Valoa tunneliin – Tekstuuritunnelin salat julki

Tekstuuritunneli on etenkin demoissa suosittu graafinen temppu, joka on käyttökelpoinen niin kasibittisillä kotimikroilla kuin shader-ohjelmoitavilla nykylaitteillakin.

Teksti: Ville-Matias Heikkilä Kuvat: Wallu, Ville-Matias Heikkilä

3D-grafiikka rakennetaan useimmiten monikulmioista, ja myös tekstuuritunnelin voi aivan hyvin toteuttaa näin. Tunnelit ovat kuitenkin nauttineet suurinta suosiota sellaisilla laitteilla, joissa kaiken toteuttaminen yleiskäyttöisellä 3D-moottorilla on joko hyvin hidasta tai suorastaan mahdotonta. Huomattava osa näiden laitteiden demoissa nähdystä kolmiulotteisuudesta perustuukin tästä syystä kikkoihin, joita on mahdollista käyttää vain onnekkaissa erikoistapauksissa. Tunnelin kohdalla tämä onnekkuus kumpuaa kappaleen muodosta ja jatkuvuudesta.

Tekstuuritunneleita on monenmuotoisia, mutta yleisin malli on äärettömän pitkä sylinteri. Tällainen kappale pysyy näytöllä vakiomuotoisena riippumatta siitä, millä tavoin katsoja liikkuu sen sisällä edestakaisin tai pyörii syvyysakselin ympäri, kunhan katse suuntautuu jatkuvasti äärettömyyteen. Ainoastaan sylinterin pinnassa oleva tekstuuri liikkuu.

Umpitunnelissa

Klassinen, kiinteän kameran tekstuuritunneli perustuu koordinaattitaulukkoon, joka yhdistää näytön jokaisen pikselin johonkin tekstuurin pikseliin. Tätä taulukkoa kutsutaan uv-kartaksi tai movelistiksi. Yksinkertaisimmillaan piirtosilmukka käy taulukon läpi seuraavaan tapaan:

```
for(i=0;i<320*200;i++)
        screen[i] = texture[uvmap[i]];
```

Tekstuurin koko ja suorittimen ominaisuudet määräävät, kannattaako liikkuminen ja pyöriminen toteuttaa tekstuuritaulukkoa vierittämällä vai piirtosilmukassa olevalla koordinaattiynäyksellä. Mikäli taas koneessa on niin iso uudelleen määriteltävä väripaletti, että koko tekstuuri mahtuu siihen, ei suoritinta tarvitse juuri vaivata, sillä kaiken voi toteuttaa paletinvierityksenä.

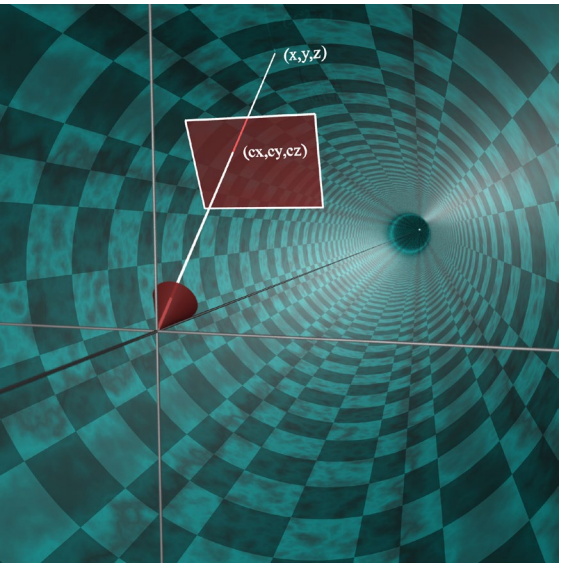
Taulukon koko puolittuu helposti käyttämällä hyväksi symmetriaa. Jos tekstuuri mahtuu reunan ympäri parillisen määrän kertoja, näytön loppupuoliskon pikselit ovat samat kuin alkupuoliskon pikselit mutta käänteisessä järjestyksessä. Jos muistia on hyvin vähän, voi uv-taulukosta tallentaa vaikkapa vain neljänneksen ja tuottaa loput koordinaatit peilausperiaatteella. Jos taas suoritin on hidas mutta muistia on riittävästi, voi olla kannattavaa rullata koko uv-taulukko auki koodiksi: ”lue tekstuurin pikseli 123, sijoita se näyttömuistin kohtiin 456 ja 789”.

Taulukkolaskentaa

Perinteinen sylinteritunneli asettuu näytölle siten, että pakopiste on näytön keskellä. Näin toinen tekstuurikoordinaateista riippuu siitä, kuinka lähellä piste on näytön keskipistettä. Toinen taas riippuu kulmasta, jonka pisteen sijainti muodostaa koordinaattiakselien suhteen. Eräänlainen perspektiivikorjattu napakoordinaattimuunnos siis. Tekstuurikoordinaatit voi siis laskea uv-taulukkoon seuraavasti, mikäli pakopiste on origossa:

```
u = atan2(x,y) * 256 / PI;
v = 100000 / sqrt(x*x + y*y);
```

Pakopiste ympäristöineen jää helposti röpelöiseksi, sillä v-koordinaatti lähestyy siinä kohti ääretöntä. Ongelman voi ratkaista peittämällä alue esimerkiksi mustalla ympyrällä tai käyttämällä sylinterin sijaan sopivasti kaartuvaa tunnelin muotoa, esimerkiksi rinkelin sisäpintaa. Myös erilaisia sumutyyppisiä häivytyksiä voi käyttää. Syvyyshäivytys ja muutamat muut efektit saattavat helpottua,



Vapaakameratunnelin geometriaa: pisteen (cx,cy,cz) läpi kameraan saapuva valonsäde on lähtöisin tunnelin reunalta pisteestä (x,y,z).

jos perspektiivikorjaus jätetään pois uv-taulukosta ja hoidetaan se tekstuuritaulukkoa täytettäessä.

Taulukkoperiaate sopii kaikenlaisten sorvimaisesti symmetristen kappaleiden pyörittelyyn symmetria-akselinsa ympäri. Sylinterien ja rinkelien ohella esimerkiksi pallo on ollut suosittu kappale. Muunlaiset kappaleet pyörivät samalla koodilla kuin tunnelikin; vain taulukko on laskettava eri kaavalla.

Vapaa liikkuminen

Jos koneessa on riittävästi tehoa, ei muistinvaraista uv-taulukkoa edes tarvita, vaan koordinaattimuunnoksen voi laskea lennossa. Näin tarjoutuu myös houkutteleva mahdollisuus tehdä muunnos hieman eri tavalla joka kuvalle, esimerkiksi vaihdella kameran sijaintia ja asentoa vapaammin. Freedir- eli vapaakameratunnelit ilmestyivät PC-demoihin 1990-luvun puolenvälin kieppeillä.

Vapaakameratunnelissa näytön pikseliä vastaava tekstuurikoordinaatti selvitetään yleensä raytracing-periaatteella: laskemalla, missä pisteessä suora ja sylinteri, eli valonsäde ja tunnelin reuna, leikkaavat toisensa.

Säteenseurannan matematiikkaa

Laskennan yksinkertaistamiseksi kannattaa asettaa niin paljon vakioita kuin mahdollista: kamera origoon, tunnelin syvyysakseliksi z-koordinaattiakseli ja tunnelin säteeksi 1. Tällöin yhtälöryhmän saa kuvattua peruskoulumatematiikalla seuraavasti:

```
x = d * cx
y = d * cy
z = d * cz
x*x + y*y = 1
```

Tässä piste (cx,cy,cz) kuuluu ilmassa leijuvaan suorakaiteeseen tai ”kameran linssiin”, jonka läpi valonsäteiden kuvitellaan kulkevan. Kutakin näytön pikseliä vastaavan suorakaiteen pisteen saa helposti pikselin näyttökoordinaateista. Z-koordinaatiksi kannattaa asettaa jokin samaa suuruusluokkaa oleva vakio, esimerkiksi näytön korkeus. Mitä pienempi z-koordinaatti, sitä laajempi kuvakulma saadaan. Kameran suuntaa vaihdetaan 3D-pyörittämällä suorakaiteen pisteitä.

Valonsädesuoran muut pisteet saa kertomalla koordinaatteja (cx,cy,cz) erilaisilla d:n arvoilla. Sopivalla arvolla löydetään piste, joka osuu sylinterin reunaan, joka on ympyrämäinen ja jonka yhtälö on siksi sama kuin ympyrän yhtälö.

Valmista kaavaa kansalle

Kun yhtälöryhmästä ratkaistaan osumapiste (x,y,z), niin d supistuu kokonaan pois ja saadaan suoraan koodiin kelpaavat kaavat:

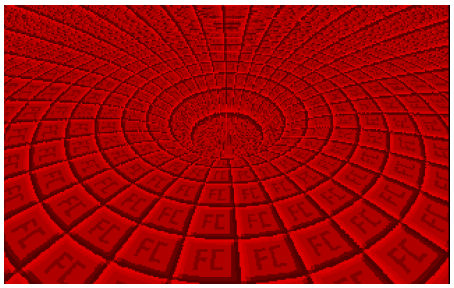
```
y = sqrt((cy*cy)/(cy*cy+cx*cx))
x = y*cx/cy
z = y*cz/cy
```

Syvyyskoordinaattia z voi käyttää toisena tekstuurikoordinaattina, ja toisen voi laskea arkustangentilla `atan2(x,y)`, kuten kiinteäkameratunnelissakin. Tekstuuri voi olla myös kolmiulotteinen, jolloin osumapistettä voi käyttää suoraan tekstuurikoordinaatteina. Testausvaiheessa ei välttämättä kannata vielä käyttää aitoa tekstuuria, vaan pikselin väriksi voidaan ottaa esimerkiksi koordinaattienvälisen xor-operaation tulos (C-syntaksissa `x^y^z`). Koska xor on kokonaislukuoperaatio, kannattaa koordinaatit kertoa sopivilla vakioilla.

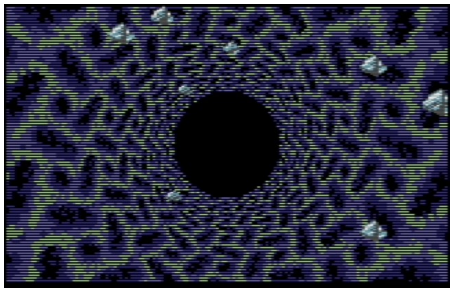
Optimointia ja ehostusta

Vaikka pikselikohtaista laskentaa voi optimoida jonkin verran pisteiden peräkkäisyyttä hyödyntämällä, jää laskettavaksi silti esimerkiksi neliöjuuri. Mikäli laitteen tehot eivät riitä, voidaan pistemäärää kuitenkin vähentää. Kun vapaakameratunneli yleistyi, yleisin ratkaisu oli laskea vain yksi leikkauspiste 8×8 pikseliä kohti kunnolla ja hoidella väleihin jäävät pisteet interpoloimalla eli keskiarvoperiaatteella. Jos tarjolla on nopea monikulmioiden tekstuurinpiirto, niin sitä kannattaa hyödyntää välialueiden täyttämisessä.

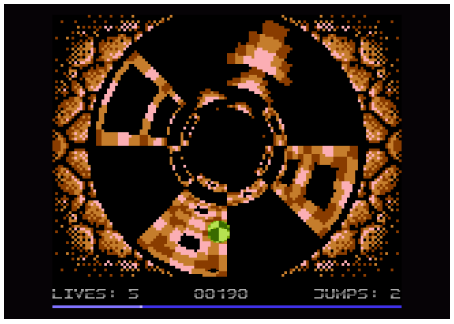
Tehokkailla nykykoneilla koko tunnelinpiirron voi toteuttaa shader-ohjelmana, joka ei käytä ulkoisia taulukoita. Tekstuurinkin voi määritellä suoraan kaavana, ja pakopisteröpelöt saa häivytettyä kauniisti esimerkiksi etäisyyssumulla. Toki shader-laitteistojen rahkeet riittävät paljon monimutkaisempienkin maailmojen raytracing-laskentaan, mutta se olisi jo toisen jutun aihe.



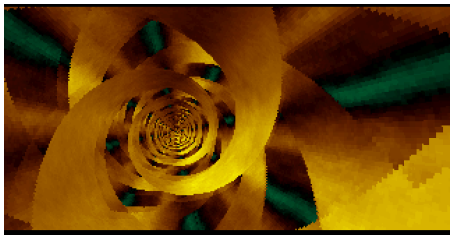
Paletinpyörityksellä toteutettu tekstuuritunneli Future Crew’n Unreal-demosta (1992).



Iso ja optimoitu sylinteritaulukkotunneli 64everryhmän C64-demosta Insomnia (2002).



Taulukkotunnelia on käytetty peleissäkin: Kasibittisen Atarin Yoomp (2007).



Sisäkkäisiä vapaakameratunneleita 256-tavuisessa PC-tuotoksessa: Tube by 3SC (2001).

Tätä vapaakameratunnelia on tehostettu aaltoilevilla kameravääristymillä. RainbowStars by Razor 1911 (PC, 2001)