértékelés érdekében. Tároljuk a $\mathbf{fp}$ talppontot viszont $\mathbf{n}$ -et nem hiszen ezt a talppontból és a $\mathbf{grid}(i)$ cellaközéppontból ki tudjuk számolni a kiértékeléskor.

Viszont ahogy az előző fejezetben is leírtam a kiszámolt vektor lehet a normálissal ellentétes irányú, ezért szükség van egy s irányskalárára, amely megmondja melyik irányba kell néznie a vektornak. Ez azért szükséges mert a szűrések működéséhez a felületnek egységes normálsöprése kell legyen. Ahogy korábban láttuk például a CSG szűrés is nagyban támaszkodik a proxim kívül-belül felosztására.

Az SDF esetén s -et az eredeti alakzat kívül-belül felosztása alapján határoztuk meg, ám ilyen parametrikus felületeknél nem mindig létezik, például egy nem zárt Bézier felület esetében.

Viszont azt feltehetjük, hogy jól megadott felület esetén a keresztszorozattal számolt normálvektorok söprése egységes lesz. Így s értékét megállapíthatjuk az alapján, hogy a keresztszorozattal kapott normálvektor iránya azonos-e a talppontból a cellaközéppontba mutató vektor irányával. Ez a proximnak is definiál egy kívül-belül felosztást és a mező így előjeles távolságmező lesz.

Tehát $s = \begin{cases} 1 & \text{ha } \langle \mathbf{n}, \mathbf{grid}(i) - \mathbf{fp} \rangle > 0 \\ 0 & \text{ha } \langle \mathbf{n}, \mathbf{grid}(i) - \mathbf{fp} \rangle < 0 \end{cases}$, ahol $\mathbf{n} = \partial_u \times \partial_v$ és $\mathbf{grid}(i)$ az $i \in \mathbb{N}^3$ indexű cella középpontja.

3.3. Implicit felület

Adott $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ implicit felület. Az implicit felület lényegében az SDF általánosítása, így a mező generálása még hasonlóbb lesz az SDF esetéhez mint a parametrikus felületnél.

3.3.1. Talppontkeresés

Adott $\mathbf{x} \in \mathbb{R}^3$ felületi ponthoz tartozó talppont keresésére, megint iteratív módszert használtam. A talppontot egyszerű Newton módszerrel kerestem meg. [10].

Ez a módszer akkor konvergál ha a keresést kellően közel indítjuk a vélt talpponthoz ezért az induló x_0 pontot, úgy választottam, hogy $\mathbf{x}$ -ből a gradienssel ellentétes irányba elléptem és a kapott pont körül egy rácsban több pontból is elindítottam a keresést.

3.3.2. Proxiillesztés

Miután megvan az $\mathbf{fp}$ talppont a sík adatainak meghatározása ugyanúgy történik mint az eddigi részben.

Az $\mathbf{n}$ normálvektor az $\mathbf{fp}$ beli gradienssel párhuzamos, és a normálvektorok söprése egységes. Az s irányskalár számítása a parametrikus felület esetéhez azonos módon történik.

3.4. A mező kiértékelése

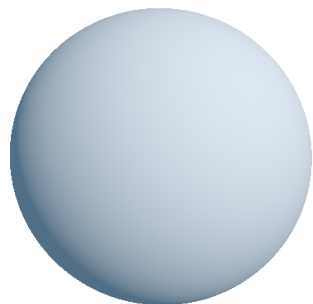
Miután legeneráltuk a mezőt egy inputfüggetlen 3D rácsot kapunk, melynek minden cellája tárolja az érintősíkot megadó talppontot és irányskalárt.

A mező kiértékelése a 2. fejezetben leírttal azonos módon történik. A kiértékelési pozíció alapján kiolvassuk a mezőből az adatokat, majd felírjuk a normálvektort a (2.3) képlettel. A távolságfüggvény a (2.4) képlettel számolható. Végül a geometriákat egy gömbbel korlátozzuk, hogy csak a talppont körül érvényesüljenek, ahogy a (2.5) képletben látható.

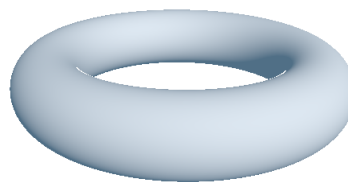
A mező szűrésére használható trilineáris módszer a legközelebbi 8 texelből olvasott geometriától vett távolságokon, vagy az előző fejezetben bemutatott 3D-s CSG szűrés is használható.

3.5. Eredmények

Az előzőekben bemutattam egy köztes reprezentációt általános felületek leírására. Az eltérő generálás miatt viszont a pontosság nagyban függhet a bemenettől, ezért először a különböző bemenetekből generált mezők pontosságát hasonlítottam össze.



((a)) Gömb



((b)) Tórusz

3.4. ábra. A három bementből generált mező összehasonlításához használt alakzatok

A méréshez az SDF inputnál már bemutatott mérési módszert használtam. Vettem egy 217^3 -ös felbontású finom rácsot, amelynek minden pontjában kiértékeltem a pontos távolságot valamint a mezőkből kapott értéket, maj a különbségekből alkotott vektorok különböző normáit vettem.

A három bemenetre generált mező összehasonlítására a 3.4. ábrán látható egyszerű alakzatokat használtam, mivel ezekre mind a három típusú bemeneti függvényt meg tudtam adni. Az egyes mezőkkel kapott eredmények a 3.1. táblázatban láthatók.

Az eredményekből látható, hogy míg a gömbön az összes mező hasonlóan jól teljesít, a bonyolultabb alakzatokon már nagyobb a hiba, főleg az implicit felület esetén. Ennek egyik oka az lehet, hogy a Newton módszerrel talált talppontok sokszor pontatlanabbak.

Ezek a módszerek akkor konvergálnak, ha a valódi talpponthoz viszonylag közlelről indítjuk, ám ilyen közeli pont keresése és a módszer stabilitásának biztosítása bonyolult feladat, ami túlmutat a dolgozat hatáskörén. A gömbön kapott jó eredmények alapján feltehető, hogy kellően robusztus talppontkereső algoritmus mellett az implicit felületekre generált távolságmező is a többihez hasonlóan hatékony lesz, de ilyen algoritmusok keresése a jövőbeli kutatás része.

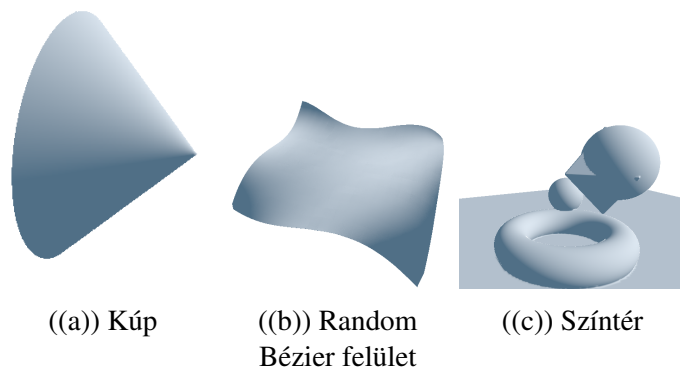
A parametrikus és SDF inputból generált mezőkre külön méréseket futtattam, hogy teszteljem az egyéni viselkedésüket.

A parametrikus felületeket a 3.5. (a) és (b) ábrán látható kúpra és véletlen generált Bézier felületre teszteltük az eredményeket a 3.2. táblázat tartalmazza. Mivel ezekre a felületekre nem ismertük a pontos SDF-et, a pontos távolságot Newton-Rhapon iterációval határoztuk meg.

SDF inputra egy komplex szintérre mértünk, az eredmények a 3.3 táblázatban láthatók.

Parametrikus							
Res.	Norm	nearest	Gömb trilin.	CSG	nearest	Tórusz trilin.	CSG
16^3	1	0.0019	0.003	0.0032	0.12	0.12	0.093
	2	0.0000093	0.000015	0.000035	0.019	0.018	0.013
	∞	0.069	0.044	0.05	0.21	0.21	0.21
32^3	1	0.0005	0.0008	0.00077	0.066	0.06	0.051
	2	0.0000022	0.000013	0.0000046	0.0061	0.0049	0.0061
	∞	0.033	0.29	0.31	0.35	0.34	0.36
SDF							
Res.	Norm	nearest	Gömb trilin.	CSG	nearest	Tórusz trilin.	CSG
16^3	1	0.0017	0.0032	0.0013	.002	0.0035	0.0014
	2	0.000001	0.000001	0	0.000001	0.000001	0.000001
	∞	0.05	0.032	0.025	0.068	0.032	0.07
32^3	1	0.0008	0.0013	0.00041	0.00069	0.0011	0.00054
	2	0	0	0	0	0	0
	∞	0.029	0.02	0.011	0.037	0.017	0.032
Implicit							
Res.	Norm	nearest	Gömb trilin.	CSG	nearest	Tórusz trilin.	CSG
16^3	1	0.0017	0.0027	0.0011	0.31	0.31	0.28
	2	0.0000062	0.000012	0.0000027	0.13	0.13	0.1
	∞	0.062	0.043	0.03	0.73	0.72	0.7
32^3	1	0.00033	0.00053	0.00022	0.24	0.28	0.28
	2	0.00000026	0.00000052	0.00000012	0.075	0.099	0.1
	∞	0.029	0.021	0.013	0.61	0.67	0.68

3.1. táblázat. A háromféle bemenetből generált mezők pontossága



3.5. ábra. A parametrikus felület és az SDF méréséhez használt alakzatok

Kúp				
Res.	Norm	nearest	G1 trilin.	CSG
16^3	1	0.00089	0.0013	0.00054
	2	0.0000039	0.0000055	0.0000014
	∞	0.085	0.087	0.082
32^3	1	0.00043	0.0013	0.00052
	2	0.00000079	0.00016	0.00000077
	∞	0.048	0.23	0.047
64^3	1	0.00047	0.0012	0.00062
	2	0.00000068	0.000027	0.00000088
	∞	0.037	0.038	0.047

Véletlen Bézier felület				
Res.	Norm	nearest	G1 trilin.	CSG
16^3	1	0.031	0.065	0.023
	2	0.0028	0.014	0.0037
	∞	0.18	0.54	0.48
32^3	1	0.02	0.063	0.016
	2	0.0011	0.015	0.000053
	∞	0.14	0.51	0.13
64^3	1	0.012	0.064	0.016
	2	0.00037	0.015	0.000055
	∞	0.11	0.52	0.14

3.2. táblázat. Parametrikus felületek pontossága

Scene				
Res.	Norm	nearest	G1 trilin.	CSG
16^3	1	0.003	0.0042	0.0021
	2	0.000003	0.000002	0.000002
	∞	0.27	0.25	0.59
32^3	1	0.001	0.0015	0.00071
	2	0.000001	0.000001	0.000001
	∞	0.27	0.26	0.59
64^3	1	0.00056	0.00067	0.00044
	2	0.000001	0.000001	0.000001
	∞	0.25	0.25	0.27

3.3. táblázat. Az SDF bementből generált mező pontossága

Res.	Parametric	SDF	Implicit
16	0.06 ms	0.03 ms	0.01 ms
32	0.20 ms	0.07 ms	0.05 ms
64	1.33 ms	0.39 ms	0.32 ms

3.4. táblázat. A mező generálásának ideje különböző inputra

Ezekből a mérésekből látható, hogy a mezők nagy pontosságot biztosítanak, már kis felbontás mellett is. Az is látható hogy a hiba főleg a maximum hiba nem csökken a felbontás növelésével, és még akkor is nagy értéket vesz fel, amikor az átlaghiba és a 2-es norma kicsi. Ennek oka a talppontok számolásakor fellépő apró hiba lehet, mivel a módszerek erősen támaszkodnak a talppont pontosságára.

Az elmélet validálására elvégeztem egy tesztet. Vettem egy gömböt amelyre nagyon pontosan meg tudjuk határozni a talppontokat és az egyik talppontot kézzel elrontottam, majd a szokásos módon kiszámoltam a proxikat. Ez a várt eredményre vezetett: az átlaghiba kicsi maradt, de a maximum hiba jelentősen megnőtt. Amikor a talppontot egy 0.01 hosszú vektorral toltam el a maximum hiba négyszeresére nőtt, ebből is látható hogy a talppontok nagy szerepet játszanak ebben a jelenségben.

Az is megfigyelhető, hogy a legtöbb alakzaton a CSG szűrés teljesít a legjobban az inputtól függetlenül. Ez igazolja hogy a távolságmezők hatékony használatához szükség van a megfelelő szűrési eljárásra. Néhány esetben, főleg magasabb felbontáson a CSG szűrés hasonló vagy kicsit pontatlanabb eredményt ad mint a szűretlen mező, ennek oka az lehet, hogy ilyenkor kellően nagy a felbontás, ahhoz hogy a mező önmagában pontos legyen, így a CSG szűrés csak numerikus hibát ad az eredményhez, a talppontok pontatlanságai miatt.

Lemértem a mezők generálásának idejét is a különböző bemenetekre. A méréshez a gömböt és a tóruszt használtam, eredménynek az egyes alakzatokon mért idők átlagát vettem. A mérést NVIDIA GTX 1650-es laptop GPU-n végeztem. Az eredmények mezőnként a 3.4. táblázatban láthatók.

Az eredmények azt mutatják hogy a parametrikus felület mezőjének generálása kicsit lassabb a többinél, ennek oka az lehet hogy ki kell számolni az iteráció kiindulópontját.

4. fejezet

Összefoglalás

A korábbi eredményeimre támaszkodva bemutattam az elsőrendű geometriai távolságmezőket és egyszerű szűrési algoritmusokat. Az eredményekkel megindokoltam egy a geometriai adathoz jobban illeszkedő szűrés szükségességét.

Ezután bemutattam az új eredményeimet, kezdve a CSG szűrés módszerével amely 2D-s elsőrendű mezők szűrésére alkalmas és megőrzi az adatok geometriai jelentését. Ez a megoldás képes éles sarkokat és pontos cut locus-t tárolni.

A CSG szűrés hatékonyan számítható és tárolható, valamint a mérések megmutatták hogy már alacsony felbontáson nagy pontosságot biztosít és más hasonló módszereknél lényegesen jobban teljesít.

A CSG szűrést általánosítottam 3D-ben, a koncepció egyszerűségének megtartásával és itt is igazoltam, hogy jelentős javulást lehet elérni a használatával. A CSG fák tárolása itt még nem optimális ezért a jövőben szeretnék kidolgozni egy hatékonyabb tárolási módszert. Emellett szeretném bevezetni a CSG szűrést magasabb rendű mezőkre is.

A harmadik fejezetben bemutattam, hogy a távolságmező koncepciója általánosítható általános felületre és létrehozható egy általános felületreprezentáció geometriai proxim használatával.

Megindokoltam, hogy miért a rácson vett mintapontok használata a leghatékonyabb, majd bemutattam a talppontok számolását és a proxim generálását a különböző felületekre.

Megmutattam, hogy a mező az inputtól függetlenül egységesen kiértékelhető és a CSG szűréssel kombinálva hatékony eredményt kapunk. Az eredményekből látszott, hogy a távolságmezők már alacsony felbontáson pontosak viszont nagyon érzékenyek a talppontok pontatlanságaira. Ezért a továbbiakban szeretném megvizsgálni a talppontkeresés egyéb lehetőségeit és kidolgozni egy stabilabb és hatékonyabb algoritmust.

Köszönetnyilvánítás

A KULTURÁLIS ÉS INNOVÁCIÓS MINISZTERIUM ÚNKP-23-2 KÓDSZÁMÚ ÚJ NEMZETI KIVÁLÓSÁG PROGRAMJÁNAK A NEMZETI KUTATÁSI, FEJLESZTÉSI ÉS INNOVÁCIÓS ALAPBÓL FINANSZÍROZOTT SZAKMAI TÁMOGATÁSÁVAL KÉSZÜLT.

Irodalomjegyzék

- [1] Róbert Bán és Gábor Valasek. „Geometric Distance Fields of Plane Curves”. *Acta Cybernetica* 25.2 (2021. aug.), 187–203. old. DOI: 10.14232/actacyb.289248.
- [2] Charles Loop és Jim Blinn. „Rendering Vector Art on the GPU”. *GPU Gems 3*. Szerk. Hubert Nguyen. Addison-Wesley Professional, 2007. 25. fej. URL: <https://developer.nvidia.com/gpugems/gpugems3/contributors>.
- [3] David Turner, Robert Wilhelm és Werner Lemberg. *The FreeType Project*. URL: <https://www.freetype.org/index.html>.
- [4] Towaki Takikawa, Andrew Glassner és Morgan McGuire. „A Dataset and Explorer for 3D Signed Distance Functions”. *Journal of Computer Graphics Techniques (JCGT)* 11.2 (2022. ápr.), 1–29. old. ISSN: 2331-7418. URL: <http://jcgt.org/published/0011/02/01/>.
- [5] Inigo Quilez. *2D SDF Primitives*. <https://www.shadertoy.com/playlist/MXdSRf>. Accessed: (2023-11-29). 2023.
- [6] Viktor Chlumsky. *Shape Decomposition for Multi-channel Distance Fields*. Master’s thesis. Available at <https://github.com/Chlumsky/msdfgen/files/3050967/thesis.pdf>. Example City, CA, 2015. máj.
- [7] Róbert Bán és Gábor Valasek. „Tetrahedral Interpolation on Regular Grids”. *Eurographics 2021 - Posters*. Szerk. Jirí Bittner és Manuela Waldner. The Eurographics Association, 2021. ISBN: 978-3-03868-134-2. DOI: 10.2312/egp.20211031.
- [8] Inigo Quilez. *3D Signed Distance Functions*. <https://iquilezles.org/articles/distfunctions/> [Accessed: 30 November, 2023]. 2022.

- [9] Michael Kallay. „A geometric Newton–Raphson strategy”. *Computer Aided Geometric Design* 18.8 (2001), 797–803. old. ISSN: 0167-8396. DOI: [https://doi.org/10.1016/S0167-8396\(01\)00070-X](https://doi.org/10.1016/S0167-8396(01)00070-X). URL: <https://www.sciencedirect.com/science/article/pii/S016783960100070X>.
- [10] Jeffrey D. Blanchard és Marc Chamberland. „Newton’s Method Without Division”. *The American Mathematical Monthly* 130.7 (2023), 606–617. old. DOI: 10.1080/00029890.2022.2093573. eprint: <https://doi.org/10.1080/00029890.2022.2093573>. URL: <https://doi.org/10.1080/00029890.2022.2093573>.