Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Irányítástechnika és Informatika Tanszék

Procedurális, animálható erdőmodellek generálása és megjelenítése

Tudományos Diákköri Konferencia
dolgozat

Készítette

Burkus Viktória
Kárpáti Attila

Konzulens

Dr. Szécsi László

2017

Tartalomjegyzék

Kivonat

A dolgozat egy olyan módszert mutat be, amely véges elemkészletből és az elemek összeillesztési lehetőségeit meghatározó szabályrendszer segítségével a grafikus kártyán hatékonyan és közvetlenül is megjeleníthető növénymodelleket képes létrehozni. Az igény azért merült fel, mert a hasonló geometriák megjelenítésére irányuló módszerek gyakran számításigényesek és nem teszik lehetővé a modellek valós idejű animációját. A megoldás támogatja a modellek csontváz alapú animációját, amely segítségével az azonos szabályok alkalmazásával létrehozott, megegyező topológiájú növények is egyedi pózt vesznek fel, ezáltal olyan animációk is megvalósíthatók, mint például a szél keltette mozgás. A létrehozott modellek felépítése eltérő, így nem keltik azt a hatást, hogy ugyanazt a modellt többször lemásolták. A fák véges számú elemkészletből épülnek fel. Ezeket előzetesen modelleztünk le egy modellező program segítségével. A számítógépen való megjelenítés hatékonyságát növelve a részletességi szintek változtatását a hardware-es tesszelláció biztosítja. Így az alkalmazásban a fákhoz közeledve az egyes elemek nagyobb részletességi szinttel rajzoldónak ki, míg távolodva az alacsonyabb részletességi szintű tesszellálás nélküli modellek jelennek meg. Javasolunk egy procedurális módszert a modellek textúrázására. Ennek célja, a szabályszerűen felépített növény-geometriák természetes hatású megjelenítése. Olyan megoldást alkalmazzunk, amely amellett, hogy nem igényel manuális textúrázást, nagy változatosságot biztosít. A textúra illeszkedik az előállított geometriához, az építőelemek közötti vágást elfedi. A lombozat megjelenítésére több módszert kínál a szakirodalom, az implementált megoldás törekszik a GPU-n való hatékony megjelenítésre.

Bevezetés

A számítógépes játékokban, illetve mindazon számítógépes vizualizációval foglalkozó alkalmazásokban, melyek során a növényzet megjelenítése is szerepet játszik, felmerül a kiterjedt növényzet-geometriák előállításának és hatékony megjelenítésének problémája. Elsősorban az előbb említett számítógépes játékoknál fordul elő, hogy költséghatékonysági okokból a kevésbé hardverigényes megoldásokra törekszenek. Hiszen egy modellező program segítségével megalkotott nagy részletességű famodell, melynek háromszöghálóját számos csúcspont alkotja, többszöri megjelenítése túlzottan erőforrás igényes lehet. Ezért ezt a megoldást lehetőleg kerülik. Az egyéb módszerek használatával a fejlesztők kompromisszumokat kötnek, a részletesen kidolgozott növényzet és a hatékony megjelenítés között.

Gyakran alkalmazott módszer a *billboard*-ok használata. Vagyis egy két dimenziós, növényzetet ábrázoló plakát behelyezése a virtuális világ színterébe, mely a kamera pozíció irányába fordulva jelenik meg. Így mindig szemből látható. Habár ezen módszer alkalmazása jelentősen csökkenti az erőforrás igényeket, sok esetben nem lehet az elvárt mértékben megközelíteni a valódi növényzet látványát. Az implementációk többségében a kamera mozgatásakor válik feltűnővé a háromdimenziós geometriák hiánya. Egy billboard-on nem szükséges a teljes növényzetet megjeleníteni. Mi billboard-okat a lombozat levélfelhőinek megjelenítésére ajánljuk. A billboard-on megjelenő képnek nem statikus képet, hanem egy a nézési iránytól függő dinamikusan előállított képet használunk fel. Ezzel a módszerrel a billbord-okat megfelelő távolságból szemlélve a kamera mozgatásakor se keltenek két dimenziós hatást a változó billboard kép miatt.

Egy további alkalmazható módszer lehet valódi háromdimenziós modellek használata különböző részletességi szintek variálásával. A kamerához közel elhelyezkedő geometriák nagy, míg a kamerától távol elhelyezkedő geometriák alacsony részletességi szintű megjelenítése. Ezáltal nagyobb vizuális élményt érhetünk el, mint csak alacsony felbontású modellek használatával. Valamint kisebb erőforrásigény érhető el, mint csak nagy részletességi szintű modellek használata esetén. A modellek több részletességi szintű megjelenítésére kézenfekvő megoldás, hogy egy adott modellt több részletességi szinten is megmodellez az alkotója. Mi hardware-es tesszellációt használunk. A használt modelljeinket alacsony részletességűre modelleztük. Mivel így többszöri megjelenítésük is hatékony, alkalmasak a kamerától távol elhelyezkedő elemek megjelenítésére. A közeli elemek nagy részletességű

megjelenítésekor használjuk a GPU által nyújtott tesszellációs lehetőségeket. Vagyis a modell vertexei mellé új vertexeket generálunk, így a fa elem modellek nem keltenek szögletes hatást, az élek lekerekednek.

Növényzet, főként fák és ágak, animálására egy lehetőség a csontvázalapú karkteranimáció. Azt általunk módosított és megvalósított formája nagyban hasonlít a Pirk és szerzőtársai által javasolt Plastic Trees [8] módszerben használthoz, ahol a csontvázanimációt szintén fák ágainak és törzseinek animálására használták. Velük ellentétben a mi célunk nem biológiai hatásokból következő lombformák megjelenítése. Számunkra két fontos célja van az animációnak. Egyrészt az elemek ízületeit változatosabb kezdőpozícióba tudjuk állítani, ezáltal az azonos elemkonfigurációt használó modellek eltérő megjelenésűek lehetnek. Másrészt a kezdőpozícióhoz viszonyítva további, általában finomabb, változásokkal tudjuk szimulálni a szél keltette mozgásokat.

Sok egyforma elem megjelenítésére alkalmas módszer az úgy nevezett „instancing”. Ez a megoldás nem csak a növénymegjelenítés során elterjedt, bárhol használható ahol sok egyforma vagy közel egyforma objektumot kell megjeleníteni. Nem szükséges minden elemet külön rajzolás hívással megjeleníteni. Amennyiben a rajzolni kívánt elemek kevés paraméterben különböznek, egyetlen rajzolás hívással kirajzolható az összes elem. Csak definiálni kell különböző paramétereket és azok értékeit. Az instancing előnye a renderelés során jön elő, a példányok hatékonyan jeleníthetőek meg, mivel a többszöri rajzolás hívás overhead-ét megspórolja egyetlen rajzolás hívással. Az instancing egy fajtája az úgy nevezett „approximate instancing” [1]. Ennek lényege, hogy egy procedurálisan generált, nagy változatosságú és részletességű modellben, amely nem tartalmaz pontosan ismétlődő elemeket, a növényeket vagy azok részeit hasonló, de véges elemkészletből vett, modellek példányaira cserélik. Ekkor a megjelenítés gyors lehet, a látványban pedig a különbség észrevétlenis maradhat. Ez a megoldás nem csak a növénymegjelenítés terén elterjedt. A mi megközelítésünk tekinthető az approximate instancing fordítottjának, mert véges elemkészletből építkezünk, majd az objektumokat egyedivé tesszük, de úgy, hogy továbbra is megjeleníthetők legyenek instancinggel. Például eltérő csontváz beállításokat alkalmazunk a fák elemeire. Az instancing során nem csak komplett növények másolatait jeleníthetjük meg. Kisebb struktúrák megjelenítésére is alkalmas. Mi a fa törzsét alkotó fa elemek megjelenítésére alkalmazzuk. Az elemek modelljei egy véges számú készletből választhatóak, viszont különböző elhelyezkedéssel, textúrázással és animációs fázissal kerülnek megjelenítésre.

A projektünk fő feladata ezen elemek pozíciójának és animációjának az összehangolása. Valamint az elemek egymáshoz illeszkedő textúrázása változatos módon. Mindezt valós időben, hatékony algoritmusok alkalmazásával.

A feladatot hat részre osztottuk. Az első feladat a fa törzsét és ágait alkotó elemek modelljeinek elkészítése, valamint az elemek háromdimenziós megjelenítése. Az elkészült modellek az 1. ábrán láthatóak.

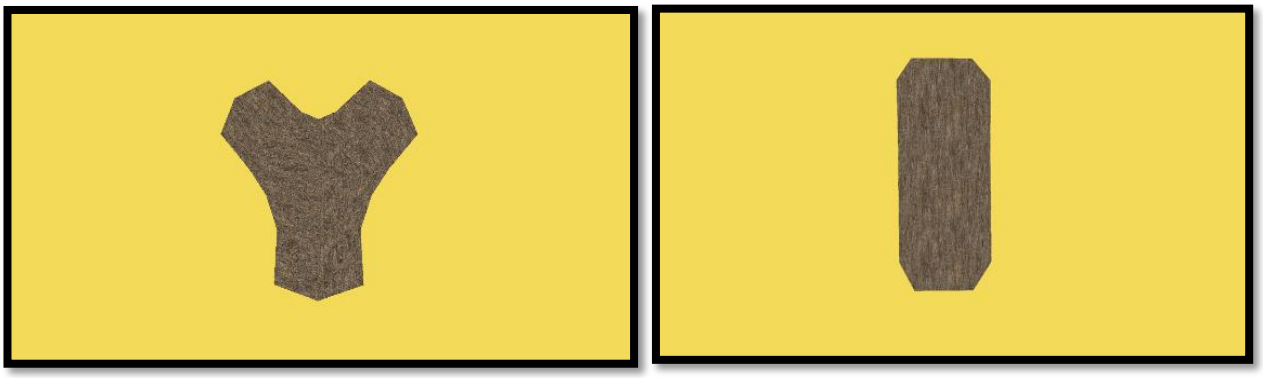
A második feladat az elkészült modellek felhasználásával az elemekből komplett fák előállítása. Az ehhez szükséges transzformációk algoritmikus előállítása. A probléma megoldását az első fejezetben fejtjük ki részletesen. Az eredményül előállított fa a 2. ábrán látható.

Harmadik feladatunk az elemek csontváz alapú animálása. Valamint az animált elemek illeszkedésének megvalósítása. Az animálás a második fejezetben kerül kifejtésre. Az animációs során előálló, nem modellezési pozícióban lévő elemek felhasználásával készült, fa képe a 3. ábrán látható.

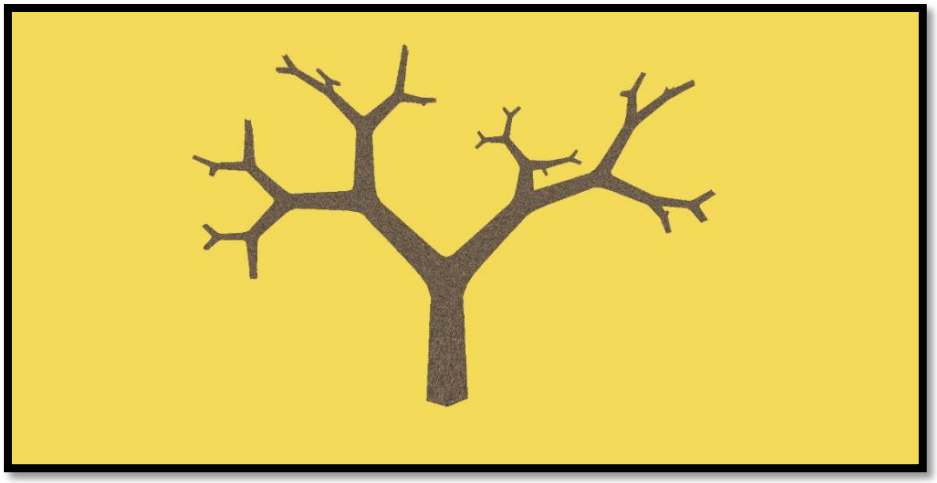
Negyedik feladatunk az elemek különböző részletességi szintű megjelenítése hardware-es tesszellálás segítségével. A tesszellálás a harmadik fejezetben kerül kifejtésre.

Ötödik részfeladatunk az elemek procedurális textúrázása. Elvárás, hogy a textúrázott elemek mintázata ne csak egy elemen belül legyen látszólag folytonos. Az elemek illeszkedésénél se keletkezzenek szakadások a mintázatban. A textúrázás a negyedik fejezetben kerül kifejtésre.

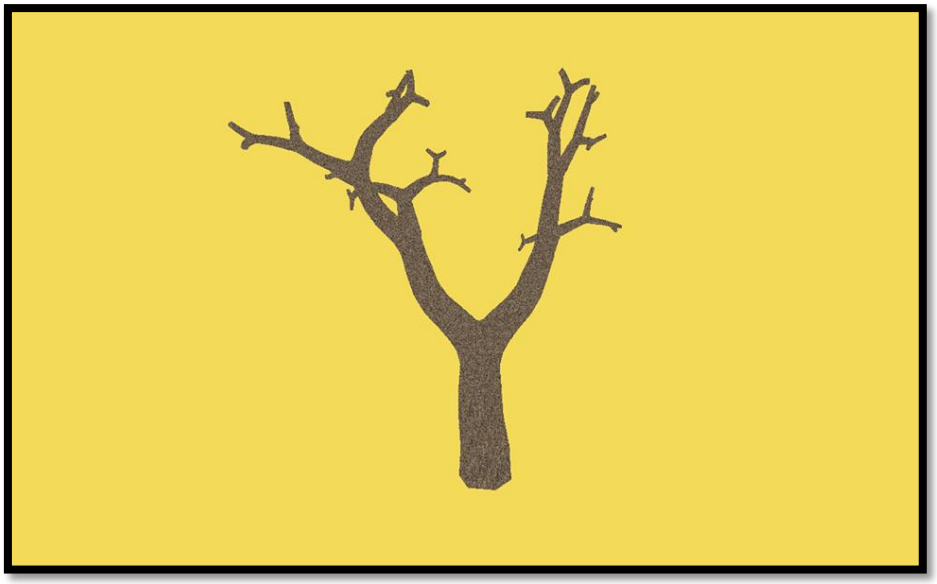
Hatodik részfeladatunk lombozat illesztése az elkészült fákhoz. A lombozatot billboard-ok használatával valósítjuk meg. A lombozat megvalósításának részletei az ötödik fejezetben található. Lombbal rendelkező fa képe tekinthető meg a 4. ábrán.



1. ábra. Általunk elkészített elemkészlet.



2. ábra. Elemekből felépített statikus fa.



3. ábra. Animált fa. Alkotó elemek nem modellezési pozícióban állnak.

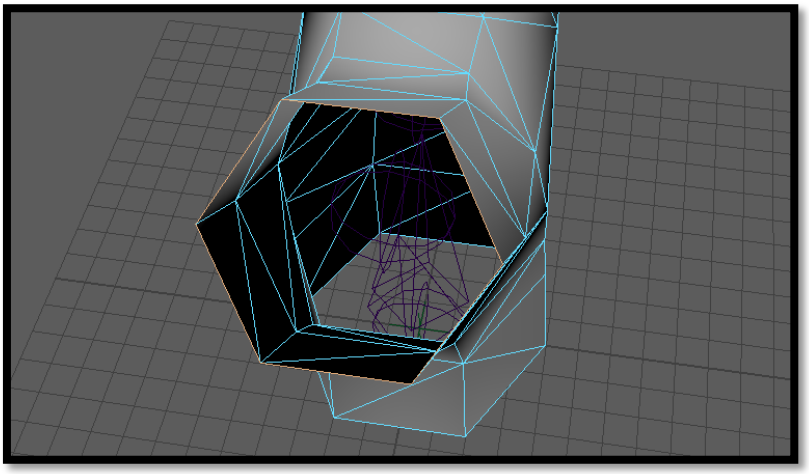
4. ábra. Lombozat.

1. fejezet

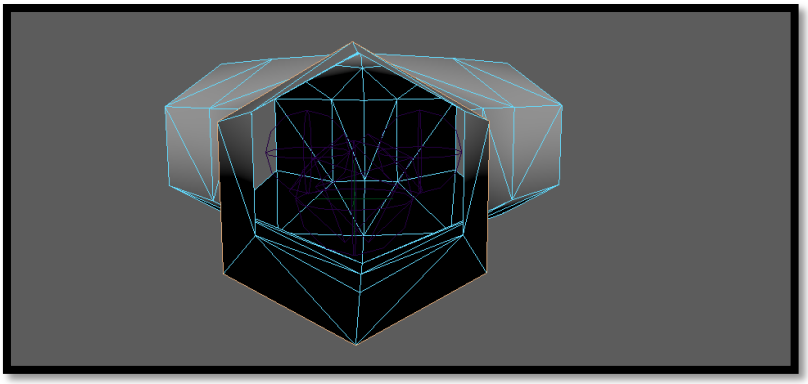
Fák felépítése

Modellek illeszthetősége

Az algoritmusok egységesítése érdekében a felhasználható fa részlet modelleknek különböző elvárásokat támasztottunk. A modellek összeillesztési helyének kijelölésére a modellek háromszöghálójában lap csoportokat definiáltunk. Ezen lap csoportokra illesztő felületekként hivatkozunk. Egy modellnek rendelkeznie kell egy gyökér illesztő felülettel és rendelkezhet tetszőleges számú levél illesztő felülettel. A gyökér illesztő felületen csatlakozik az adott modell a fa törzs felöli részfájához. A levél illesztő felületen az adott modellhez további modellek illeszthetőek. Az illesztő felületektől elvárt, hogy egy modell levél, valamint egy másik modell gyökér illesztő felületét megfelelő affin transzformációkkal a térben egymást fedő felültetekké lehessen transzformálni. Ebből következően, ha nem csak az illesztési felületeket, hanem a modelleket az illeszkedési felületükkel együtt transzformáljuk, akkor a két modell, az illeszkedési felületek illeszkedése miatt, egy darab összetartozó modell hatását kelti. Az 5. ábrán látható az Y modell levél illesztő felülete. A 6. ábrán látható az Y modell gyökér illesztő felülete.



5. ábra. Y modell levél illesztő felület.



6. ábra. Y modell gyökér illesztő felület.

A transzformációk egyszerűsítése érdekében illesztési felületnek a szabályos hatszöget határoztuk meg. Valamint az összes modell gyökér illesztési felületének azonos az orientációja és mérete is. A szabályos hatszög felület garantálja, hogy az illesztési transzformáció skálázási résztranszformációja a három tengely irányában azonos mértékű skálázásból álljon. Így a modellek illesztés után azonos arányokkal rendelkeznek, mint modellezési koordinátarendszerben. A megegyező gyökér illesztő felület garantálja, hogy egy adott levél illesztő felülethez ugyan azzal a transzformációval illeszthetjük bármely modell gyökér illesztő felületét. A szabályos hatszögek további előnye, hogy hat különböző, a hatszög középpontját metsző és a síkjára merőleges tengelyű, elforgatással is illeszthetőek.

Tetszőleges elemek illesztési transzformációjának meghatározásához szükség van a modellek levél illesztő felületeinek a közös gyökér illesztő felülettel összeillesztő transzformációjára. Egy levél illesztő felülethez egy transzformáció tartozik. Ha egy modellen több levél illesztő felület is van, akkor mindegyikhez külön transzformáció tartozik. Ezen transzformációkat metaadatként tároljuk a modellekhez. A modellek elkészítésekor ezen transzformációkat is elő kell állítani. A modell építő transzformációiként hivatkozunk ezen transzformációkra.

Fa elemek elhelyezkedésének meghatározása

Egy fa felépítésének első lépése egy gyökér elem választással kezdődik. Minden más elemre kiszámítunk egy transzformációt, amely a saját modellezési koordinátarendszeréből a gyökér elem modellezési koordinátarendszerében épülő fában elfoglalt helyére transzformálja a modellt. Ezen transzformációkra topológiai transzformációként hivatkozunk. Az elemek topológiai transzformációi rekurzívan számíthatóak. A közvetlen a gyökérelemhez illeszkedő elemek topológiai transzformációi megegyeznek a gyökér elem modelljéhez definiált építő

transzformációkkal. Tetszőleges szülő elemhez illeszkedő elem topológiai transzformációja két részből áll. Az első transzformáció a szülő topológiai transzformációja. Ezt alkalmazva az új elem a szülő koordinátarendszerébe kerül. Csak ezt az első résztranszformációt alkalmazva az új elemre a szülő és az új elem gyökér illesztő felületei illeszkednek egymásra. Az illeszkedő felületek normálvektorai megegyező irányba álnak. A második transzformációja az új elem topológiai transzformációjának az új elem gyökér illesztő felületét a szülő elem gyökér illesztő felületéről a szülő elem levél illesztő felületére transzformáló transzformáció. Ez megegyezik a szülő elem modelljének építő transzformációjával. Ebből következően egy tetszőleges elem topológiai transzformációja a gyökér elemtől az adott elem szülőjéig vezető elemek modelljeinek megfelelő építő transzformációinak egymás utáni végrehajtása.

Minden elemre alkalmazzuk a hozzá tartozó topológiai transzformációt, így az elemek összerendeződnek, egy összefüggő fát alkotnak. Ha az egész fát ezután transzformálni szeretnénk a világ koordinátarendszerben elfoglalt helyére a gyökérelem modellezési koordinátarendszeréből, minden elemre alkalmaznunk kell a topológiai transzformációt követően a modellezési transzformációt is. Több különböző fa esetén minden fát a saját gyökér elemétől fel kell építeni, majd elhelyezhető a világ koordinátarendszerben.

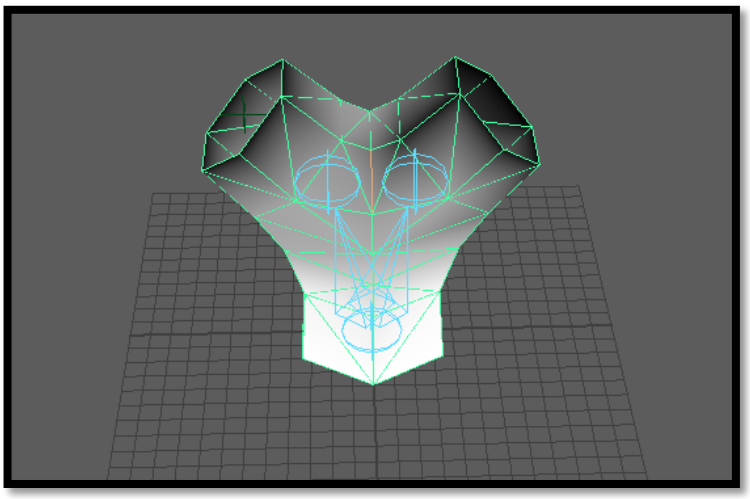
2. fejezet

Fák animációja

Általános csontváz animáció

A fának egy adott elemének animálásához általános csontváz alapú karakter animációt alkalmazunk. Ebben a szakaszban összefoglaljuk a csontvázanimációnak számunkra lényeges alapjait. A modellekben definiálunk ízületeket. Az Y modell ízület hierarchiája a 7. ábrán látható. Az ízületek egy fa struktúrába vannak rendezve. Minden ízület rendelkezik egy forgatási transzformációval, amely kezdetben egységmátrix. Valamint rendelkezik egy ízület transzformációval, amely az ízület koordinátarendszerét áttranszformálja modellezési koordinátarendszerbe. Az ízület forgatási transzformációja az ízület koordinátarendszerében értelmezett. Az animálás közben az ízületek forgási transzformációjának változása határozza meg a vertexek pozíciójának változását. A vertexek külön súlyozva függenek az egyes ízületektől. A vertexek transzformálásakor nem csak az adott elemre vonatkozó, az elemen belül egységes, transzformációkat hajtjuk végre, hanem a vertexhez rendelt ízületek transzformációját is súlyozva. Vertex transzformálásakor az ízület forgatási transzformációja nem egyezik meg a vertexen végzett transzformációval. Először az ízület forgatási transzformációját az ízület koordináta rendszeréből modellezései koordináta rendszerbe kell transzformálni. Ehhez a forgatási transzformáció előtt alkalmazni kell az ízülethez tartozó ízület transzformáció inverzét, mely a modellezési koordináta rendszerből ízület koordináta rendszerbe transzformál. Ezt követően elvégezhető a forgatási transzformációja. Végül újra alkalmazva az ízületi transzformációt invertálás nélkül visszakerülünk ízület koordináta rendszerből modellezési koordinátarendszerbe. Egy adott vertex nem csak a hozzá rendelt ízületektől függ, hanem azoktól az ízületektől is, amik a hozzá rendelt ízületek szülő ízületei, akár tranzitívan is. Ebből következően egy ízület egy vertexét nem csak a saját modellezési koordinátarendszerbe transzformált forgatási transzformációjával módosítja, hanem ezen transzformáció előtt a gyökér elemtől kezdve a hozzárendelt ízületig az összes ízület modellezési koordináta rendszerbe transzformált forgatási transzformációját végre kell hajtani. Az így kapott transzformációra hivatkozunk az ízület animációs transzformációjaként. Az animált elem megjelenítéséhez minden vertexet az elem topológiai transzformációja előtt a

vertexhez kötött ízületek animációs transzformációját is végre kell hajani az adott vertexhez tartozó ízület súlyok szerint.



7. ábra. Y modell ízületek.

Animált elemek illesztése

Az elemek animálása következtében a statikusan elhelyezett elemek nem illeszkednek, mert az elemek nem követik a szülő elemek animálását.

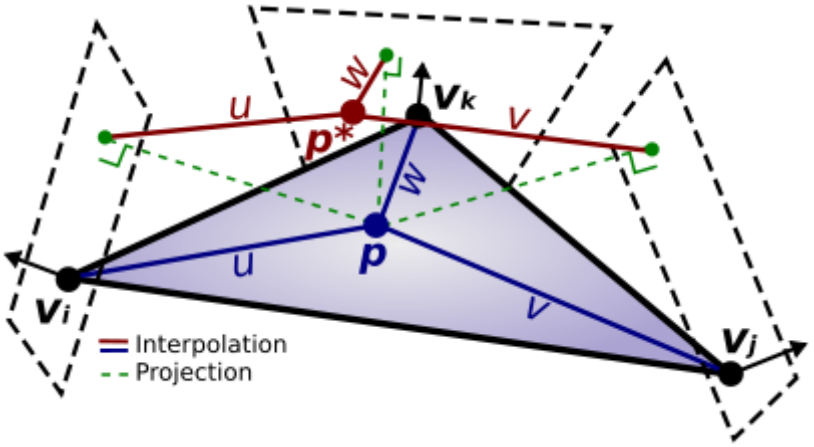
A probléma megoldása érdekben további megkötéseket kellett tennünk a modellek felé. Adott illesztési felület vertexei csak egy ízülethez köthetőek pontosan 1-es súlyozással. A gyökér illesztési felület vertexei pedig nem transzformálódhatnak animálás hatására. Az új megkötések biztosítják, hogy az eddig összeilleszthető felületek továbbra is összeilleszthetőek maradjanak, mert így maguk a felületek nem változnak, csak egy plusz transzformáción mennek keresztül animálás következtében.

Animálás nélkül az illesztő felületek illeszkednek. Animálás használatával a levél illesztő felületen egy plusz transzformáció lett végrehajtva. Ez a transzformáció a levél illesztő felület vertexeihez rendelt ízületben kiszámolt animációs transzformáció. Ugyanezt a transzformációt kell alkalmazni a levél illesztő felülethez illesztett teljes elemre. Így animálás használatakor a topológiai transzformáció kiszámításakor minden modell levélillesztő felületéhez tartozó építő transzformáció után el kell végezni a felület ízületéhez tartozó animációs transzformációt is. Ebből következően újra kell számolni az elemek topológiai transzformációját az animációs transzformációkkal kiegészítve. Vagyis egy topológiai transzformáció már nem a gyökér elemtől a szülő elemig vezető elemek modelljeinek építő transzformációinak sorozata. Hanem az építő transzformáció és az építő transzformációhoz tartozó levél illesztő felülethez tartozó ízület animációs transzformáció párosok sorozata.

3. fejezet

Tesszellálás

Hardware-es tesszellációt használtunk. A távolság függvényében exponenciálisan növekedő felbontási szintet javasolunk egy a futtató környezethez megfelelően megválasztott maximummal. Mivel a mi modellünk négyszögháló modell és ismertek a normálvektorok, ezért a feladatunk PN patch tesszellálás. A patch a négyszögháló egy négyszöge. Tesszelláláskor a patch négy csúcsa között feszülő felületre veszünk fel új, köztes csúcspontokat. Egy patch feldolgozásakor a patch csúcsainak pozíciója és normálvektora adott, erre utal a tesszellálás PN minősítője. A tesszellálás megvalósításához két algoritmus merült fel: *A Phong* [9] és a *Local, Polynomial G 1 PN Quads* [10] nevezetű algoritmusok. Mivel a Phong tesszellálás egyszerűbben számítható, ezért alkalmasabb több modell futásidejű tesszellálására. Ebből következően Phong tesszellálással valósítottuk meg a fa elemek tesszellálását. A Phong tesszellálást követően az új létrejött vertexek pozíciói a modell egy adott patch-ében lévő vertexek és azok normálvektoraitól függ. A 8. ábrán látható egy három vertexből álló patch Phong tesszellálásának illusztrációja. Mi ezzel szemben nem 3 hanem 4 vertexből álló pacthekkel dolgozunk. Amennyiben a létrejött vertex nem a patch belsejében, hanem egy él vonalában jött létre az új vertex pozíciója csak az él két vertexének pozíciójától és normálvektorától függ.

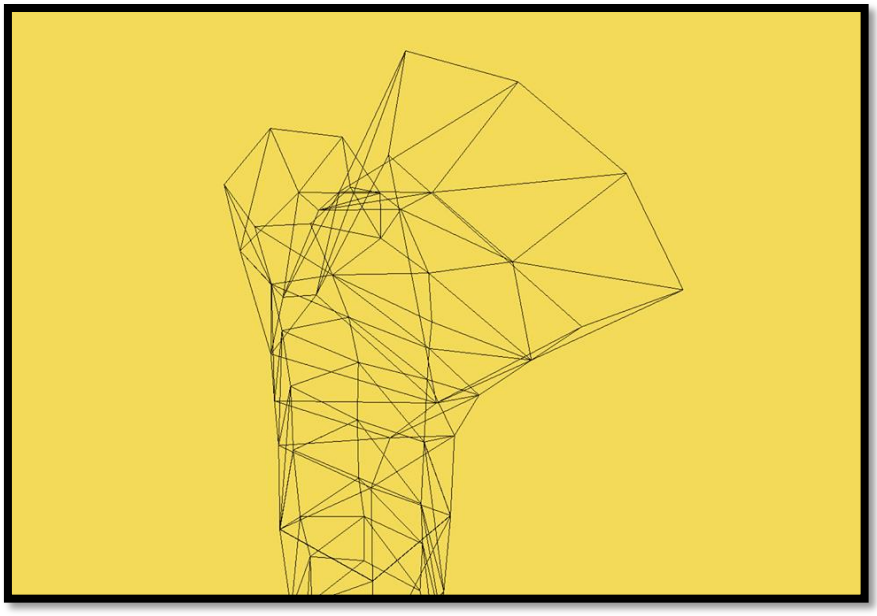


8. ábra. Phong tesszellálás egy patch-re. Forrás: [9]

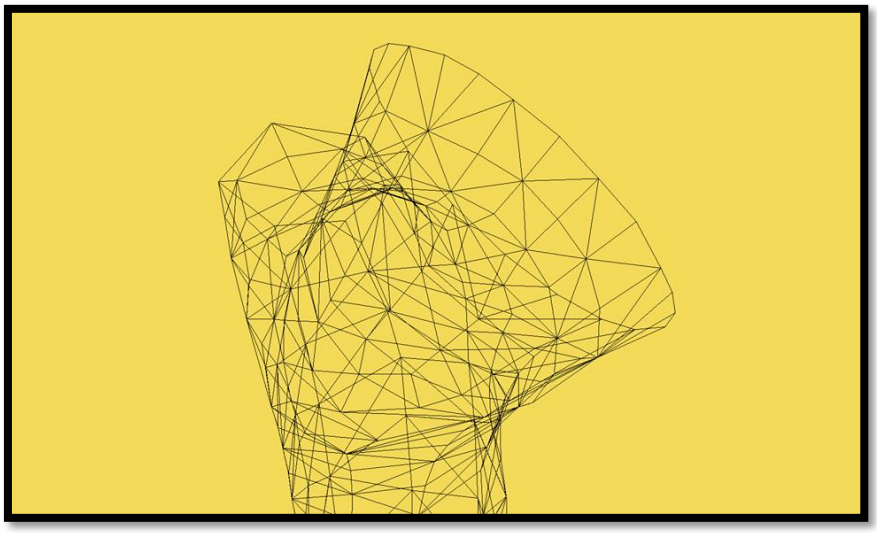
Az illesztési felületek vertexeinek illeszkedést tesszellálás nélkül garantáltuk. Tesszellálás használatakor az újonnan létrejött vertexek illeszkedését is garantálni kell. Egyrészt garantálni kell az új vertexek számának egyezését. Másrészt azok pozíciójának illeszkedését is. A tesszellálási szint egyezése egyszerűen megoldható, ha az élek tesszellálási szintje csak az él két vertexétől függ, mert azok illeszkedése már garantált. Természetesen az állítás csak abban az esetben igaz, ha a tesszellálási szint kiszámításakor az illeszkedő vertexek valóban egyező tulajdonságuk alapján számoljuk. Jelen esetben az illeszkedő vertexek azonos tulajdonsága az azonos pozíció, ebből kifolyólag a kamerától mért távolság is azonos. A tesszellációs szintet két vertex alapján határozzuk meg, ezért fontos hogy mindkét vertex szimmetrikusan szereplejen a számításban, ezáltal ne függjön a tesszellálási szint a vetrexek sorrendjétől. Például két pozíció számtani közepe szimmetrikus, így nem függ a vertexek sorrendjétől. Ebből kifolyólag az élek tesszellálási szintjét a két vertexpozíció átlagának és a kamera pozíciójának távolsága alapján határoztuk meg.

Az új vertexek pozíciója függ az illeszkedési felület vertexeinek pozíciójától és normálvektorától is. Ennek megfelelően az illesztési felületek vertexeinek illesztéskor nem csak a pozícióknak kell egyeznie, hanem a normálvektoroknak is. Az elvárás újabb kitételt szab a használandó modellek felé. Mi a normálvektorok illeszkedését egy normálvektor számítási szabállyal oldottuk meg. A szabályos hatszög illesztési felületek vertexeinek normálvektorai a hatszög középpontjából a vertexbe mutató vektorral párhuzamosak lettek, ezáltal a normálvektorok egyezését az egyező pozíciókból származtattuk. Így a hatszögek bármely csúcsa bármellyel lesz illesztve a normálvektorok is azonosak lesznek, mert az illeszkedő vertexbe az illeszkedő középpontból mutató vektor is azonos irányú lesz. Ez a megoldás az illesztési felületek elforgatását továbbra se zárja ki.

A megoldás hátránya, hogy feltételezi, az illesztő felület vertexeinek ilyen módon számított normálvektorai nem térnek el nagymértékben az optimális normál vektoroktól. Optimális normálvektor alatt egy modellező személy vagy algoritmus által beállított, a modell felületéhez illő normálvektort értünk. A mi modelljeinkben az illesztő felületek merőleges vágásként jelennek meg az ág elemeken. Az ág elemek illesztő felülethez közeli részletei jól közelíthetőek hengerrel. Ilyen esetben az illesztő felület egy vertexének normálvektora nagymértékben megegyezik az ág irányára merőleges és a vertexen átmenő egyenes irányvektorával, ami a közelítő henger normálvektora. Ezáltal az optimális normálvektor hatását kelti. A tesszellálás eredménye az 9. és 10. ábrákon tekinthető meg.



9. ábra. Y modell tesszellálás előtt.



10. ábra. Y modell tesszellálás után.

4. fejezet

Textúrázás

A következő fejezetben általánosságban összefoglalnánk a textúrázás fogalmát és az alkalmazott módszereket, majd ezután térünk ki a probléma során alkalmazott textúrázási megoldásra.

Alapfogalmak

A számítógépes grafikában jól ismert fogalom a textúrázás, melyet azért alkalmazzuk, hogy általában 2D-s képeket ráfeszítsük a 3D-s modellünkre és ezáltal növeljük az okozott vizuális élményt és a részletességét a modellnek anélkül, hogy a felületek geometriáját bonyolítanánk. Erre egy jó példa lehetne az, hogy szeretnénk egy házat modellezni. Ekkor lehetőségünk lenne minden részletet lemodellezni az ablakpárkány kiugró részével együtt, azonban hatékonyabb módszer az erőforrásokra nézve, ha olyan textúrát illesztünk az oldalára, amely a ház oldalán lévő ablak is szerepel. Persze a végeredmény hatása sokban függ a művészi munka minőségétől is, amelyet különböző képmanipuláló alkalmazások segítségével lehet fokozni.

Tehát lényegében a 2D-s képet ráillesztjük egy 3D-s modellre, de a kérdés az, hogy ez hogyan lehetséges? A textúra egy úgynevezett textúra térben van, amelyben a textúra minden pontja egy színértékkel rendelkezik, és ezeket a pontokat texelnek hívjuk. A texeleket a textúrakoordináták segítségével tudjuk elérni, és ezen koordinátákat u és v-nek nevezzük. Emellett ezek az u és v értékek normalizált tartományban vannak, vagyis [0,1] az értéktartományuk, ezáltal egységesen címezhetők a textúraállományok. Azt a folyamatot amikor ilyen koordináták által meghatározott textúra-pontokat a háromszögháló-modell csúcsaihoz rendeljük textúra-paraméterezésnek (uv-mapping) [2] nevezzük. Ez természetesen leginkább programok segítségével történik, de egyszerűbb testekre manuálisan is elvégezhetjük.

Megkell említeni egy problémát, amely előfordul a testre való textúrakép feszítésekor, amit eltérő méretek okoznak, vagyis az, hogy a textúrakép és a modellt alkotó háromszögek képernyőtérbeli mérete eltérő lehet, ami azt okozza, hogy vagy a kép összenyomódik, vagy

szétfeszül vagyis ha nagyon más méretűek, akkor a pixelszín nem következik a texelszínekből. A textúraszűrés során erre a problémára kapunk megoldást. Az egyik textúraszűrési lehetőség például

- Nearest-Point Sampling, amely során kiszámolja a texel címét és a legközelebbi texelt rendeli hozzá. Ezt leginkább hasonló méretek esetén célszerű alkalmazni.
- A linear filtering eredménye jobb, mint az előzőleg bemutatott módszer, de számításigényesebb, mivel ez a négy legközelebbi texel súlyozott átlagát használja fel a szín kiszámításához.
- Ezeken felül az Anisotropic szűrő segítségével megoldható az a probléma is, amikor a végeredményben kapott modell elfordul a kamerához képest és a textúrakép torzul.
- A mipmapekkel való textúraszűrés esetén van egy textúraképünk, és abból felezéssel kapunk több képet, ezáltal amikor a modell közelebb van a nézőhöz akkor a nagyobb képet rakja rá, míg a távolibbra elég az alacsonyabb felbontású kép illesztése.

A multitextúrázás során arra van lehetőségünk, hogy a modellen több textúrát is alkalmazhatunk, vagyis azt jelenti, hogy a több textúrából származó színeket összemixeljük és a kapott színeket alkalmazzuk.

Alapvetően a textúrákat számítógépes programok segítségével alkotjuk meg, rajzoljuk meg. Ennek hátránya, hogy ha bonyolultabb textúraképet szeretnénk létrehozni kézileg az sokáig tart és nehézkes. A procedurális textúrázás módszere megoldást nyújt erre a problémára, hiszen matematikai módszerek segítségével képesek lehetünk textúraképeket létrehozni lényegesen gyorsabban és hatékonyabban. Jelenleg rengeteg program elérhető neten melyekkel generálhatunk procedurális textúrát.

A Direct3D-ben a textúra erőforrás [3], amelyet texelek tárolására tervezték. Az alkalmazható textúratípusok:

- 1D textúrák: Az 1D textúrák tulajdonképpen texelek tömbje melyben a texelek az egyetlen textúrakoordináta (u) segítségével címezhetők.
- 1D textúratömbök: Az 1D textúra tömbök esetén az előbb felvázolt 1D textúrák tömbje.
- 2D textúrák: minden texel eléréséhez két koordináta megadására van szükség, vagyis u és v-re.
- 2D textúratömbök: A 2D textúrák az előbb felvázolt textúrák tömbje, tehát textúra kockaként is lehetne vizualizálni.
- 3D textúrák: 3D esetén egy w koordináta is bejön a képbe.

A probléma

Két alapelem segítségével (egy hat oldalú henger és egy Y alakú test), algoritmikusan épül fel a fa különböző szabályok betartása mellett. A fa részletességének és művészi hatásának növelése érdekében a kérget textúrázás segítségével lehet elérni. Az első lehetséges megoldás az, hogy minden építőelemre ugyanaz a textúrakép illesztődik, ez azonban kerülendő, ha nem szeretnénk, hogy minden elem ugyanúgy nézzen ki.

Célravezetőbb lehet egy olyan megoldás, amelynél kihasználható az a tény, hogy a modellek négyszögekből épülnek fel, és ezekre a négyszögekre különálló képeket illeszthetünk. Ezt a metódust textúracsempézésnek nevezzük.

A csempézés eredményeként kapott feltextúrázott modell minden poligonján ugyanaz a kép fog ismétlődni, ami helyett előnyösebb megoldás lehetne az, hogy egy előre meghatározott textúrakészletből véletlenszerűen választott kép kerül egy megadott poligonra, így a csempék valamely mértékben eltérők lesznek, végeredményben pedig a randomizálás hatásaként a modellek is.

A csempézés során már felmerült az egy textúrakép helyett a textúrakészlet alkalmazása. Ekkor a csempék illeszkedése érdekében illesztési szabályok kellenek. Ennek egy speciális esete, amikor adott, hogy a modell élein milyen illeszkedő minta van. Ezek után a négyszögek kerete fix, a belseje meg változhat. A lényege az, hogy meghatározzuk, hogy egy csempe négy éléhez mely másik csempék illeszkednek, és ezen szabályok segítségével épül fel a textúra.

Egy kérdés itt azonban tisztázatlan, milyen képek kerüljenek az Y modell hajlásaiba? Illetve, ha szeretnénk megjeleníteni egyéb részleteket, mint például odú azt hogyan és hova kerüljön? A megoldás plusz információtömbbel bővült, amelynek célja csupán az, hogy kategorizálja a poligonokat a rájuk kerülhető textúrák szerint.

Ez fakéregtextúrázás esetén egyszerű, hiszen megmondhatjuk, hogy az adott poligonok olyan típusba tartoznak, melyekhez illeszthetünk sima kéregtextúrát vagy odútextúrát, míg egy másik részletre olyan textúrák kerülhetnek melyen a kéreg illeszkedik a hajlatba. Ez egy egyszerű példa, természetesen az odú helyett sokféle más textúra is illeszthető. Az, hogy azonban ahhoz a típushoz tartozó textúrák közül melyik kerül alkalmazásra, véletlenszerűen dől el.

Algoritmus

Csempézés

Első lépésként a modellen való változtatásokat kell elvégezni, ahhoz, hogy mindegyik poligonra kerülő kép orientációja megfelelő legyen. Ez abban az esetben nem fontos, ha az amúgy is egymáshoz illeszkedő textúraképekre nem helyezünk olyan részletet melynek orientációja fontos lenne, például fagomba. Ellenkező esetben célszerű már a modellezéskor beállítani mindegyik poligonra a textúrakoordináták elforgatását.

A modell textúrájának betöltésére a Direct3D részletes segítséget nyújt. A képpontárnyaló segítségével a háromszögháló négyszögeire könnyen kirajzolhatók. A csempézés hátránya ekkor is megjelenik, hiszen, ha textúrák nem illeszkednek egymáshoz, akkor ez sajnos nagyon feltűnő tud lenni, amelyre későbbi részben térünk ki.

Textúrakészlet

Előző rész során előállt eredményen látható, hogy minden csempére ugyanaz a mintázat fog kerülni, ezt elkerülendően a következő módszert vesszük igénybe. Az alapötlet az, hogy nem egy textúrát alkalmazunk, hanem egy textúratömböt. A textúratömbök Direct3D-be dds formátum segítségével tölthetők be. Egy texassemble [4] nevű program segítségével hatékonyan és gyorsan előállítható ez a formátum, elég parancssorból néhány paraméter segítségével futtatni az alkalmazást.

Ahhoz, hogy az árnyaló képes legyen eldönteni, hogy a tömb melyik eleme kerüljön a poligonra, szükséges információnak meg kell határozni, hogy melyik poligon milyen típusba tartozik és ezt feltölteni az árnyalónak. A processzor oldalon egy tömbbe elmenthető ez az információ, amely azért egyszerű megoldás, mert csak az Y modell esetén a hajlatokban tér el a típus, az egyenesek esetén mind ugyanabba a kategóriába kerülnek. A tömb annyi elemet fog tartalmazni, ahány poligonból áll a modell, illetve az index határozza meg a poligon sorszámát, a beírt szám pedig a típust.

Ezen kívül alkalmazható még egy másik tömb is, melyben az az információ kerül eltárolásra, hogy az előzetesen betöltött textúrakészletből mely textúrák tartoznak melyik típushoz. Ha tudjuk, hogy egy típusba körülbelül három textúra fog tartozni, akkor a címzés egyszerűen megvalósítható. Amennyiben eltérő ez a szám, egy másik tömbben is eltárolható az információ, hogy az ugrások könnyen számolhatók legyenek.

Végső eredményben a képpontárnyalóban az adott felületre a típusának megfelelő textúrakészletből véletlenszerűen kerül egy kép, ezzel biztosítva az egyediséget a modellek és végeredményben a fák között.

Illeszkedés

Csempézésnél, két szomszédos csempe közötti eltérés az érintkező élen jól látható, amennyiben az alkalmazott minták nem illeszkedők az adott élen. Az illeszkedés itt annyit jelent, hogy a jól látható csempehatárok helyett egy folytonosnak tűnő felülettextúrát szeretnénk látni. A felületek csempéire kerülő képeknek olyan textúraelemnek kell lenniük, melyek puzzle-szerűen összeépítve biztosítják az egységes hatást. Ehhez bizonyos szabályok leírására van szükség, amely meghatározza, hogy egy adott minta melyik oldalához mely minták illeszkednek. A textúrák elkészítéséhez használjuk az Image Quilting algoritmust [5], melynek eredményeként előállnak olyan képek, amelynek egyes oldalai más meghatározott textúrákkal összeillesztve folytonos hatást keltenek.

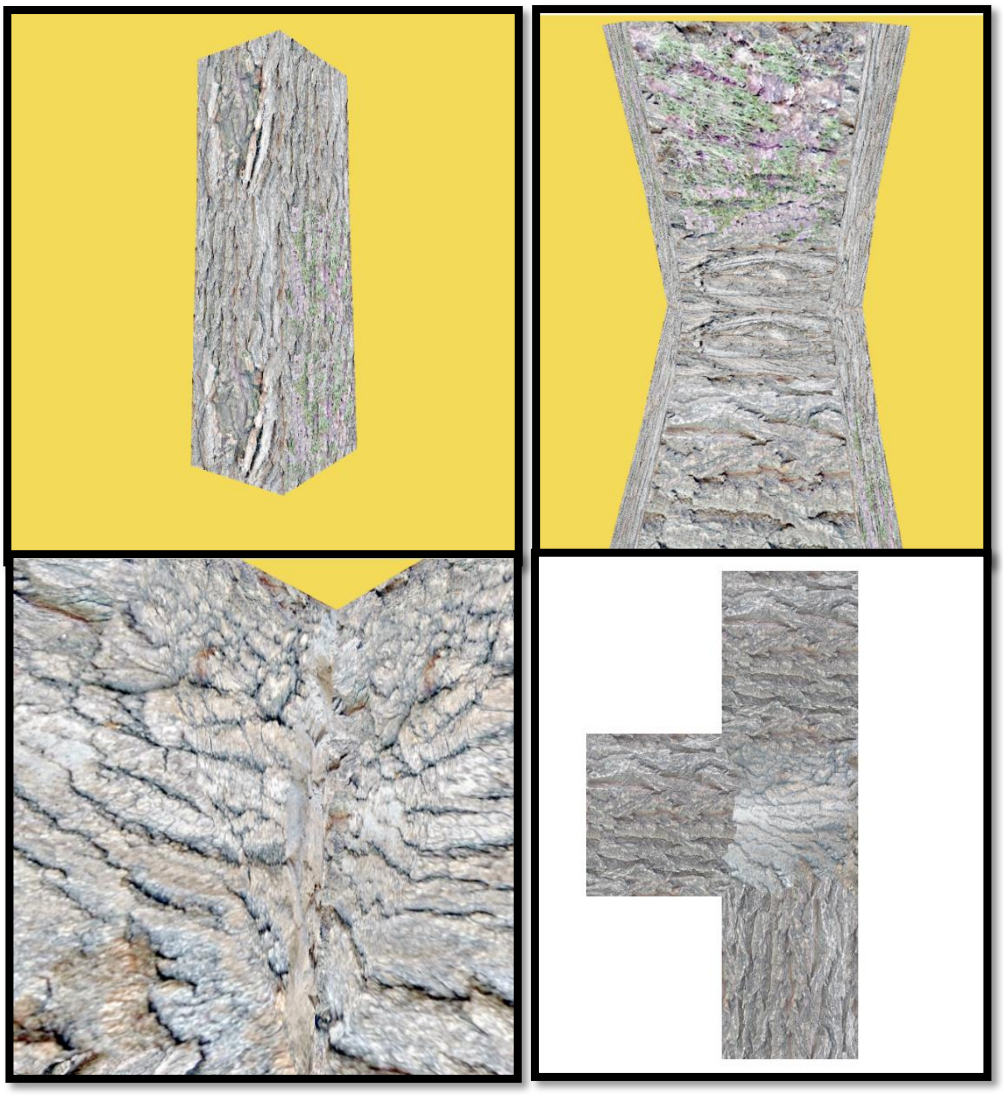
Wang csempék [6] szintén arra a problémára nyújtanak megoldást, amikor egy textúraképpel felszeretnénk csempézni a modellt, akkor a csempék közötti vágások feltűnőek. A módszer olyan ábrázolási módot használ melyben az illeszkedő éleket színekkel jelöli, és két él összeilleszthető, ha azonos a színük. Tehát ami fontos az a textúra kerete, így a belseje szabadon választható, ami egyfajta variálási lehetőséget ad. Az összes csempézési lehetőségre való variáció tárolása nem túl hatékony, ehelyett van jobb módszer mely esetén nem szükséges a teljes variáció készlet eltárolása a memóriában. Elég a nyugati és keleti színek összes kombinációs lehetőségére legalább két csempe meghatározása, illetve elhelyezésüknél néhány szabály betartása. Például miután választottunk egy csempét és elhelyeztük, mellé úgy teszünk másikat, hogy vagy ugyanazt a csempét rakjuk, vagy egy olyan random választottat, melynek nyugati éle illeszkedik az előző keleti éléhez. Egy csempe alá olyan elem kerülhet melynek északi éle illeszkedik a déli éléhez, és ezen csempe mellé a nyugati, keleti szabálynak megfelelően kerülhet elem.

Image Quilting [5] esetén két összeillesztendő kép esetén a képeket egymás mellé helyezzük, úgy, hogy az illesztendő élen átfedjenek, majd meghatározunk egy minimális hibahatár vágást Dijkstra algoritmusa vagy dinamikus programozás használatával. Ott szeretnénk, hogy legyen a vágás két átlapolódó blokkján a pixeleknek, ahol a két textúra legjobban megegyezik, vagyis az átlapolódási hiba alacsony. Az algoritmus, amit használunk az a minimális költségű út meghatározására jó. Például van két vertikálisan átfedő blokkunk, A és B, A_{ov} és B_{ov} átfedő régiókkal, akkor a hibafelületet definiálhatjuk a két régió különbségének

négyzeteként. Ahhoz pedig, hogy megtaláljuk a minimális vertikális vágást végig kell számolnunk a hibafelületet, és meghatároznunk a kummulatív minimális hibát minden útra. Mivel soronként haladunk, a végén az utolsó sorban lévő minimális hiba indikálni fogja a minimális vertikális vágás végét a felületen, így visszafelé haladva meghatározhatjuk a legjobb vágás útvonalát. A kummulatív minimális hiba kiszámítási képlete a következő:

$$E_{i,j} = e_{i,j} + \min(E_{i1,j1}, E_{i1,j}, E_{i1,j+1})$$

Ahol $e_{i,j}$ a hibafelület, $E_{i,j}$ pedig a kummulatív minimális hiba. Az (i,j) pedig a pixel koordinátája. A módszer hasonlóan alkalmazható horizontális átfedésre. Ha pedig vertikális és horizontális átfedés van, a minimális költségű út (overall minimum) választható vágáshoz. Végeredményként a létrejött csempékkel feltextúrázott I elem és Y hajlat a 11-14. ábrán látható.



11-14. ábrák. Illeszkedő textúra csempék.

5. fejezet

Lombozat

A lombozat megvalósításához plakátokat használunk fel, melyeket elhelyezzük a fán. A plakátok pozícióját a fa ágai határozzák meg, így törzsre nem kerül levélcsoport. Azt az információt, hogy egyes levélcsoportok milyen pozícióba fognak kerülni, a fa határozza meg. Ennek az elemei rendelkeznek arról az információról, hogy mely levelek tartoznak hozzá. A levélcsoportok a fa bejárásához hasonlóan rajzolódnak ki.

A lombozatot létre lehetne hozni csupán billboard-ok segítségével is. Ekkor megjelenítünk egy képet, amely mindig a kamera irányába fog fordulni amennyire csak lehet, miközben egy tengely körül forog. Ennek előnye az, hogy sokkal gyorsabbá válik a renderelés, mint például levélmodellek alkalmazása esetén. Azonban hátránya az, hogy a plakátok két dimenziós felületek és a mélységteszt ezekre hajtódik végre, a levelek három dimenziós geometriája helyett.

Hatékonyabbnak tűnt viszont a 2,5 dimenziós imposztorok [7] használata. Ebben a módszerben a plakátokat mélységinformációval egészítjük ki, így azok a háromdimenziós geometria mintavételezett reprezentációjának tekinthető, az ilyen plakátokra a valódi 3D geometriára megfelelő mélységtesztelés alkalmazható. A nézeti irány változásakor a plakáton lévő geometria is forogni látszana, ezért a kamera nézeti irányának megfelelően a plakátokat mindig újra számoljuk. Modellező program segítségével létrehoztunk egy levélfelhő modellt, melyet beimportáltunk a projektbe. Két menetes a renderelés, és az első renderelési szakaszban a modellt textúrába rendereljük ki. A jelenet kamerájának a pozícióját megváltoztatjuk csak erre a renderelésre, de az iránya nem változik. A megjelenítendő levélfelhő pozíciója az origóban van. Az így kapott textúra alfa csatornájába mentjük el a távolságinformációt, vagyis a szempozíció és a billboard közötti távolságot, mely segít annak az eldöntésében, hogy csak a látható elemeket rajzoljuk ki. Második fázisban ezt a képet alkalmazzuk, mint plakát és helyezzük el az ágakra.

6. fejezet

Implementáció

Alkalmazott technikák

A szoftveres implementációt DirectX11 és C++ segítségével készítettük el. A DirectX sok függvény segítségével teszi lehetővé grafikus alkalmazások fejlesztését, emellett elfedi a videókártya sajátosságait. A shaderek programozásához a HLSL nyelvet vettük igénybe.

A DirectX-hez az Effect Framework-ot, amelynek segítségével technique-kat definiálhatunk, amik összegyűjtik a hozzájuk tartozó passokat. A pass fogja össze a rendereléshez szükséges shadereket, vagyis a vertex és a pixel shader mellett az egyéb alkalmazott shadereket.

Fejlesztőkörnyezetnek a Visual C++ 2010 Expresst használtuk. A modellezéshez pedig AutoDesk Maya modellezőszoftvert használtuk fel, melyből a modelleket DAE formátumba exportáltuk ki. Ezen fájlok beimportálását a projektbe az Assimp segítségével végeztük el. Az Assimp egy nyílt forrású könyvtár elsősorban C és C++-hoz, amellyel könnyen tölthetünk be projektünkbe több 3D fájl formátumot. Ezen kívül a Boost-ot használtuk fel, amely szintén C++-hoz biztosít könyvtárakat, és így modern C++ nyelvi elemekkel bővítettük az implementációnkat.

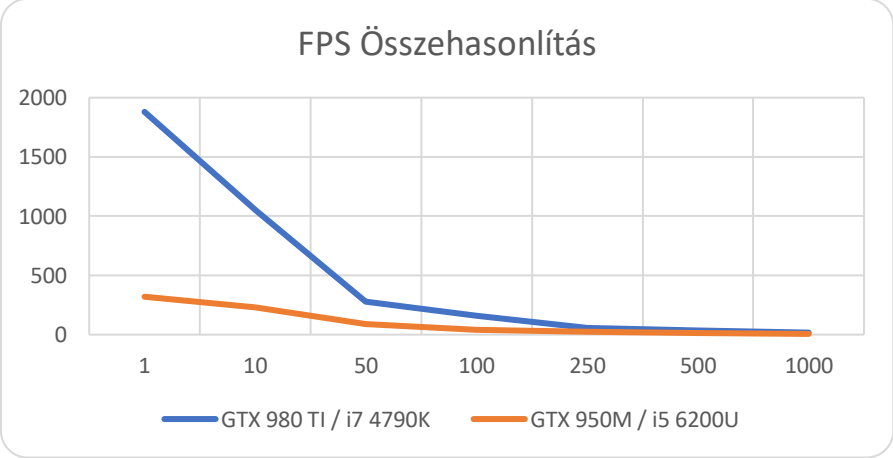
A textúratömb elkészítéséhez a már említett texassemble nevű programot használjuk, amely dds formátumba menti ki a kellő eredményt.

7. fejezet

Eredmények és konklúzió

Az elkészült implementációt két különböző teljesítményű gépen hajtottuk végre. A futtatás során legeneráltunk fákat és ezekhez néztük az FPS-t majd az eredményeket feljegyeztük:

Fák száma	GPU: GTX 980 Ti CPU: i7 4790K	GPU: GTX 950M CPU: i5 6200U
1	1880 FPS	320 FPS
10	1050 FPS	230 FPS
50	280 FPS	90 FPS
100	160 FPS	40 FPS
250	57 FPS	26 FPS
500	35 FPS	12 FPS
1000	17 FPS	6 FPS



A megoldásunk teljesíti a vele szemben támasztott követelményeinket, vagyis azt, hogy véges számú elemkészletből algoritmikusan épülnek fel a fák, úgy, hogy bizonyos mértékű változatosságot hordoznak magukban. A fára az előre meghatározott textúrakészletből véletlenszerűen kerülnek képek, míg lombozat az ágakra illeszkedően rajzolódik ki. Emellett viszonylag nagyszámú fák megjelenítésére van lehetőségünk.

A jövőben szeretnénk a fa elem készlet és a csempe készlet bővítését, valódi fa típusoknak megfelelő ágazatok generálását, textúrák használatát, fák animálását fizikai motor segítségével.

Irodalomjegyzék

[1] Dietrich, Andreas, Gerd Marmitt, and Philipp Slusallek. "Terrain guided multi-level instancing of highly complex plant populations." Interactive Ray Tracing 2006, IEEE Symposium on. IEEE, 2006.

[2] Nyisztor, „Shaderprogramozás Grafika és játékfejlesztés DirectX-szel”, SZAK Kiadó, 2009

[3] MSDN, Direct3D 11 Graphics (Hivatkozás)

[4] Texassemble (Hivatkozás)

[5] Efros, Alexei A., and William T. Freeman. "Image quilting for texture synthesis and transfer." Proceedings of the 28th annual conference on Computer graphics and interactive techniques. ACM, 2001.

[6] Alstes, Anton. Wang tiles for image and texture generation. Vol. 49432. Technical Report, 2004.

[7] Szijarto, Gabor, Jozsef Koloszar, and Magyar Tudósok Krt. "Real-time Hardware Accelerated Rendering of Forests at Human Scale." Proc. of WSCG, Pilsen, Czech Republic. 2004.

[8] Pirk, Sören, et al. "Plastic trees: interactive self-adapting botanical tree models." ACM Transactions on Graphics 31.4 (2012): 1-10.

[9] Boubekeur, Tamy, and Marc Alexa. "Phong tessellation." ACM Transactions on Graphics (TOG). Vol. 27. No. 5. ACM, 2008.

[10] Papazov, Chavdar. "Local, Polynomial G1 PN Quads." *International Symposium on Visual Computing*. Springer, Cham, 2014.