Tartalomjegyzék

Dinamikusan generált textúra alapú vonalkázás

Szécsi László, Szirányi Marcell **

Budapesti Műszaki és Gazdaságtudományi Egyetem
{szecsi,sziranyi}@iit.bme.hu

Absztrakt. Cikkünk egy paraméterezett felületek vonalkázott megjelenítésére alkalmas valósidejű procedurális textúrázó algoritmust mutat be. Úgy terjesztjük ki a *Tonal Art Maps* technika koncepcióját, hogy a mintázat önhasonló jellegűvé váljon, így végtelen sok részletességi szinten alkalmazható legyen. Így korlátlanul ráközelíthetünk a felületekre anélkül, hogy a művészi stílus csorbulna vagy a minta a felületen elcsúszna. Feltárjuk az ilyen jellegű vonásminták matematikai tulajdonságait, és ismertetjük a procedurális modellek kialakítására, illetve a valósidejű textúrázásra javasolt algoritmusainkat.

1. Bevezetés

A valamely művészi kifejezésmódot, illusztrációs technikát utánozó eljárásokat [6, 13, 15], összefoglalóan *nem-fotorealisztikus képalkotásnak (NPR)* nevezik. A legalapvetőbb technikák egyike a vonalkázás, mely a kézi rajzhoz hasonlatos, azaz a képtérben közel azonos méretű, illetve animáció során az objektum felületéhez tapadó vonalkák segítségével érzékelteti a felületek alakját és mozgását, mindezt anélkül, hogy időbeli hibajelenségek lépnének fel. Különösen arra kell ügyelni, hogy amikor a kamera látószöge, vagy objektumtól mért távolsága változik, a vonások objektumtérbeli sűrűsége a vonások elmozdulása vagy villódzása nélkül alkalmazkodjon, miközben a képtérben a kézi rajzra jellemző egyenletes, de véletlenszerű eloszlás megmarad [9, 1].

Ebben cikkben a *Recursive Proedural Tonal Art Maps (RPTAM)* nevű, vonalkázásra kifejlesztett új módszert mutatjuk be, mely teljesíti a fenti kritériumokat, de kevesebb megkötéssel jár, mint a korábbi megoldások. Nevezetesen, a képek alkotóelemei vonások (és nem minta-textúrák), korlátlan nagyítás lehetséges (nem csak véges számú részletességi szint), és a lokális árnyalás elegendő (nem szükséges a geometria feldolgozása a takart vonások törléséhez). Az alapötlet, hogy a vonásokat textúratérben, önhasonló magpontkészlet alapján helyezzük el úgy, hogy a különböző részletességi szintek között a folytonos átmenet lehetséges legyen.

A cikk a következőképpen épül fel. A 2. szakaszban összegezzük az NPR-rel és az önhasonlósággal kapcsolatos korábbi eredményeket. A 3. szakaszban bevezetjük a dinamikus textúra alapú vonalkázás (Recursive Procedural Tonal Art Maps – RPTAM) alapötletét. A 4. szakaszban bemutatjuk a vonásmintázatok önhasonlóságát lehetővé tevő matematikai konstrukciót. A 4.1. szakaszban

** korábban megjelent, mint: Recursive Procedural Tonal Art Maps, WSCG 2014

az vizsgáljuk, hogyan állíthatók elő jó minőségű vonáskészletek. A megjelenítési fázist az 5. szakaszban tárgyaljuk; itt írjuk le a részletességi szintek és a vonásméretek megválasztásának sémáját is. Az eredmények és a továbbfejlesztési lehetőségek összefoglalása zárja a cikket.

2. Korábbi munkák

2.1. Sűrűségi és iránymezők

A ceruzarajzokon a művészek az egyes tárgyak formáját és megvilágítását vékony vonások sűrűségével, irányával, hosszúságával, szélességével és átlátszóságával érzékeltetik. Ahhoz, hogy ezt utánozzunk, meg kell találnunk a művészi kifejezésmódot legjobban kifejező sűrűség- és iránymezőket a képsíkon. A sűrűség, hosszúság, szélesség és átlátszóság a megvilágítástól függ, míg az irányt a geometriai forma határozza meg. A művészek kereszt-vonalkázást is használhatnak, azaz több, az iránymezőhöz képest eltérő szögben álló vonalkázás-réteget.

Jelen cikkben nem tárgyaljuk az iránymezők előállításának kérdését, hanem feltételezzük, hogy a felületekhez az UV paraméterezés már adott, és a vonalkázás erre illeszkedik.

2.2. Magpont alapú vonalkázás

Számos munkában [11, 16] javasolták azt, hogy részecskéket vagy *magpontokat* rendeljünk felületekhez, és a képtérben az iránymezőt követve ezekből húzzunk vonásokat [19, 12]. Ebben a megközelítésben az alapfeladat az, hogy a magpontokat úgy helyezzük el az objektumok koordináta-rendszereiben, hogy azok a kívánt képtérbeli vonássűrűséget eredményezzék. Ennek megoldásához vagy a magpontok *tizedelésére* és *szaporítására* [18, 5], vagy *visszautasításos mintavételezésre* [16] van szükség. Ezek a technikák többnyire valós időben működnek, de több rajzolási menetet és jelentős erőforrásokat igényelnek. A vonások háromdimenziós geometriaként történő kirajzolása nem problémamentes: a kihúzott vonás-szalagok mélységének összevetése a felületek mélységével csak nagy ráhagyással és szűréssel ad villódzásmentes eredményt.

Jelen cikkben mi is átvesszük a magpontok fogalmát, hogy a vonások pozícióit illetve elosztásukat jellemezhessük. Azonban a magpontokat nem az objektumtérben, hanem a textúratérben vesszük fel, ugyanis nem háromszögszalaggá húzzuk ki őket, hanem procedurális textúrázáshoz használjuk fel a pozícióikat.

2.3. Önhasonlóság a stilizált képszintézisben

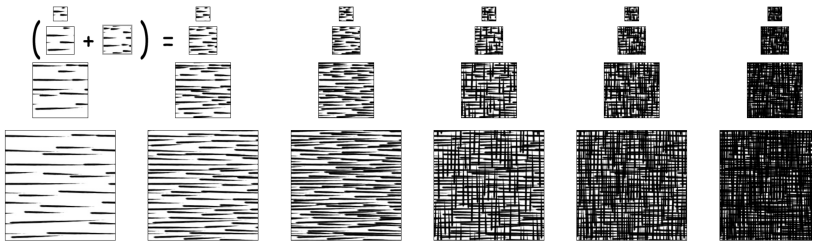
A *dynamic solid textures* technika [4] már bevezette a végtelen közelítés lehetőségét a stilizált képalkotásban, ahol önhasonló oktávok összegére bontott textúrák körkörös jellegű méretezése és súlyozása lehetővé teszi a korlátlan skálázást.

Az általunk tárgyalt megoldás motivációja ugyanez, és az önhasonlóságot is kihasználjuk, de kép alkotóelemei a vonások maradnak, ezzel kifejezetten vonalkázást, és nem általános textúrázást valósítunk meg, így elkerüljük a kontrasztvesztést és a nem kívánt frekvenciák megjelenését.

A *Halton-sorozat* is felhasználható magpontok generálására [16]. Az ilyen sorozat csonkolásával statisztikailag hasonló, de ritkább vonalkázást kaphatunk. Így tetszőleges számú magpontot lehet generálni, ami alkalmassá teszi a megoldást végtelen részletesség megvalósítására. Mi hasonló hatás elérésére törekszünk, de textúra-térben dolgozunk, hogy elkerüljük a vonásokkal kapcsolatos geometriai számításokat, és a takart vonások törlésével járó nehézségeket. További különbség, hogy mi véges, de önhasonló magpontkészlettel dolgozunk, így nem szükséges a magpontkészletet nézetfüggő módon mindig újra előállítani.

2.4. Textúra alapú vonalkázás

Annak biztosítására, hogy a vonások az objektum felületén rögzítve jelenjenek meg, a vonalkázást előzetesen textúrákba rajzolhatjuk, amit később a felületekre képezhetünk [10]. Ehhez paraméterezett felületek szükségesek. Az ilyen textúra alapú megoldások legjelentősebb hátránya, hogy csak korlátozott számú részletességi szintet tudnak megjeleníteni. Továbbá a vonások mérete a textúratérben, így – az UV koordináták hozzárendelése miatt – az objektumtérben is rögzített.



1. ábra: A TAM textúrák egymásba ágyazottsága: ha egy vonás szerepel egy képen, akkor a tőle jobbra és lentebb található képeken is szerepel. Forrás: Praun et. al [13].

Ezért az egyszerű textúrázás nem alkalmas a képtérben egységes vonalkázásra. Ennek kiküszöbölésére jelent meg a *Tonal Art Maps (TAM)* technika, amely hatékonyan növeli a textúra alapú vonalkázás használhatóságát [13]. Ezzel a módszerrel számos textúraképet generálunk előre, melyek különböző árnyalatok különböző felbontású vonalkázás-mintáit tartalmazzák. Az 1. ábra, melyet a hivatkozott cikkből vettünk át, egy ilyen textúragyűjteményt mutat be. Felületek megjelenítésekor minden képpontban a megfelelő textúrát kell kiválasztani a kívánt árnyalat, illetve a képtérbeli vonásméret alapján. A különböző részletességi szintek közötti éles határok elkerülése végett a textúrák közötti váltás folytonos áttűnéssel történik.

Az animáció során a felületen elvárt vonássűrűség folyamatosan változik, a vonások mégsem mozdulhatnak el a felülethez rögzített pozíciójukból. Megengedett, hogy a sűrűség növekedésével új vonások jelenjenek meg, illetve, hogy a sűrűség csökkenésével korábban látható vonások tűnjenek el. Azonban egymáshoz közel nem lehetnek egyszerre megjelenő és eltűnő vonások. Ezért a sűrűn vonalkázott textúráknak mindig tartalmazniuk kell a ritkább textúrák vonásait. Ezt a tulajdonságot *beágyazottságnak* nevezzük, mely a 1. ábrán figyelhető meg.

A TAM azonban csak a legjobb és legrosszabb felbontású textúra-szintek között működik. Ezért egy felületre közelítve nem kapunk részletesebb vonalkázást annál, mint amit a legnagyobb felbontású textúránk nyújtani tud, ami a kritikus szint elérése után hatalmas vonásokat és az elvártnál egyre ritkásabb vonalkázást eredményez. Ezen felül szoros összefüggés van a részletességi szintek száma és a szintek közti átmenetek között. Ha túl kevés textúrát használunk, túl sok vonás van megjelenőben egyszerre, ezért a képen a vonások erőssége nem lesz egységes.

3. Rekurzív TAM

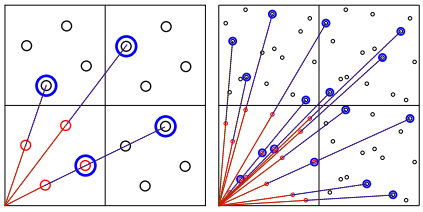
Az a célunk, hogy egyesítsük egyfelől a TAM egyszerűségét és robusztusságát, másfelől a determinisztikus magpontgenerálás és az elutasításos mintavételezés korlátlan részletességét. Ehhez a TAM technikát végtelen mélységig ismételhetővé terjesztjük ki azzal, hogy a beágyazó tulajdonságot rekurzívvá tesszük.

Ha a vonalkázás mintáját textúrának fogjuk fel, azt mondhatjuk, hogy ebben a textúrában a vonásokat *magpontokba* helyezzük. Ezt a textúrát 2×2-es rácsot alkotó négy csempére bonthatjuk. Előírjuk, hogy bármely negyedet tekintve, az ott elhelyezkedő magpontok halmaza a teljes, felére kicsinyített és a negyedre illesztett magponthalmaz részhalmaza legyen. Így a magpontok be vannak ágyazva abba a mintába, amit az eredeti minta mindkét tengely mentén való kétszeri ismétlésével kapunk (2. ábra). Ennélfogva, ha ráközelítünk valamelyik negyedre, az ott látható mintához úgy adhatunk hozzá új vonásokat, hogy azok az eredeti mintát alkossák újra. Ezt nevezzük a magpontkészlet rekurzív beágyazottságának, amit részletesebben a 4. szakaszban tárgyalunk majd.

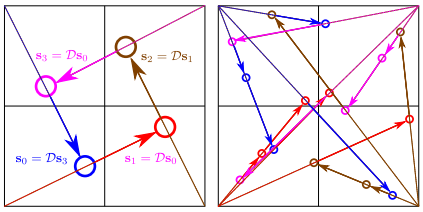
A teljes megoldás munkamenete a következő:

- Előzetesen (offline) néhány bájt hosszúságú bitmintákat generálunk. Ezek kódolják a rekurzívan beágyazott magpontkészleteket.
- Egy magpont-azonosítókat tartalmazó, ún. fedési textúrát hozunk létre. Ez jelzi, hogy egy felületi pontra mely vonások kerülnek. Ezt a textúrát csak akkor kell újrarenderelni, ha a vonalkázás stílusa (pl. vonások szélessége) megváltozik.
- A felületi pontok árnyalásakor kiszámítjuk a szükséges részletességi szintet. Ez alapján skálázzuk a fedési textúrát, és az adott pontban releváns magpont-azonosítókat kiolvassuk belőle. A magpontok pozícióit a bitmintákból illetve az azonosítókból rekonstruáljuk. A vonásokat — előre megfestett textúrával, a részletességi szintnek megfelelő méretben és átlátszósággal —

ezekre a pozíciókra helyezzük, és összegezzük hozzájárulásaikat az árnyalt felületi pont színéhez.



2. ábra: 4 és 16 elemű, rekurzívan beágyazott magpontkészletek. A nagy körök a magpontokat jelölik, a kis körök pedig a 2×2-es rácson megismételt magpontmintát. A sűrű mintát bármely sarokból kinagyítva a ritka mintát kapjuk.



3. ábra: A $\mathcal{D}$ operátor a magpontokat a legközelebbi saroktól kétszeres távolságra viszi. Magpontsorozatokat a művelet ismétlésével kapunk. Az ábrán a magpontokat aszerint színeztük, hogy mely sarkot használtuk a nagyításhoz.

Legelőször, a 4. szakaszban, a magpontok elhelyezésének problémáját oldjuk meg úgy, hogy az biztosítsa a beágyazottságot. A 4.1. szakaszban azt tárgyaljuk, hogy mely magpontkészletek alkotnak jó minőségű vonás-mintákat, és hogyan tudjuk ezeket generálni. A vonás szélességének, hosszúságának és irányának meghatározása az 5. szakasz témája.

4. Magpont-generálás

A rekurzív beágyazottság követelményéből, és a magpontkészlet véges voltából egyértelműen következik magpontok pozíciónak meghatározására használt matematikai konstrukció.

Legyen az összes magpont halmaza $S = \{\boldsymbol{s}_0, \ldots, \boldsymbol{s}_i, \ldots, \boldsymbol{s}_{N-1}\}$, ahol $\boldsymbol{s}_i = (s_{iu}, s_{iv})$. Ezek a pozíciók *magtérben* adottak. Később, hogy az UV-térbeli helyzetet megkapjuk, a *magtérbeli* pozíciókat az aktuális részletességi szintnek megfelelően fogjuk skálázni, és a mintát végtelenszer ismételni.

Ahhoz, hogy az egységnégyzeten belül a negyedekre való ráközelítést megragadjuk, bevezetjük a $\mathcal{D}$ operátort:

$$\mathcal{D}\boldsymbol{s} = \langle 2\boldsymbol{s} \rangle,$$

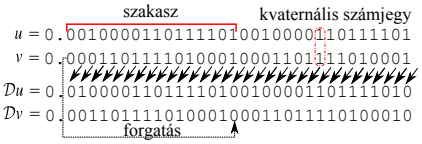
ahol a hegyes zárójelek a törtrész-képzést jelentik. Geometriai értelemben ez a művelet az egységnégyzet legközelebbi sarkától vett kétszeres középpontos nagyítást jelent. Azt adja meg, hova kerül egy magpont, ha ráközelítünk az őt tartalmazó negyedre. A rekurzív beágyazottság megköveteli, hogy az így kapott pozíció essen egybe egy másik magponttal, hiszen a nagyított verzió része kell legyen a teljes mintának.

Eszerint, ha az $\boldsymbol{s}_i$ egy ismert magpont, akkor annak $\mathcal{D}\boldsymbol{s}_i$ képe is magpont kell legyen (3. ábra). Egy N magpontból álló sorozat a következőképp áll elő:

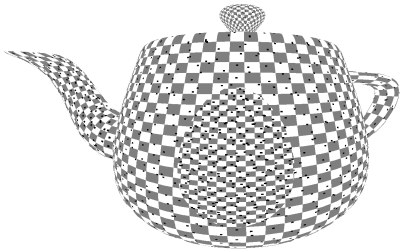
$$\boldsymbol{s}_{i+1} = \mathcal{D}\boldsymbol{s}_i, \text{ ha } 0 \leq i < N-1.$$

Mivel azonban a magpontkészlet véges, $\mathcal{D}\boldsymbol{s}_{N-1}$ is S-ben van. Ez akkor lehet igaz, ha $\boldsymbol{s}_0 = \mathcal{D}\boldsymbol{s}_{N-1}$.

Nyilvánvaló a kapcsolat az iterált függvényrendszerekkel (IFS) [2], illetve azok attraktorainak *káoszjátékkal* történő előállításával [3]. A mi konstrukciónkhoz hasonlóan, a káoszjáték is egy kezdeti pontot transzformál ismétlődően. A transzformációt véletlenszerűen választja ki egy adott halmazból. Akkor, ha a transzformációk az attraktort diszjunkt területekhez rendelik, egy adott pontról egyértelműen meghatározható, mi volt az őt generáló legutolsó transzformáció. Ekkor azonban ebből a pontból kiindulva a teljes sorozat determinisztikusan visszakövethető. Mi a káoszjáték ezen determinisztikus változatát játsszuk négy, az egységnégyzetet annak negyedeibe leképező transzformációval. A rendszerünk attraktora maga az egységnégyzet. Arra kell csak ügyelnünk, hogy a sorozat visszatérjen önmagába, ezzel egy véges elemszámú magpontkészletet kapjunk.



4. ábra: A $\mathcal{D}$ operator értelmezése bináris törteken. Szakaszos bináris törtek esetén a bitek forgathatóak.



5. ábra: A de Bruijn sorozattal generált magpontok egyenletesen helyezkednek el abban az értelemben, hogy minden rácscellába egy magpont kerül.

Ahhoz, hogy ilyen zárt ciklust eredményező kezdeti pontot találjunk, meg kell vizsgálnunk a magpont-koordináták bináris tört alakját. A $\mathcal{D}$ operátor törli a bináris pont utáni első bitet, majd a teljes sorozatot eggyel balra tolja(4. ábra). Így a fenti magpont-generáló módszert úgy is felfoghatjuk, hogy az s_{0u} és s_{0v} koordináták bitjeit minden lépésben balra toljuk. N lépés után a sorozatnak vissza kell térnie $\boldsymbol{s}_0$-ba. Ez akkor lehetséges, ha s_{0u} és s_{0v} szakaszos bináris törtek, ahol a szakaszhossz N (vagy N osztója). Bármely ilyen szakaszos bináris tört rekurzívan beágyazott magpontkészletet határoz meg, de nem mindegyik ilyen készlet magpontjainak eloszlása egyenletes és izotrópikus. Ezért a 4.1. szakaszban módszert javasolunk arra, jó minőségű magpontkészletet adó bináris törteket találjunk.

4.1. Egyenletesen elosztott magpontkészletek

Ha egymástól függetlenül, véletlen módon, minden számjegyet azonos valószínűséggel választva építünk végtelen sorozatot, minden lehetséges adott hosszú számjegysor ugyanolyan gyakorisággal fordul elő. Az olyan véges sorozatot, amire ez szintén teljesül, *de Bruijn* sorozatnak nevezik [14]. Az u és v bitjeit összevonhatjuk, hogy kvaternális számjegyeket alkossanak, s így egy kvaternális számsorozatot kapjunk. Ahhoz, hogy a magpontjaink egyenletes eloszlást mutassanak, kvaternális de Bruijn sorozatot kell keresnünk.

Geometriai értelemben az első kvaternális számjegy értéke azt jelzi, hogy az egységnégyzeten belül mely negyedben helyezkedik el a magpont. A következő számjegy adja meg, mely $\frac{1}{4} \times \frac{1}{4}$-es négyzetben található meg ezen belül, és így tovább. Amikor a magpontokat a bitek forgatásával előállítjuk, egy minta pontosan annyiszor fordul elő a bináris pont után, ahány magpont a mintához tartozó négyzetben van. Ha $N = 4^K$ valamely K egész számmal, akkor pontosan egy magpont fog esni a $2^K \times 2^K$ rács minden cellájába (5. ábra). Ez nagyon hasonló az alacsony diszkrepanciájú Halton sorozat elemi intervallum tulajdonságához [7]. A de Bruijn sorozat előállítására számtalan hatékony algoritmus ismert [14], ezért ennek ismertetésétől eltekintünk.

5. Megjelenítés

A magpont-mintázatot a felületen megfelelő léptékben ismételve biztosíthatjuk a megkívánt képtérbeli vonalméretet és -sűrűséget. A szükséges lépték a felületen változó, így a mintának időben és térben folytonosan alkalmazkodnia kell. Ezt úgy érjük el, hogy először egy közelítő kettőhatvánnyal skálázzuk a magpont-mintázatot (5.1. szakasz), majd a közelítés hibáját kiküszöbölve magukat a vonásokat is skálázzuk (5.2. szakasz), valamint vonások kihalványításával biztosítjuk a különböző sűrűségek közötti átmenetet(5.3. szakasz).

5.1. A magpont-mintázat skálázása

A részletességi szint kiválasztásának folyamata a *mipmapping* [17] eljárással analóg, azzal a különbséggel, hogy esetünkben minden részletességi szint azonos, de kettő valamely hatványával skálázott. Másodszor, a részletességi szint választása nem előre számított textúrák közüli választást jelent, hanem azt dönti el, milyen skálán ismételjük a magpontokat az UV-térben. Mivel a végcél az adott képtérbeli sűrűség elérése, ez a mag- és UV-terek közötti leképezés a lehető legnagyobb mértékben ki kell, hogy egyenlítse az UV- és képterek közötti leképezést. Ezek a leképezések globálisan ugyan nem lineárisak, sőt nem is minden pontban definiáltak, de egy felületi pont vonatkozásában mindig értelmezhetőek, akár lineáris transzformációként is.

Legyen $\boldsymbol{L}$ a keresett, ismeretlen, magpontokat textúrakoordinátákhoz rendelő lokális leképezés:

$$\boldsymbol{x}_{\mathrm{uv}} = \boldsymbol{L}\boldsymbol{x}_{\mathrm{s}},$$

ahol $\boldsymbol{x}_\mathrm{s}$ magtérbeli pozíció, $\boldsymbol{x}_\mathrm{uv}$ a textúra-koordináták, és $\boldsymbol{L}$ kettőhatvánnyal történő izotróp skálázás kell legyen. Célunk az, hogy a skálatényezőt bármely felületi pontra megtalálhassuk. Jelen levezetésünkben előbb megkötés nélkül keresünk skálatényezőt, azután fogjuk az azt közelítő kettőhatványt kiválasztani.

Legyen $\boldsymbol{T}$ a textúra-leképezés operátorának lokális inverze. Ezt a modell textúra-paraméterezése határozza meg. Így

$$\boldsymbol{x}_\mathrm{obj} = \boldsymbol{T}\boldsymbol{x}_\mathrm{uv},$$

ahol $\boldsymbol{x}_\mathrm{obj}$ a modelltérbeli pozíció. Ez nem más, mint a jól ismert tangenstérből modelltérbe vivő transzformáció, ahol a tangens- és binormálvektorok a modelltérbeli pozíció u és v szerinti parciális deriváltjai, normalizálás nélkül.

Legyen $\boldsymbol{G}$ a képalkotási csővezeték teljes, modelltérből nézetablaktérbe vivő leképezése. Ebben a modellezési, kamera-, vetítési és nézetablak-transzformációk egyaránt benne foglaltatnak. Az objektumok és a kamera paraméterei ezeket egyértelműen megadják. Így

$$\boldsymbol{x}_\mathrm{vp} = \boldsymbol{G}\boldsymbol{x}_\mathrm{obj},$$

ahol $\boldsymbol{x}_\mathrm{vp}$ a nézetablaktérbeli pozíció.

Ezekkel a magtérből nézetablaktérbe vivő transzformáció a következőképpen írható:

$$\boldsymbol{x}_\mathrm{vp} = \boldsymbol{G}\boldsymbol{T}\boldsymbol{L}\boldsymbol{x}_\mathrm{s}. \tag{1}$$

Ismét megjegyezzük, hogy minden művelet függ a felületi ponttól, és mindegyik a globális leképezések lokális linearizációja.

Legyen $\boldsymbol{h}$ a *részletességi irány*, egy differenciális irányvektor a magtérben, ami a vonások irányára merőleges. Számunkra az az érdekes, hogy a részletességi irányt hogyan skálázza az $\boldsymbol{L}$, $\boldsymbol{T}$ és $\boldsymbol{G}$ leképezés. Legyenek ezek a skálatényezők az L részletességi tényező, a T textúratorzítás, és a G geometriai faktor:

$$L = \frac{|\boldsymbol{L}\boldsymbol{h}|}{|\boldsymbol{h}|}, \quad T = \frac{|\boldsymbol{T}\boldsymbol{L}\boldsymbol{h}|}{|\boldsymbol{L}\boldsymbol{h}|}, \quad G = \frac{|\boldsymbol{G}\boldsymbol{T}\boldsymbol{L}\boldsymbol{h}|}{|\boldsymbol{T}\boldsymbol{L}\boldsymbol{h}|}.$$

Ezekkel az 1. egyenletet differenciálisokra alkalmazhatjuk, így a részletességi irány skálázására a következő formulát kapjuk:

$$|\boldsymbol{h}_\mathrm{vp}| = GTL|\boldsymbol{h}|,$$

ahol $|\boldsymbol{h}_\mathrm{vp}|$ a nézetablakban megjelenő részletességi irányvektor hossza. A G geometriai faktort és a T textúratorzítást könnyen számolhatjuk (a képleteket nem ismertetjük, mivel a modellezési-nézeti-vetítési és a tangenstérhez kapcsolódó transzformációk egyaránt jól ismertek), az L pedig az a skálafaktor, ami a részletességi szint választásából kell következzen.

Az $F = |\boldsymbol{h}|/|\boldsymbol{h}_\mathrm{vp}|$ arány ragadja meg azt, hogy a magpontok a nézetablakban milyen sűrűn jelennek meg. Ez szabadon választható művészi paraméter, és globális konstans, mivel az árnyalatoknak a sűrűség módosításával való visszaadását itt még nem vesszük figyelembe. Akárcsak a hagyományos textúrázásnál, alacsonyabb F érték választása nagyobb részletességet eredményez — vagyis

több mag, így több vonás esik egységnyi felületre —, de a minta gyorsabb ismétlődését is. Ezzel a megkívánt L részletességi tényező így fejezhető ki:

$$L = (GTF)^{-1}.$$

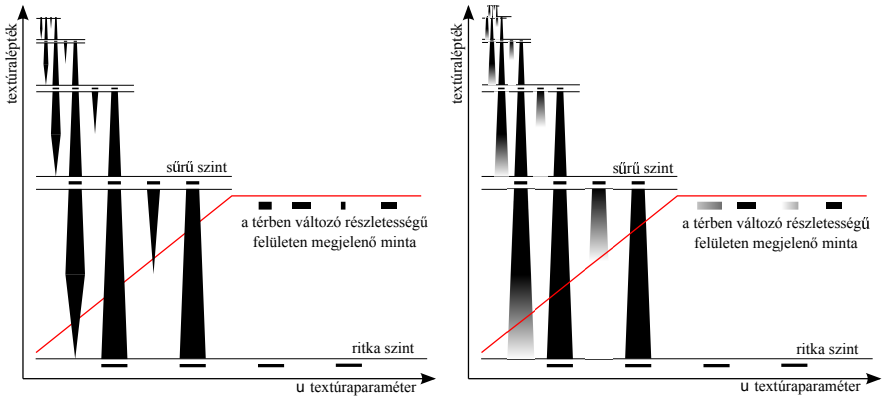
Hogy a magpontok beágyazottságát kihasználhassuk, L kettőhatvány kell legyen, így $L \approx 2^{\lfloor M \rfloor}$, ahol az $\lfloor M \rfloor$ egész számot *beágyazási szintnek* nevezzük. Először az M valós számot számítjuk ki:

$$M = \log_2 L = \log_2 (GTF)^{-1} = -\log_2 GTF,$$

ezután ennek egészrészét vesszük, hogy megkapjuk az $\lfloor M \rfloor$ beágyazási szintet. Ezzel a magtérbeli és a textúra-koordináták közötti skálatényező a következőnek adódik:

$$\boldsymbol{x}_{\mathrm{s}} = 2^{-\lfloor M \rfloor} \boldsymbol{x}_{\mathrm{uv}}.$$

A megoldás pontos, amikor M egész szám, és az egész értékek *beágyazási szinteket* határoznak meg. Az M nem-egész értékeire, a $\lfloor M \rfloor$ *sűrű szint* folytonosan kell, hogy átmenjen a $\lfloor M \rfloor + 1$ *ritka szintbe*. Ezért a vonásokat alkalmasan kell skáláznunk, és azokat, amelyek a ritka szinten nem láthatóak, el kell halványítanunk (6. ábra).



6. ábra: A beágyazási szintek közötti sima átmenet megvalósítása a méret (jobb), illetve az átlátszóság (bal) módosításával, 2D ábrán.

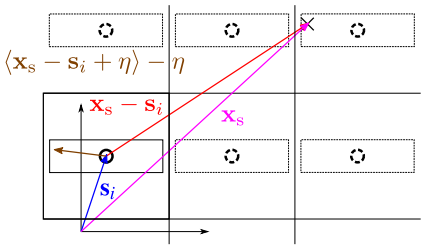
5.2. A vonásméret interpolációja

A vonások nézetablaktérbeli hossza és szélessége művészi paraméter, és az $\boldsymbol{e}$ kétdimenziós vektorral adjuk meg. Az F definíciója szerint tudjuk, hogy a magtérbeli vonásméret $F\boldsymbol{e}$ kellene legyen, ha az L nem lenne kettőhatványokra

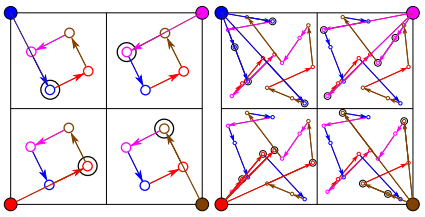
kvantálva. A kvantálás hatásának ellensúlyozására a magtérbeli vonásméreteket az alábbi tényezővel skálázzuk:

$$\frac{2^M}{2^{\lfloor M \rfloor}} = 2^{M-\lfloor M \rfloor} = 2^{\langle M \rangle} = 2^m,$$

ahol m tekinthető a beágyazási szintek közötti *interpolációs tényezőnek*, ami a sűrű szinten felvett 0 értéktől a ritka szinten felvett 1 értékig változik. Tudjuk, hogy a sűrű és ritka szintek a kétszeres nagyítástól eltekintve azonosak, így könnyen látható, miért kell a vonásoknak kétszeresére nőniük az interpolációs tényező növekedésével.



7. ábra: A vonástérbeli pozíció számítása (skálázás és forgatás nélkül). × jelöli az árnyalt pontot. Ennek magtérbeli pozíciója $\boldsymbol{x}_s$, az éppen feldolgozott magpont $\boldsymbol{s}_i$, a magponthoz képesti pozíció $\boldsymbol{x}_s - \boldsymbol{s}_i$, amit a $\langle \boldsymbol{x}_s - \boldsymbol{s}_i + \boldsymbol{\eta} \rangle - \boldsymbol{\eta}$ kifejezéssel a magpont körüli egységnégyzetbe képezünk.



8. ábra: A kék, sűrű csempén a kék magpontok egybeesnek a ritka magpontokkal. A kék csempében a kék nyíl egyszerre jelent egy lépést a sűrű csempe káoszjátékában és egy sűrű magpont ritka magponthoz rendelését.

Ahhoz, hogy az $\boldsymbol{x}_s$ pontban megállapíthassuk a vonások hatására előálló színt, meg kell találnunk, mely vonások fednek rá a pontra, és azok textúrájának mely eleme esik rájuk. Így minden $\boldsymbol{s}_i$ magpontra meg kell találnunk az $\boldsymbol{x}_s$ magtérbeli ponthoz tartozó $\boldsymbol{z}_i$ *vonástérbeli koordinátákat* (7. ábra). Az $\boldsymbol{x}_s$ árnyalt pontnak a magponthoz relatív pozíciója $\boldsymbol{x}_s - \boldsymbol{s}_i$, de a magokat végtelenszer ismételjük, hogy a teljes síkot lefedjék. Így a $\boldsymbol{x}_s - \boldsymbol{s}_i$ tartalmaz egy egészértékű eltolást, ami ahhoz az egység oldalhosszú csempéhez tartozik, amiben a magpont elhelyezkedik. Azt szeretnénk, hogy a vonások középpontjai essenek a magpontokra, ezért az egészértékű eltolás kiiktatásához a $\boldsymbol{x}_s - \boldsymbol{s}_i$ pontot az origó-középpontú egységnégyzetre képezzük a $\langle \boldsymbol{x}_s - \boldsymbol{s}_i + \boldsymbol{\eta} \rangle - \boldsymbol{\eta}$ kifejezést kapva, ahol $\boldsymbol{\eta} = (1/2, 1/2)$. Ez az $\boldsymbol{x}_s$ pont relatív pozícióját az $\boldsymbol{s}_i$ magpont legközelebbi példányához képest adja meg.

Ezt a vonás irányának megfelelően forgathatjuk, szélességének és hosszának megfelelően skálázhatjuk, így a következő képletet kapjuk a vonástérbeli koordinátákra:

$$\boldsymbol{z}_i = (\boldsymbol{R}\,(\langle \boldsymbol{x}_s - \boldsymbol{s}_i + \boldsymbol{\eta} \rangle - \boldsymbol{\eta}))\,\varnothing(2^m F\boldsymbol{e}) + \boldsymbol{\eta},$$

ahol $\boldsymbol{R}$ a vonásiránynak megfelelő elforgatási mátrix, a $\varnothing$ operátor pedig az elemenkénti osztást jelöli. Ezek a koordináták használhatóak a vonástextúra címzésére.

5.3. Sűrűség-interpoláció

Amikor eltávolodunk egy felülettől, ahogy az objektumtérbeli magpontsűrűségnek csökkennie kell, úgy látjuk a vonásmintát a sűrű szintből a ritka szintbe átmenni. Néhány vonás textúratérbeli mérete növekszik, hogy a nézetablaktérben megőrizzék kiterjedésüket, míg más vonások el kell tűnjenek, hogy a vonalkázás ritkuljon. Amikor két szint között interpolálunk, azonosítanunk kell a maradó vonásokat.

Négy sűrű csempe fed le egy ritka csempét (8. ábra). Tekintsük a bal felső (a kék sarokhoz tartozó) csempében levő sűrű magpontokat. A káoszjáték folyamán ezen magpontok mindegyikét egy másik sűrű magpont skálázott képeként állítottuk elő, némelyiküket — a kék magpontokat — a bal felső sarkot skálázási középpontnak használva. Így a kék magpontok és a ritka magpontok egyaránt a sűrű magpontok kék sarokból kinagyított képei, vagyis egybeesnek. Ugyanez a gondolatmenet minden csempére és sarokra megismételhető. Így tehát az, hogy egy sűrű magpont egybeesik-e egy ritka magponttal, két indikátor egybeesésétől függ: a négy sűrű csempe melyikében helyezkedik el, illetve a négy lehetséges skálázás melyike állította elő a káoszjátékban. A sűrű csempe sor- és oszlopindexeinek paritásbitjei jelzik, melyik csempében van a ritka csempén belül. A csempe indexe nem más, mint a $\langle \boldsymbol{x}_\mathrm{s} - \boldsymbol{s}_i + \boldsymbol{\eta} \rangle$ képlettel eldobott egészrész, vagyis $\boldsymbol{w} = \lfloor \boldsymbol{x}_\mathrm{s} - \boldsymbol{s}_i + \boldsymbol{\eta} \rfloor$.

Hogy mely skálázást alkalmazzuk a káoszjáték során, az az s_{iu} és s_{iv} első bitjeitől függ. A bitek forgatásával ezek az új magpont bitmintájának utolsó bitjei lesznek. Így tehát a sűrű magpont ritka magponttal akkor és csak akkor esik egybe, ha az u és v bitminták paritásai ugyanazok, mint a $\boldsymbol{w}$ sor- és oszlopindexek paritásai. Amikor távolodunk a felülettől, ezek a magpontok maradnak meg, míg a többi magpontot az m interpolációs tényező növekedésével el kell tüntetni.

A vonások eltüntetése többféleképp megoldható. Modulálhatjuk az átlátszóságot vagy a vonásméretet, valamint az interpolációs tényező növekedésével egyszerre vagy egymás után is eltüntethetjük a vonásokat. A vonások egyidejű modulációja sima átmenetet tesz lehetővé hirtelen megjelenő vagy eltűnő vonások nélkül, de a vonások nagy része félig átlátszó vagy köztes méretű lesz, így a vonalkázás kevésbé lesz egyenletes. Ha a vonalak egymást követően jelennek meg, a moduláció módszere alig számít, mivel hirtelen tűnnek elő, de mind azonosnak látszanak. A rekurzív beágyazás miatt a vonások csak akkor jelennek meg vagy tűnnek el, ha a vonalkázásnak sűrűbbé vagy ritkábbá kell válnia, így villódzás nem jelentkezik.

5.4. Árnyalás

A megvilágítás ábrázolására a lokálisan megkívánt árnyalatnak megfelelően kell módosítani a vonalkázás sűrűségét. Ahogy a vonalakat a kisebb részletesség

kedvéért el tudtuk tüntetni, úgy méretük és átlátszóságuk az árnyalatnak megfelelően is modulálható. A vonásokat itt is halványíthatjuk egyszerre (simább animációt eredményezően, de sok félig látható vonással), vagy egymás után (konzisztensebb megjelenésű, de hirtelen megjelenő vonásokkal). A művészi gyakorlatban az árnyalatokat gyakran úgy hangsúlyozzák, hogy nagyjából négy különálló, szögben álló rétegben rajzolnak vonásokat [8], ami *keresztvonalkázásként* ismert. Ennek szimulációjához több önhasonló magpont-készletet használunk.

6. Implementáció

Ha egy megfelelő magpont-halmazt u és v bitminták formájában előzetesen előállítottunk, az algoritmust (1. algoritmus) egyetlen árnyalóprogramban megvalósíthatjuk. Didaktikai okokból és a könnyebb érthetőség kedvéért a pszeudokódot a következő egyszerűsítésekkel írtuk le:

Nem használunk többrétegű vonalkázást, de erre az algoritmus könnyedén kiterjeszthető, úgy, hogy a kívánt árnyalatot a rétegekhez rendelt tartományokból képezzük, és az algoritmust minden rétegre különböző bitmintákkal, vonásméretekkel, forgatásokkal és vonástextúrákkal végrehajtjuk.

Egyidejű modulációt használunk, tehát nem egymás után halványítjuk a vonásokat, a részletesség és az árnyalat esetében egyaránt. Ha egyesével szeretnénk halványítani a vonásokat, egy smoothstep függvényt kell alkalmaznunk az a árnyalatra és/vagy az m interpolációs tényezőre, az $[i/(N+1), (i+1)/(N+1)]$ intervallumon, a 11. és/vagy 14. sorokban.

Az átlátszóságot moduláljuk, de a vonásméret modulációja is hasonló.

Minden magpontot feldolgozunk a nyers erő módszerével. A gyakorlatban csupán néhány vonás befolyásolja egy felületi pont színét, így a 9. sorbéli hurok lecserélhető a 6.1. szakaszban leírt gazdaságosabb megoldásra.

6.1. A magpontok előszűrése

A nyers erővel dolgozó implementáció vonalkázási rétegenként N textúramintát használ. Négy réteggel és 64 elemű magpont-halmazzal ez képpontonként 256 mintát jelent. Ennek ellenére a naív implementáció nem teljesít olyan gyengén, mint várnánk. A 256 minta ugyanabból a feltételezhetően kicsi, hatékonyan gyorsítótárazott textúrából származik. A textúraolvasások nagy része túlcímzés miatt memóriaolvasás nélkül megoldható, és a többletszámítást a valóban bekövetkező memóriaolvasások késleltetése eltakarja. Ezzel együtt a megközelítés teljes potenciálja akkor érhető el, ha csak azokat a vonásokat kell feldolgozni, melyek egy felületi pont árnyalásához hozzájárulhatnak.

Ezek azonosításához egy *vonásfedési textúrát* hozunk létre, amely a magtérbeli egységnégyzetet ábrázolja, és minden texelben az átfedő vonások listáját tároljuk. Amikor a szintek között interpolálunk, a vonásokat legfeljebb kétszeresükre skálázzuk. Így azokat a maximális méretben kell rajzolni. A legegyszerűbb úgy összegyűjteni a texelekbe a magpontok azonosítóit, hogy bitmaszkok bufferébe

Algorithm 1 Egy felületi pont árnyalása. Globális bemenetek a magpontkészletet definiáló $\boldsymbol{b} = (b_u, b_v)$ de Bruijn bitsorozatok, és a következő művészi paraméterek: a vonalkához tartozó elforgatási mátrix $\boldsymbol{R}$, a globális nézetablak-beli magpont-sűrűség F, a vonalka szélesség és hosszúság $\boldsymbol{e}$. A művész által rajzolt vonalka textúrája a *strokeTex* amely nullával tér vissza ha a határain kívül mintavételezzük.

function SHADE(texture coords $\boldsymbol{x}_{\text{uv}}$, position $\boldsymbol{x}_{\text{obj}}$)
  $a \leftarrow$ a megvilágításból adódó árnyalat $[0, 1]$
  $\boldsymbol{c} \leftarrow \boldsymbol{1}$ ▷ kezdetben a pixel a háttér színét veszi fel
  $T \leftarrow$ a textúra torzítása az $\boldsymbol{x}_{\text{obj}}$ pontban
  $G \leftarrow$ a geometriai faktor az $\boldsymbol{x}_{\text{obj}}$ pontban
  $M \leftarrow -\log_2 GTF$ ▷ beágyazottsági mélység
  $\boldsymbol{x}_{\text{s}} \leftarrow 2^{-\lfloor M \rfloor}\boldsymbol{x}_{\text{uv}}$ ▷ magponttérbeli pozíció az UV-ből
  $m \leftarrow M - \lfloor M \rfloor$ ▷ interpolációs tényező
  for $i \leftarrow 0, N-1$ **do** ▷ valamennyi magpontra
    $\boldsymbol{s}_i \leftarrow (0.\boldsymbol{b}_u\boldsymbol{b}_u \ldots, 0.\boldsymbol{b}_v\boldsymbol{b}_v \ldots)$ ▷ a magpont bitjei
    $\alpha \leftarrow a$ ▷ átlátszóság az árnyalatból
    $\boldsymbol{w} \leftarrow \lfloor \boldsymbol{x}_{\text{s}} - \boldsymbol{s}_i + \boldsymbol{\eta} \rfloor$ ▷ a sűrű csempe indexe
    if $\boldsymbol{w} \not\equiv \boldsymbol{b} (\text{mod} 2)$ **then** ▷ a magpont nincs benne a ritkább mintában
      $\alpha \leftarrow \alpha(1-m)$ ▷ átlátszóság a részletességi szintből
    $\boldsymbol{z}_i \leftarrow (\boldsymbol{R}\,(\langle \boldsymbol{x}_{\text{s}} - \boldsymbol{s}_i + \boldsymbol{\eta} \rangle - \boldsymbol{\eta}))\,\varnothing(2^m F\boldsymbol{e}) + \boldsymbol{\eta}$
    $\boldsymbol{y} \leftarrow \text{strokeTex}[\boldsymbol{z}_i]$ ▷ textúra-mintavételezés
    $\alpha \leftarrow \alpha y_\alpha$ ▷ az alpha érték alkalmazása a textúrán
    $\boldsymbol{c} \leftarrow (1-\alpha)\boldsymbol{c} + \alpha\boldsymbol{y}$ ▷ alpha blending
    $\boldsymbol{b} \leftarrow \boldsymbol{b} \circlearrowleft 1$ ▷ a bitek körkörös eltolása a következő magponthoz
  return $\boldsymbol{c}$

renderelünk, ahol minden bit egy magpontot jelöl, és *atomi bitenkénti vagy* műveleteket használunk. Egy második menetben a bitmaszkokat azonosítólistákká alakíthatjuk, melyek kevesebb bitet foglalnak.

A textúrát csak akkor kell frissíteni, ha a művészi paraméterek megváltoznak. Az árnyaló algoritmusba egyszerűen beépíthetjük, hogy olvassa ki a fedési textúrát az $\boldsymbol{x}_{\text{s}}$ pontban, és csak a releváns magpontokat dolgozza fel.

7. Eredmények

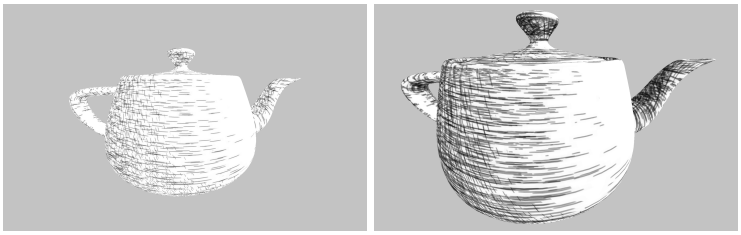
7.1. Teljesítmény

Egy NVIDIA GeForce 780-as kártyán 1920×1200 teljes képernyős felbontáson teszteltük az algoritmust, és egy képkocka rajzolásának idejét mértük. Összevetettük módszerünket a dinamikus tömör textúrákkal (DST) [4] és a TAM [13] módszerrel, a szerzők által közzétett programkódokat használva. Az RPTAM-hoz négy réteget használtunk, rétegenként 16, illetve 64 magponttal (9. ábra). Szürkeskálás textúrákat használtunk minden módszerben. A 1. táblázat eredményei azt mutatják, hogy bár a nyers erőt használó implementáció valós idejű teljesítményt nyújt, minden más módszer rajzolási ideje gyakorlatilag elhanyagol-

ható, és a textúra-hozzáférés sávszélessége a meghatározó. Míg a nyers erőt használó módszer lineárisan skálázódik a magpontok számával, a fedési textúrát használó gyorsítás ezt a problémát megszünteti. Bár az RPTAM így is lassabb, mint a TAM, a különbségnek mai hardveren nincs jelentősége.

DST	TAM	B16	C16	B64	C64
0.54	0.28	4.80	1.66	16.6	1.81

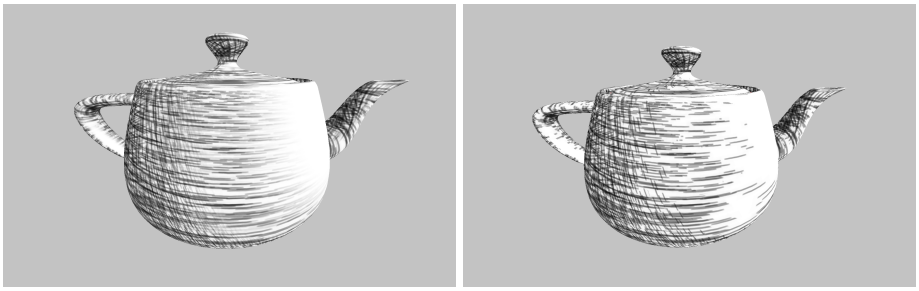
1. táblázat: Egy képkockára eső renderelési idő ms-ban, 1920×1200-as felbontás mellett. A B a nyers erő módszerét, a C a fedési textúrát jelenti, a szám a magpontok száma rétegenként.



9. ábra: Teáskanna-modell 4 darab, 16 magpontos réteggel, 600 FPS (bal), és 4 darab, 64 magpontos réteggel, 500 FPS (jobb). A képernyőfelbontás: 1920×1200.

7.2. Minőség

A módszerünk, akárcsak a dinamikus tömör textúrák módszere, vagy a TAM, kifogástalan időbeli koherenciát biztosít. A dinamikus tömör textúrázás a vonalkázást egyfajta *bináris stílussal* képes közelíteni, míg a mi módszerünk stilizált vonásokkal is tud dolgozni. A TAM a mi módszerünkkel minőségben ekvivalens, de nem teszi lehetővé a végtelen nagyítást. A TAM-ok kézzel szerkeszthetőek, vagy automatikusan is generálhatóak, vonások véletlenszerű beillesztésével, a korábbi vonásokkal ütköző vonások eldobásával. A TAM előállítási módszere hosszadalmas próbálkozásos keresés, míg a mi magpontjaink előállítására lineáris idejű algoritmus adható. Mindkét módszer egyenletes eloszlású vonásokat hoz létre. A mi módszerünk geometriai értelemben garantálja az egyenletességet. Ennél fontosabb az, hogy a TAM-mal szemben a magpontokat nem kell újra előállítani, ha a művészi paraméterek, például a vonáshossz, vagy a vonástextúra, változnak. Így ezek akár animálhatók is. A vonásokat egyidejűleg is halványíthatjuk így a TAM-hoz hasonló megjelenítést elérve (10 ábrán balra), de egyenként is (10 ábrán jobbra), ami egyedi képesség a textúra alapú vonalkázási módszerek között.



10. ábra: Teáskanna-modell egyidejű vonalka-elhalványítással (balra), illetve a vonások elhalványítása egyenként (jobbra).

8. Továbblépési lehetőségek

Amikor egy felület egész részletességi tényezővel látszik, minden vonás teljesen látható. Egyébként néhány közülük elhalványodás közben átmeneti állapotban van. Ez a különbség észrevehetetlen, amikor a vonások egyenként halványodnak, de a stílus enyhe periodikus módosulása figyelhető meg egyidejű halványítás esetén. Ezért szeretnénk kiterjeszteni a beágyazottság fogalmát súlyozott magpont-halmazokra, ahol az interpolált halmazok ugyanolyan súlyeloszlással rendelkeznek, így az egész szintek vonásmintái az interpoláltaktól megkülönböztethetetlenek. Úgy gondoljuk, ez egyben egy új művészi paraméter bevezetését is lehetővé fogja tenni, ami az egyidejű és egyenkénti halványítás közötti köztes stratégiákat is lehetővé tesz. Ebben az átlapolt halványítási modellben a vonások egy szabályozható, de állandó százaléka lesz átmeneti állapotban bármely időpillanatban és részletességi szinten.

Azt is tervezzük, hogy a körvonal-rajzolási technikákkal való együttműködést vizsgáljuk, és felmérjük, hogy képtérbeli szűréssel a vonások túlhúzása szimulálható-e.

Köszönetnyilvánítás

Köszönjük az OTKA PD-104710 és az MTA Bolyai János ösztöndíjának támogatását.

Irodalom

1. Zainab AlMeraj, Brian Wyvill, Tobias Isenberg, Amy A Gooch, and Richard Guy. Automatically mimicking unique hand-drawn pencil lines. *Computers & Graphics*, 33(4):496–508, 2009.
2. Michael F Barnsley and Stephen Demko. Iterated function systems and the global construction of fractals. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 399(1817):243–275, 1985.

3. Michael F Barnsley and Andrew Vince. The chaos game on a general iterated function system. *Ergodic Theory and Dynamical Systems*, 31(04):1073–1079, 2011.
4. Pierre Bénard, Adrien Bousseau, and Joëlle Thollot. Dynamic solid textures for real-time coherent stylization. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, pages 121–127. ACM, 2009.
5. Derek Cornish, Andrea Rowan, and David Luebke. View-dependent particles for interactive non-photorealistic rendering. In *Graphics interface*, volume 1, pages 151–158, 2001.
6. Paul Haeberli. Paint by numbers: Abstract image representations. In *ACM SIGGRAPH Computer Graphics*, volume 24, pages 207–214. ACM, 1990.
7. John H Halton. Algorithm 247: Radical-inverse quasi-random point sequence. *Communications of the ACM*, 7(12):701–702, 1964.
8. A. Hertzmann and D. Zorin. Illustrating smooth surfaces. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 517–526. ACM Press/Addison-Wesley Publishing Co., 2000.
9. Pierre-Marc Jodoin, Emric Epstein, Martin Granger-Piché, and Victor Ostromoukhov. Hatching by example: a statistical approach. In *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, pages 29–36. ACM, 2002.
10. Hyunjun Lee, Sungtae Kwon, and Seungyong Lee. Real-time pencil rendering. In *Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, pages 37–45. ACM, 2006.
11. Barbara J Meier. Painterly rendering for animation. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 477–484. ACM, 1996.
12. Afonso Paiva, Emilio Vital Brazil, Fabiano Petronetto, and Mario Costa Sousa. Fluid-based hatching for tone mapping in line illustrations. *The Visual Computer*, 25(5-7):519–527, 2009.
13. Emil Praun, Hugues Hoppe, Matthew Webb, and Adam Finkelstein. Real-time hatching. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, page 581. ACM, 2001.
14. Meltem Sonmez Turan. Evolutionary construction of De Bruijn sequences. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 81–86. ACM, 2011.
15. Thomas Strothotte and Stefan Schlechtweg. *Non-photorealistic computer graphics: modeling, rendering, and animation*. Elsevier, 2002.
16. Tamás Umenhoffer, László Szécsi, and László Szirmay-Kalos. Hatching for motion picture production. In *Computer Graphics Forum*, volume 30, pages 533–542. Wiley Online Library, 2011.
17. Lance Williams. Pyramidal parametrics. In *ACM Siggraph Computer Graphics*, volume 17, pages 1–11. ACM, 1983.
18. Andrew P Witkin and Paul S Heckbert. Using particles to sample and control implicit surfaces. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 269–277. ACM, 1994.
19. Johannes Zander, Tobias Isenberg, Stefan Schlechtweg, and Thomas Strothotte. High quality hatching. In *Computer Graphics Forum*, volume 23, pages 421–430. Wiley Online Library, 2004.