Avaruusjako tietokonegrafiikassa

Timo Heinonen
kandidaatintutkielma
tietojenkäsittelytiede
Turun yliopisto

Lokakuu 2016

Sisältö

1 Johdanto

Kolmiulotteinen tietokonegrafiikka on merkittävässä osassa animaatioelokuvissa, peleissä, sekä tietokoneavusteisessa suunnittelussa esimerkiksi arkktehtuurin ja teollisuuden alalla. Eräs aktiivinen tutkimuskohde on tietokonegrafiikan tekniikoiden soveltaminen konenäköön [Hughes et al., 2013] Tietokonegrafiikka on osin jopa syrjäyttämässä perinteistä valokuvaustyötä: huonekalujätti Ikea on siirtynyt käyttämään myyntikuvastoissaan valtaosin tietokoneella generoituja kuvia valokuvien sijaan. [CGSociety, 2014]

Kolmiulotteisen grafiikan piirtämistä kaksiulotteisiksi kuviksi kutsutaan renderoinniksi. Renderoinnin lähtökohtana on kuvattava maisema (engl. *scene*), joka sisältää objekteja ja valonlähteitä. Objektit ja valonlähteet on voitava mallintaa matemaattisesti, jotta niille voidaan määrittää sijainti ja suuntaus, ja niiden välisiä etäisyyksiä ja suhteita voidaan laskea. Renderointi tapahtuu aina jostakin kuvakulmasta, ja tätä varten määritellään kamera, jolla on oma sijaintinsa ja suuntauksensa maisemassa. Tämän jälkeen on selvitettävä, mitkä objektit kamera näkee, miten objekteihin osuvat valonsäteet vaikuttavat niiden väriin ja kuvan varjostukseen, ja lopuksi, mitkä värit projisoidaan kuvatason mihinkin pikseliin. [Janke, 2015]

1960-luvulla tietokonegrafiikkaa käytettiin lähinnä teollisuuden komponenttisuunnittelussa ja arkkitehtuurissa. Tietokoneella osattiin piirtää objektien ääriviivoja (engl. *wireframe*), mutta varjostustekniikoita ei tunnettu. IBM:n tutkija Arthur Appel esitteli algoritmin, joka laski suoran yhtälöitä, eli säteitä kuvasta maisemaan ja siitä valonlähteisiin. Tämän tekniikan avulla voitiin piirtää yksinkertaisia varjostuksia, ja myöhemmin tästä Ray Tracing:iksi kutsutusta tekniikasta tuli erittäin suosittu. [Appel, 1968]

Jo Appel totesi Ray Tracing -tekniikan olevan erittäin aikaavievä. [Appel, 1968] Vaikka tietokoneiden, ja varsinkin grafiikkaprosessoreiden laskentateho kasvaa jatkuvasti, ei grafiikan tuottaminen ole vieläkään halpaa tai nopeaa. Kuvista halutaan jatkuvasti fotorealistisempia, ja yksityiskohtaisemmat

kuvattavat mallit ja monimutkaiset valaisutekniikat vaativat erittäin paljon laskentatehoa. Esimerkiksi elokuvastudio Pixarin Monsterit-yliopisto -animaatioelokuvan piirtäminen vaati yli sata miljoonaa prosessoriuntia. [VentureBeat, 2013] Tämän takia renderoinnin nopeuttaminen on ollut aktiivinen tutkimuskohde jo pitkään.

1980-luvulla kehitettiin tekniikoita, joilla voitiin vähentää säteiden ja maiseman osumatarkasteluiden määrää. Steven Rubin ja Turner Whitted esittelivät tekniikan, jossa maisema jaetaan esiprosessointivaiheessa hierarkisesti suuntaissärmiöihin ihmisen toimesta. Säteiden ja suuntaissärmiöiden osumia tarkastelemalla voitiin vähentää operaatioiden kokonaismäärää. [Rubin and Whitted, 1980] Henry Fuchsin et al. metodiin kuului myös esiprosessointivaihe, tällä kertaa tietokoneen suorittamana. Maiseman objektit oli jaettu pienempiin osiin, polygoneihin, joista valittiin binääripuun juureksi sopiva. Tämän jälkeen binääripuuhun lisättiin polygoneja sillä perusteella, onko niiden vanhempi puussa etu- vai takapuolella. Näillä tekniikoilla renderointia saatiin nopeutettua siirtämällä työtä esiprosessointivaiheeseen. [Fuchs et al., 1980]

Tässä tutkielmassa esitellään avaruuden jakamiseen perustuvia tietorakenteita, joilla kolmiulotteisten kuvien renderointia voidaan nopeuttaa. Luvussa 2 määritellään joitakin grafiikan peruskäsitteitä, sekä esitetään algoritmi Ray Tracing -tekniikalle. Luvussa 3 tutkitaan Binary Space Partitioning -puuta, kd-puuta ja Bounding Volume -hierarkiaa, sekä niiden rakentamiseen ja läpikäyntiin liittyviä algoritmeja. Luvussa 4 selvitetään, miten kd-puuta voidaan käyttää optimoimaan Ray Tracing -algoritmia.

2 Kolmiulotteisen tietokonegrafiikan peruskäsitteitä

2.1 Määritelmiä

Kolmiulottesten kuvien renderoinnin kohteena ovat *objektit*, jotka mallintavat jotakin esinettä tai muotoa avaruudessa $\mathbb{R}^3$. Objektit voidaan esittää tietokoneen muistissa taulukkona pisteitä $(x, y, z)\,\epsilon\,\mathbb{R}^3$: esimerkiksi kolmiota voidaan kuvata kolmella pisteellä, ja palloa kahdella pisteellä, jotka esittävät sen keskipistettä ja yhtä pistettä sen pinnalla. [Angel and Shreiner, 2014]

Objektit jaetaan lähes kaikissa ei-triviaaleissa tapauksissa *polygoneihin*. Polygoni on samassa tasossa olevilla kärjillä rajattu alue, jonka kärkien muodostamat janat eivät leikkaa toisiaan. Useimmiten grafiikkasovelluksissa ja -rajapinnoissa valitaan polygonien muodoksi kolmiot, sillä niiden kolme kärkeä muodostavat aina tason, ja grafiikkaprosessorit osaavat operoida kolmioilla erittäin nopeasti. [Angel and Shreiner, 2014]

Jotta voitaisiin tarkastella objektien, valonlähteiden, ja kuvakulman, eli *kameran* välisiä suhteita ja suuntauksia avaruudessa, valitaan kolme koordinaatistoa, jotka on määritelty kolmella toisiinsa nähden kohtisuorilla kantavektoreilla. *Lokaalikoordinaatisto* sisältää yksittäisen objektin geometrian. Useimmitien origo sijoitetaan objektin keskipisteeseen. *Maailma-koordinaatisto* kuvaa koko avaruutta, ja sisältää tietoa siitä, mihin objektien lokaalikoordinaatistojen origot on sijoitettu. Lopuksi tämä maisema kuvataan virtuaalisella kameralla, jolla on oma *kamerakoordinaatistonsa*. Koordinaatistosta toiseen siirtyminen, koordinaatistojen skaalaus ja rotaatio voidaan toteuttaa lineaarikuvauksilla. [Janke, 2015]

2.2 Ray-Tracing -tekniikka

Ray Tracing on renderointitekniikka, jolla voidaan piirtää erittäin fotorealistisia kuvia. Ray Tracing -tekniikka pyrkii mallintamaan valonsäteitä, jotka

saavat alkunsa valonlähteistä, kulkevat avaruudessa ja osuvat objekteihin valaisten niitä, kimmoten niistä toisiin objekteihin ja muodostaen varjoja. Jotkut valonsäteet lopulta löytävät tiensä katsojan silmiin, eli kameraan. Koska olisi mahdotonta selvittää jokaisen valonsäteen kulkua avaruudessa, Ray Tracing -algoritmi ottaa huomioon vain ne säteet, jotka todella osuvat kameraan. Valonsäteitä seurataan siis käänteisessä järjestyksessä, kamerasta objekteihin, ja niistä valonlähteisiin. [Janke, 2015]

Ray Tracing -algoritmi muodostaa sille syötteenä annetusta kolmiulotteisesta maisemasta kameran sijainnin perusteella kaksiulotteisen kuvan. Jokaisen kuvatason pikselin läpi ammutaan säde $\vec{R} = O + t\vec{D}$, missä O on kameran sijainti maailma-koordinaatistossa, D kuvaa säteen kulkusuuntaa ja $t \epsilon \mathbb{R}$. Säteellä etsitään törmäyspistettä lähimmän objektin kanssa, eli sellaista mahdollisimman pientä arvoa t, että piste $P = O + t$ on jonkin objektin pinnalla. Tällöin osuman saaneen objektin piste P voi näkyä kameraan, mikäli siihen osuu valoa. Osumakohdasta ammutaan uusi, varjostussäteeksi kutsuttu säde. Jos varjostussäde osuu suoraan, tai kimmoten muista objekteista valonlähteeseen, lankeaa objektin pinnalle valoa. Tekniikan pseudokoodi on esitetty algoritmissa 1. [Janke, 2015]

Algoritmin suoritusnopeutta rajoittaa se, että jokaista sädettä kohti on käytävä läpi kaikki maiseman polygonit ja testattava osuuko säde niihin. Säteiden ja polygonien leikkauksien määrittämiseen joudutaan joissain tapauksissa käyttämään jopa 95% koko laskenta-ajasta. [Whitted, 1980] Algoritmia saataisiin siis nopeutettua huomattavasti, jos testattavien polygonien määrää jokaista sädettä kohti saataisiin vähennettyä. Yleisesti käytetty tapa leikkaustestien vähentämiseksi on muodostaa maisemasta ennen varsinaista renderointia hierarkinen tietorakenne, jota läpikäymällä saavutetaan nopeasti säteen leikkaama polygoni. [Rubin and Whitted, 1980]

Input:

Kuvataso: $x * y$ kokoinen taulukko pikseleitä

Maisema: joukko valonlähteitä ja polygoneihin jaettuja objekteja

Output:

Kolmiulotteinen maisema projisoituna kuvatasolle

```
1  foreach kuvatason pikseli (x, y) do
2      etaisyys ← ∞
3      foreach polygoni maisemassa do
4          Ammu säde R⃗ = O + tD⃗ kamerasta pikselin läpi maisemaan
5          if säde osui polygoniin pisteessä P and t < etäisyys then
6              Valon määrä V ← 0
7              foreach valonlähde L do
8                  Ammu varjostussäde R⃗ = L − P valonlähdettä kohti
9                  Kasvata valosummaa V
10             end
11             Aseta pikselin (x, y) väri valosumman V mukaisesti
12         else
13             Aseta pikseli (x, y) taustan väriseksi
14         end
15     end
16 end
17 return kuvataso
```

Algoritmi 1: Ray-Tracing -algoritmi

3 Avaruusjakopuut

3.1 BSP-puu

3.2 kd-puu

3.3 Bounding Volume Hierarchy

4 Renderoinnin optimoiminen avaruusjakopuiden avulla

4.1 kd-puuta käyttävä Ray-Tracing -algoritmi

Viittaus [Ranta-Eskola, 2001], [Samet, 2005] [Havran, 2000]

Viitteet

[Angel and Shreiner, 2014] Angel, E. and Shreiner, D. (2014). *Interactive Computer Graphics with WebGL*. Addison-Wesley Professional, 7th edition.

[Appel, 1968] Appel, A. (1968). Some techniques for shading machine renderings of solids. In *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference*, AFIPS '68 (Spring), pages 37–45, New York, NY, USA. ACM.

[CGSociety, 2014] CGSociety (2014). Building 3d with ikea. `http://www.cgsociety.org/index.php/cgsfeatures/cgsfeaturespecial/building_3d_with_ikea`. Luettu: 28.10.2016.

[Fuchs et al., 1980] Fuchs, H., Kedem, Z. M., and Naylor, B. F. (1980). On visible surface generation by a priori tree structures. *SIGGRAPH Comput. Graph.*, 14(3):124–133.

[Havran, 2000] Havran, V. (2000). *Heuristic Ray Shooting Algorithms*. PhD thesis, Czech Technical University, Praha, Tšekki.

[Hughes et al., 2013] Hughes, J. F., van Dam, A., McGuire, M., Sklar, D. F., Foley, J. D., Feiner, S. K., and Akeley, K. (2013). *Computer graphics: principles and practice (3rd ed.)*. Addison-Wesley Professional, Boston, MA, USA.

[Janke, 2015] Janke, S. J. (2015). *Mathematical Structures for Computer Graphics*. Wiley.

[Ranta-Eskola, 2001] Ranta-Eskola, S. (2001). Binary space partitioning trees and polygon removal in real time 3d rendering. Master's thesis, Uppsalan yliopisto, Uppsala, Ruotsi.

[Rubin and Whitted, 1980] Rubin, S. M. and Whitted, T. (1980). A 3-dimensional representation for fast rendering of complex scenes. *SIGGRAPH Comput. Graph.*, 14(3):110–116.

[Samet, 2005] Samet, H. (2005). *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[VentureBeat, 2013] VentureBeat (2013). How pixar made monsters university, its latest technological marvel. `http://venturebeat.com/2013/04/24/the-making-of-pixars-latest-technological-marvel-monsters-university/view-all/`. Luettu 30.10.2016.

[Whitted, 1980] Whitted, T. (1980). An improved illumination model for shaded display. *Commun. ACM*, 23(6):343–349.

Lista algoritmeista