Avaruusjako tietokonegrafiikassa

Timo Heinonen
LuK-tutkielma
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 [5] Tietokonegrafiikka on osin jopa syrjäyttämässä perinteistä valokuvaustyötä: huonekalujätti Ikea on siirtynyt käyttämään myyntikuvastoissaan valtaosin tietokoneella generoituja kuvia valokuvien sijaan. [2]

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. [6]

Vaikka tietokoneiden, ja varsinkin grafiikkaprosessoreiden laskentateho kasvaa jatkuvasti, ei grafiikan tuottaminen ole suinkaan 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 prosessorituntia. [10] Tämän takia renderoinnin nopeuttaminen on ollut aktiivinen tutkimuskohde jo pitkään.

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 kuvasta maisemaan ja siitä valonlähteisiin. Tämän tekniikan avulla voitiin piirtää yksinkertaisia varjostuksia, ja myöhemmin tästä Ray Tracing:iksi kutsutusta tekniikasta tuli erittäin suosittu. [1]

Jo Appel totesi Ray Tracing -tekniikan olevan erittäin aikaavievä. Vuonna 1980 julkaistiin kaksi artikkelia, joissa esiteltiin tapoja nopeuttaa tätä tekniikkaa. [3] [8]

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 3D-grafiikan peruskäsitteitä

2.1 Avaruus $\mathbb{R}^3$, objektit ja polygonit

2.2 Ray-Tracing tekniikka

Eräs suosittu tapa renderoida tietokonegrafiikkaa on Ray Tracing -tekiikka. Ray Tracing:illa pyritään mallintamaan mahdollisimman tarkasti valon kulkemista avaruudessa, heijastumista objekteista sekä tietysti sitä, mitkä objektit piirretään mihinkin kohtaan kuvaa.

Ray-Tracing -tekniikan pseudokoodi on esitetty algoritmissa 1.

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 jois-

Input:
Kuvataso: $x * y$ kokoinen taulukko pikseleitä
Maisema: joukko valonlähteitä ja polygoneihin jaettuja objekteja
Output:
Kolmiulotteinen maisema projisoituna kuvatasolle

```
1  foreach pikseli (x, y) näytöllä do
2      foreach polygoni maisemassa do
3          Ammu säde R = O + tD_1 kamerasta pikselin läpi maisemaan
4          if säde osui polygoniin pisteessä P then
5              foreach valonlähde L do
6                  Ammu varjostussäde R = L − P valonlähdettä kohti
7                  Kasvata pisteeseen P kohdistuvaa valosummaa
8              end
9              Aseta pikselin (x, y) väri valosumman mukaisesti
10         else
11             Aseta pikseli (x, y) taustan väriseksi
12         end
13     end
14 end
```

Algoritmi 1: Ray-Tracing -algoritmi

sain tapauksissa käyttämään jopa 95% koko laskenta-ajasta. [11] 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. [8]

3 Avaruusjakopuut

3.1 BSP-puu

3.2 kd-puu

3.3 Bounding Volume Hierarchy

4 Renderoinnin optimoiminen avaruusjakopuiden avulla

4.1 *-puuta käyttävä Ray-Tracing -algoritmi

Viittaus [7], [9] [4]

Viitteet

[1] Appel, Arthur: *Some Techniques for Shading Machine Renderings of Solids*. Teoksessa *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference*, AFIPS '68 (Spring), sivut 37–45, New York, NY, USA, 1968. ACM. http://doi.acm.org/10.1145/1468075.1468082.

[2] CGSociety: *Building 3D with Ikea*. http://www.cgsociety.org/index.php/cgsfeatures/cgsfeaturespecial/building_3d_with_ikea, kesäkuu 2014. Luettu: 28.10.2016.

[3] Fuchs, Henry, Zvi M. Kedem ja Bruce F. Naylor: *On Visible Surface Generation by a Priori Tree Structures*. SIGGRAPH Comput. Graph., 14(3):124–133, 1980, ISSN 0097-8930. http://doi.acm.org/10.1145/965105.807481.

[4] Havran, Vlastimil: *Heuristic Ray Shooting Algorithms*. väitöskirja, Czech Technical University, Praha, Tšekki, marraskuu 2000.

[5] Hughes, John F. *et al.*: *Computer graphics: principles and practice (3rd ed.)*. Addison-Wesley Professional, Boston, MA, USA, heinäkuu 2013, ISBN 0321399528.

[6] Janke, Steven J: *Mathematical Structures for Computer Graphics*. Wiley, 2015.

[7] Ranta-Eskola, Samuel: *Binary Space Partitioning Trees and Polygon Removal in Real Time 3D Rendering*. Pro Gradu -työ, Uppsalan yliopisto, Uppsala, Ruotsi, 2001.

[8] Rubin, Steven M. ja Turner Whitted: *A 3-dimensional Representation for Fast Rendering of Complex Scenes*. SIGGRAPH Comput. Graph., 14(3):110–116, 1980, ISSN 0097-8930. http://doi.acm.org/10.1145/965105.807479.

[9] Samet, Hanan: *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geo-*

metric Modeling). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005, ISBN 0123694469.

[10] VentureBeat: *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/, huhtikuu 2013. Luettu 30.10.2016.

[11] Whitted, Turner: *An Improved Illumination Model for Shaded Display.* Commun. ACM, 23(6):343–349, 1980, ISSN 0001-0782. http://doi.acm.org/10.1145/358876.358882.

Lista algoritmeista