



TAMPEREEN TEKNILLINEN YLIOPISTO

*Tietotekniikan koulutusohjelma*

**MARKO MOISIO**

**3D-VISUALISOINTI AJONEUVOSIMULAATTORISSA**

Diplomityö

Tarkastajat:

professori Seppo Pohjolainen

erikoistutkija Pekka Ranta

Tarkastajat ja aihe hyväksytty Tieto- ja

sähkötekniikan tiedekuntaneuvoston

kokouksessa 7.4.2010

# TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO

Tietotekniikan koulutusohjelma

**MOISIO, MARKO:** 3D-visualisointi ajoneuvosimulaattorissa

Diplomityö, 57 sivua

Huhtikuu 2010

Pääaine: Hypermedia

Tarkastajat: professori Seppo Pohjolainen, erikoistutkija Pekka Ranta

Avainsanat: 3D-visualisointi, simulaattori, koulutus

Nykyaikaisissa koulutussimulaattoreissa käytetään lähes poikkeuksetta 3D-visualisointia, jonka avulla havainnollistetaan koulutettavan kohteen liikkeitä ja ympäröivää virtuaalimaailmaa. Kehittyneet visualisointimenetelmät ja laitteisto mahdollistavat tarkkojen 3D-visualisointien esittämisen, mutta samalla visualisoinnin tekemiseen käytettävä aika on kasvanut. Tämän työn tarkoituksena on tutkia, mitä ajoneuvosimulaattoriin soveltuvalta 3D-visualisoinnilta vaaditaan ja miten se voidaan toteuttaa. Toinen aihealue on tutkia, miten 3D-visualisointiin vaadittava virtuaalimaailma voidaan rakentaa tehokkaasti ja miten rakennusprosessia voisi automatisoida.

Työ pohjautuu kirjallisuusselvitykseen 3D-visualisoinnista ja virtuaalimaailman luomisesta automatisoidusti. Selvityksessä tutkittiin koulutuskäyttöön tarkoitetun ajoneuvosimulaattorin 3D-visualisointia ja sen vaatimuksia niin ohjelmiston, kuin laitteistonkin osalta. Virtuaalimaailman luomista varten selvitettiin erilaisten semanttisten tietojen soveltumista 3D-mallien automaattiseen luomiseen. Selvityksen pohjalta toteutettiin yksinkertainen käytännön sovellus, jossa luodaan automatisoinnin keinoin ajoneuvosimulaattoriin soveltuva virtuaalimaailma.

Kirjallisuusselvityksen ja käytännön sovelluksen perusteella voidaan todeta, että nykyään 3D-visualisoinnin esittäminen on melko helppoa valmiiden visualisointikirjastojen avulla. Visualisointikirjastot ovat käytettävyydeltään hyviä, mutta tarjoavat silti monipuolisesti ominaisuuksia erilaisten 3D-visualisointien tekemiseen. Sen sijaan visualisointiin tarvittavan virtuaalimaailman rakentaminen on huomattavasti työläämpää. Rakentamisprosessin tiettyjä osia automatisoimalla voidaan tarvittavan käsityön määrää vähentää. 3D-mallien tekeminen on yksi suuritöisimmistä osa-alueista, joten niiden automaattinen generointi on tehokas tapa nopeuttaa virtuaalimaailman rakentamisprosessia.

## ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY

Master's Degree Programme in Information Technology

**MOISIO, MARKO:** 3D Visualisation in Driving Simulator

Master of Science Thesis, 57 pages

April 2010

Major: Hypermedia

Examiners: Professor Seppo Pohjolainen, Senior Researcher Pekka Ranta

Keywords: 3D visualisation, simulator, training

In modern training simulators 3D visualisation is widely used to visualise the movements of training vehicle in virtual environment. Advanced visualisation software and hardware have made it possible to use very detailed and large 3D visualisations but in the same time creating virtual environments for 3D visualisation takes lot of time. There are two main goals for this thesis. First one is to examine how to create 3D visualization suitable for driving simulators. The second goal is to study how to efficiently create virtual environment that can be used in driving simulator and how the process of building the virtual environment can be automated.

This thesis is started with studying books related to 3D visualisation in both software and hardware. More specific area was to study what requirements driver training simulators give for visualisation. For creation of virtual environment it was studied how semantic information can be used for automating the creation of 3D models. Based on these studies simple software for creating and visualising virtual environment was made. This software uses automated techniques to create the virtual environment.

As a result of this thesis it can be said that basic 3D visualisation is quite easy to do with visualisation libraries. Modern 3D visualisation libraries are easy to use and offer many useful functions for different needs. The part that takes more time and effort is building suitable virtual environment for driving simulator. The building process can be made much quicker by automating certain parts of building process. One of the most time consuming parts is creation of all 3D models needed for large virtual environment. Lot of time can be saved by automating the creation of 3D models.

## ALKUSANAT

Tämä diplomityö on tehty Tampereen teknillisen yliopiston Hypermedialaboratoriossa osana Smart Simulators -tutkimusryhmän tutkimusprojektia ajoneuvosimulaattorin liikenneympäristön 3D-visualisoinnista ja sen automaattisesta generoinnista. Diplomityön tekemisen aikana olen saanut työskennellä Hypermedialaboratoriossa ajoneuvosimulaattoreita käsittelevissä projekteissa, jotka ovat osaltaan tukeneet työn kirjoittamista.

Haluan kiittää Seppo Pohjolaista ja Pekka Rantaa mahdollisuudesta tämän mielenkiintoisen diplomityön tekemiseen, sekä kaikesta saamastani tuesta ja palautteesta työn tekemisen aikana. Kiitokset myös työtovereilleni Hypermedialaboratoriossa mukavasta ja kannustavasta työilmapiiristä. Lopuksi haluan kiittää vanhempiani ja muita sukulaisiani opintojeni tukemisesta.

Tampereella 13.4.2010

Marko Moisio

# SISÄLLYS

Tiivistelmä .....	II
Abstract .....	III
Alkusanat .....	IV
Termit ja niiden määritelmät .....	VII
1. Johdanto .....	1
2. Simulaattorit .....	2
2.1. Simulaattoreista yleisesti .....	2
2.2. Esimerkkejä koulutussimulaattoreista ja visualisointijärjestelmistä .....	6
2.3. Skenaariot .....	9
2.4. Laitteisto .....	10
2.5. Ohjelmisto .....	12
3. 3D-visualisoinnin teoria .....	14
3.1. 3D-mallit .....	14
3.2. Geometriset muunnokset .....	15
3.2.1. Siirto .....	15
3.2.2. Kierto .....	16
3.2.3. Skaalaus .....	16
3.3. 3D-maailman rakenne .....	16
3.3.1. Scene graph tietorakenne .....	17
3.4. 3D-piirron liukuhihna .....	17
3.4.1. Sovellusvaihe .....	18
3.4.2. Geometriavaihe .....	18
3.4.3. Rasterointivaihe .....	18
3.5. Grafiikkarajapinnat .....	19
3.5.1. OpenGL .....	19
3.5.2. Direct3D .....	19
3.6. 3D-visualisointikirjastot .....	20
3.7. Optimointi .....	20
3.7.1. Näkymättömien objektien rajaus .....	21
3.7.2. LOD (level of detail) .....	22
4. Ajoneuvosimulaattorin visualisoinnin erityispiirteet .....	24
4.1. Suorituskykyvaatimukset .....	24
4.2. Realistinen grafiikka .....	25
4.3. Lisätty todellisuus .....	26
4.4. Monikanavaisuus .....	26
4.5. Näyttölaitteet .....	27
4.5.1. LCD näyttö .....	27
4.5.2. Projektori .....	28
4.5.3. Virtuaalikypäriä .....	28
4.6. Simulaattorisairaus .....	29

4.6.1. Simulaattorisairauden torjuminen.....	30
5. Virtuaalimaailman luominen.....	31
5.1. Paikkatieto.....	32
5.2. Paikkatietoformaatit .....	35
5.3. Automaattinen 3D-mallien generointi.....	36
5.4. Maaston luominen .....	36
5.5. Tiestön luominen.....	37
5.5.1. Tiesuunnitelmien hyödyntäminen.....	38
5.6. Objektien luominen .....	39
5.7. Semanttisen tiedon hyödyntäminen .....	40
6. Käytännön toteutus.....	42
6.1. Hyödynnetyt ohjelmistot.....	43
6.2. Tallennusformaatit .....	43
6.3. Maaston visualisointi .....	44
6.4. Tiestön visualisointi .....	47
6.5. Objektien visualisointi .....	49
6.6. Lopputulos .....	51
7. Arviointi ja jatkokehitys.....	52
8. Yhteenveto .....	54
Lähteet.....	55

# TERMIT JA NIIDEN MÄÄRITELMÄT

<b>3D-malli</b>	Käytetään 3D-visualisoinnissa esineiden kuvaamiseen, koostuu polygoneista
<b>3D-piirron liukuhina</b>	3D-piirron vaiheet peräkkäisessä järjestyksessä
<b>3D-visualisointi</b>	Tässä työssä 3D-visualisoinnilla tarkoitetaan tietokoneella esitettävää kolmiulotteista grafiikkaa
<b>ASCII</b>	American Standard Code for Information Exchange, merkistö, joka sisältää 128 merkkiä
<b>CAD</b>	Computer Aided Design
<b>DEM</b>	Digital Elevation Model, korkeuskartta
<b>Direct3D</b>	Microsoftin kehittämä matalan tason grafiikkarajapinta
<b>GIS</b>	Geographical Information System, paikkatietojärjestelmä
<b>Grafiikkamoottori</b>	Ohjelmistokirjasto, joka on tarkoitettu grafiikan esittämiseen
<b>Korkeuskartta</b>	Kuvamuotoinen kuvaus maaston pinnanmuodoista
<b>Koulutussimulaattori</b>	Koulutuskäyttöön tarkoitettu simulaattori
<b>LCD</b>	Liquid Crystal Display, näyttötekniologia
<b>LOD</b>	Level of detail, optimointimenetelmä 3D-visualisoinnissa
<b>OGRE</b>	Object-Oriented Graphics Rendering Engine, avoimen lähdekoodin 3D-visualisointikirjasto
<b>OpenGL</b>	Silicon Graphicsin kehittämä matalan tason grafiikkarajapinta
<b>Piste</b>	Avaruuden piste, jonka sijainti kerrotaan kolmella koordinaatilla
<b>Polygoni</b>	Kolmen pisteen rajaama kolmio
<b>Simulaattori</b>	Tekninen laite tai tietokoneohjelma, joka jäljittelee eli simuloi jonkin laitteen tai järjestelmän toimintaa
<b>XML</b>	Extensible Markup Language, merkintäkieli

# 1. JOHDANTO

Tietyillä toimialoilla simulaattoreita on jo pitkään käytetty koulutuksessa ja nykyään yhä useammalla toimialalla ollaan siirtymässä osittain simulaattoripohjaiseen koulutukseen. Kehittyneen 3D-visualisoinnin avulla simulaattoreista saadaan tehtyä varsin aidon näköisiä ja samalla myös havainnollisia koulutusvälineitä. Koulutuskäytössä simulaattorilta vaaditaan erilaisia ominaisuuksia verrattuna tutkimussimulaattoreihin. Koulutussimulaattoreissa on arviointi ja hallintatoiminnot, joiden avulla kouluttaja pystyy seuraamaan harjoituksen suoritusta. Koulutussimulaattoreissa ja erityisesti ajoneuvosimulaattoreissa on usein käytössä laaja virtuaalimaailma, jossa voidaan harjoitella monipuolisesti erilaisia tilanteita. Virtuaalimaailman tuottaminen käsityönä on työlästä ja aikaa vievää, joten luonnollisena suuntana on automatisoida virtuaalimaailman muodostamista.

Tämä työ käsittelee 3D-visualisointia ajoneuvosimulaattoreissa. Työ keskittyy erityisesti koulutuskäyttöön tarkoitettujen ajoneuvosimulaattoreiden visualisointiratkaisuihin. Tarkempia tarkastelukohteita ovat 3D-visualisoinnin toteuttaminen visualisointikirjastojen avulla, virtuaalimaailman rakentaminen ja rakentamisen automatisoiminen. Koko rakentamisprosessia on hyvin vaikeaa automatisoida, mutta jo tiettyjen suuritöisten vaiheiden automatisoinnilla voidaan saavuttaa huomattavaa ajallista säästöä. Yksi suuritöisimmistä vaiheista on 3D-mallien tekeminen, joten 3D-mallien automaattinen generointi on yksi tämän työn keskeisimmistä tavoitteista. Automaattista generointia varten tarvitaan aineisto, jonka perusteella malli luodaan. Tässä työssä käytetään paikkatietoaineistoa 3D-mallien lähtötietona.

Luvussa kaksi käsitellään simulaattoreita ja niiden yleisrakennetta niin ohjelmiston, kuin laitteistonkin osalta. Erityishuomiota kiinnitetään koulutussimulaattoreihin ja skenaarioihin. Luvussa kolme käydään läpi 3D-visualisoinnissa tarvittavat periaatteet ja tärkeimmät käsitteet. Luku neljä keskittyy 3D-visualisointiin ajoneuvosimulaattoreissa ja siihen, mitä erityispiirteitä niiden visualisoinnissa on. Luku viisi keskittyy virtuaalimaailman luomisen automatisointiin. Automatisoinnissa paikkatieto ja 3D-mallien automaattinen generointi ovat keskeisimmät aiheet. Luvussa kuusi on toteutettu yksinkertainen virtuaalimaailman 3D-visualisointi hyödyntäen 3D-mallien automaattista generointia. Käytännön toteutuksessa on käytetty avoimen lähdekoodin ohjelmistoja. Luvussa seitsemän on arvioitu käytännön toteutuksen ratkaisuja ja niiden soveltuvuutta todelliseen simulaattorikäyttöön. Luvussa kahdeksan on esitetty lyhyt yhteenveto työstä.

## 2. SIMULAATTORIT

### 2.1. Simulaattoreista yleisesti

Simulaattoreita käytetään useisiin eri tarkoituksiin, mutta yleisesti simulaattorilla tarkoitetaan jonkin reaali maailman ilmiön jäljittelyä teknisellä laitteella. Simulaattori voidaan määritellä esimerkiksi seuraavasti: ”Simulaattori on teknisen laitteen, järjestelmän, yksittäisen henkilön tai ryhmän toiminnan jäljittelyyn eli simulointiin tarkoitettu tavallisesti tekninen laite tai elektroninen tutkimus-, opetus- tai tietokoneohjelma” [Peltoniemi 2001]. Peltoniemen määritelmä kuvaa ensisijaisesti sotilaskäyttöön tarkoitettuja simulaattoreita, mutta se soveltuu hyvin myös muun tyyppisille simulaattoreille. Ajoneuvosimulaattorit pyrkivät jäljittelemään tietyn tyyppisen ajoneuvon käyttäytymistä erilaisissa ajo-olosuhteissa ja liikennetilanteissa. Ajoneuvosimulaattorissa on tyypillisesti vastaavat hallintalaitteet, kuin oikeassa ajoneuvossa.

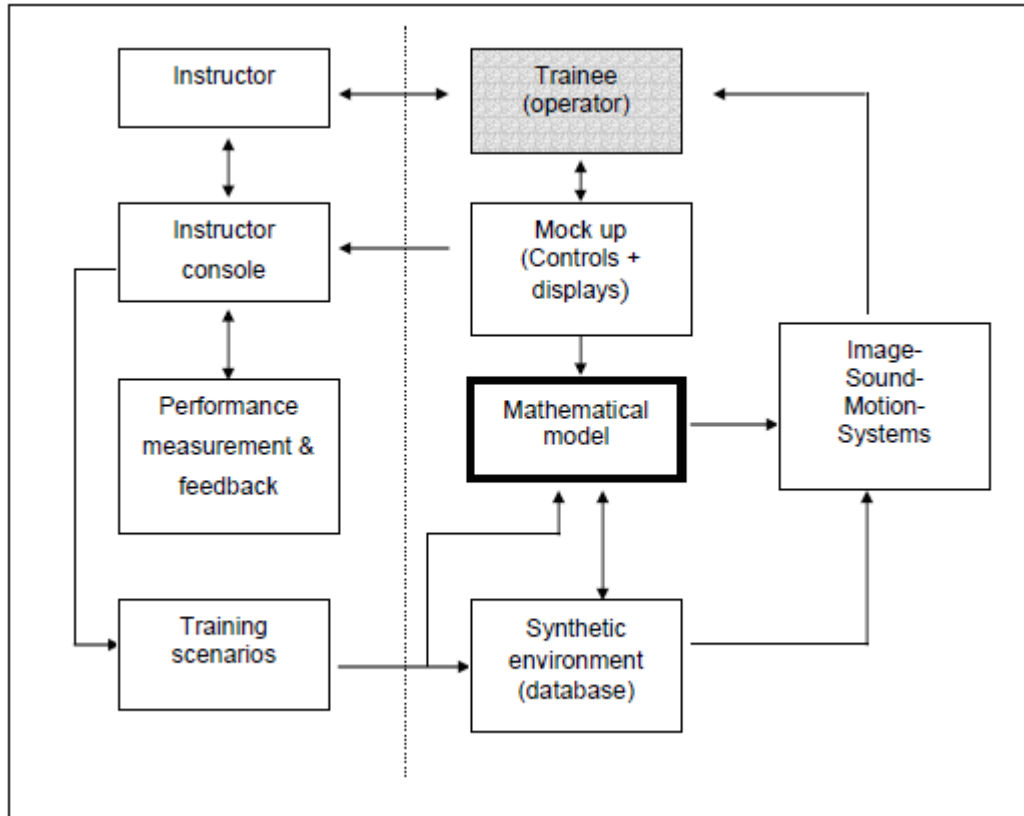
Simulaattorit voidaan jakaa kahteen eri ryhmään käyttötarkoituksen mukaan: tutkimus- ja koulutussimulaattorit [van Emmerik 2004]. Eri käyttökohteet vaativat simulaattorilta erilaisia ominaisuuksia. Tutkimuskäytössä simulaattorin tärkein ominaisuus on usein erittäin tarkka fysiikan mallinnus. Koulutuskäytössäkin pitää tuki olla hyvä fysiikanmallinnus, mutta yhtä tarkeässä asemassa ovat hyvät koulutusominaisuudet, jotka vaativat simulaattorilta hyvää käytettävyyttä ja tarkoitukseen soveltuvaa virtuaaliympäristöä. Koulutussimulaattoreissa on lisäksi arviointi ja hallintatoiminnot, joiden avulla kouluttaja pystyy seuraamaan suoritusta [van Emmerik 2004].

Koulutussimulaattoreita on jo pitkään hyödynnetty tietyillä toimialoilla. Erityisesti lentokonesimulaattorit ovat pitkän kehityksen tulos, sillä ensimmäiset lentokonesimulaattorit otettiin käyttöön jo vuonna 1910 [Teikari & Vartiainen 1985]. Myös merenkulussa, lääketieteessä ja sotilaskäytössä on simulaattoreita hyödynnetty jo pitkään koulutustarkoituksessa. Nykyään simulaattorit ovat useimmiten tietokonepohjaisia ratkaisuja. Tietokoneiden yleistyminen ja samalla hintojen aleneminen on mahdollistanut simulaattoreiden hyödyntämisen myös monilla muilla toimialoilla. Nykyään esimerkiksi autokoulut hyödyntävät simulaattoreita opetuksessa.

Kuvassa 2.1 on esitetty simulaattorin yleisrakenne, jossa koulutussimulaattorille ominaiset osat on esitetty katkoviivan vasemmalla puolella. Opettaja (Instructor) on merkittävässä roolissa koulutussimulaattorin tehokkaassa käytössä, sillä ilman ohjausta opiskelijan (Trainee) on vaikea oppia oikeita toimintatapoja simulaattorilla. Opettaja ohjaa simulaattorin käyttöä ja tarkkailee opiskelijan suoritusta opettajan työvälineiden (Instructor console) avulla. Erityisen tärkeää koulutuskäytössä on, että simulaattorilla

suoritettavat skenaariot on suunniteltu siten, että niiden avulla on mahdollista oppia opetussuunnitelmassa määritellyt asiat. [van Emmerik 2004]

Kuvan 2.1 mukaisesti skenaariot määrittelevät suureksi osaksi sen, mitä visualisoinnissa käytettävät aineistot ovat ja miten ne esitetään. Skenaariossa valitaan käytettävät objektit ja tapahtumapaikka (Synthetic environment) ja ne esitetään visualisointijärjestelmän (Image-system) avulla.



**Kuva 2.1 Simulaattorin yleisrakenne. Katkoviivan vasemmalla puolella on koulutussimulaattorille ominaiset osat [van Emmerik 2004]**

Koulutussimulaattoreita käytetään opetuksessa ja erilaisissa harjoituksissa. Simulaattorit ovat usein hyvin samankaltaisia oikeiden laitteiden kanssa, joten niiden avulla voidaan saavuttaa hyvä siirtovaikutus, eli taitojen siirtyvyys oikean laitteen käyttöön. Hyvän siirtovaikutuksen saavuttamiseksi on tärkeää, että simulaattorilla suoritettavat harjoitustehtävät, eli skenaariot sisältävät todellisen tehtävän kannalta oleelliset taitovaatimukset. [Teikari & Vartiainen 1985]

Autolla ajamiseen tarvittavia taitoja ja niiden opettamista on tutkittu TRAINER (System for driver Training and Assessment using Interactive Evaluation tools and Reliable methodologies) EU-projektissa. TRAINER -tutkimuksessa on esitetty taulukko (kuva 2.2), jossa ajamiseen liittyvät asiat on jaettu neljään tasoon.

- Taulukon alin taso (Vehicle manouvering) sisältää ajoneuvon käsittelyyn tarvittavia perustaitoja. Tämän tason asiat ovat perusvaatimuksena onnistuneelle ajosuoritukselle.

- Toiseksi alin taso (Mastery of traffic situations) käsittelee liikenteessä tarvittavia taitoja, kuten riskien tunnistamista.
- Toiseksi ylin taso (Driving goals and context) käsittelee ajamisen taustalla olevia asioita, kuten ajankohdan valintaa ja matkan tarkoitusta.
- Ylin taso (Goals for life and skills for living) käsittelee hyvin laajasti elämässä tapahtuvien asioiden vaikutusta ajokäyttäytymiseen.

		Essential curriculum		
		Knowledge and skills	Risk-increasing factors	Self-evaluation
<b>Hierarchical levels of behaviour</b>	<b>Goals for life and skills for living (general)</b>	<b>Knowledge about/control over how life-goals and personal tendencies affect driving behaviour</b> <ul style="list-style-type: none"> <li>lifestyle/life situation</li> <li>peer group norms</li> <li>motives</li> <li>self-control, other characteristics</li> <li>personal values</li> <li>...</li> </ul>	<b>Risky tendencies</b> <ul style="list-style-type: none"> <li>acceptance of risks</li> <li>self-enhancement through driving</li> <li>high level of sensation seeking</li> <li>complying with social pressure</li> <li>use of alcohol and drugs</li> <li>values, attitudes towards society</li> <li>...</li> </ul>	<b>Self-evaluation/awareness of</b> <ul style="list-style-type: none"> <li>personal skills for impulse control</li> <li>risky tendencies</li> <li>safety-negative motives</li> <li>personal risky habits</li> <li>...</li> </ul>
	<b>Driving goals and context (journey-related)</b>	<b>Knowledge and skills concerning</b> <ul style="list-style-type: none"> <li>effects of journey goals on driving</li> <li>planning and choosing routes</li> <li>evaluation of requested driving time</li> <li>effects of social pressure inside the car</li> <li>evaluation of necessity of the journey</li> <li>...</li> </ul>	<b>Risks connected with</b> <ul style="list-style-type: none"> <li>driver's condition (mood, BAC, etc.)</li> <li>purpose of driving</li> <li>driving environment (rural/urban)</li> <li>social context and company</li> <li>additional motives (competitive, etc.)</li> <li>...</li> </ul>	<b>Self-evaluation/awareness of</b> <ul style="list-style-type: none"> <li>personal planning skills</li> <li>typical driving goals</li> <li>typical risky driving motives</li> <li>...</li> </ul>
	<b>Mastery of traffic situations</b>	<b>Knowledge and skills concerning</b> <ul style="list-style-type: none"> <li>traffic regulations</li> <li>observation/selection of signals</li> <li>anticipation of the development of situations</li> <li>speed adjustment</li> <li>communication</li> <li>driving path</li> <li>driving order</li> <li>distance to others/safety margins</li> <li>...</li> </ul>	<b>Risks caused by</b> <ul style="list-style-type: none"> <li>wrong expectations</li> <li>risk-increasing driving style (e. g. aggressive)</li> <li>unsuitable speed adjustment</li> <li>vulnerable road-users</li> <li>not obeying regulations/unpredictable behaviour</li> <li>information overload</li> <li>difficult conditions (darkness, etc.)</li> <li>insufficient automatism or skills</li> <li>...</li> </ul>	<b>Self-evaluation/awareness of</b> <ul style="list-style-type: none"> <li>strong and weak points of basic traffic skills</li> <li>personal driving style</li> <li>personal safety margins</li> <li>strong and weak points for hazard situations</li> <li>realistic self-evaluation</li> <li>...</li> </ul>
	<b>Vehicle manoeuvring</b>	<b>Knowledge and skills concerning</b> <ul style="list-style-type: none"> <li>control of direction and position</li> <li>tyre grip and friction</li> <li>vehicle properties</li> <li>physical phenomena</li> <li>...</li> </ul>	<b>Risks connected with</b> <ul style="list-style-type: none"> <li>insufficient automatism or skills</li> <li>unsuitable speed adjustment</li> <li>difficult conditions (low friction, etc.)</li> <li>...</li> </ul>	<b>Awareness of</b> <ul style="list-style-type: none"> <li>strong and weak points of basic manoeuvring skills</li> <li>strong and weak points of skills for hazard situations</li> <li>realistic self-evaluation</li> <li>...</li> </ul>

Kuva 2.2 Ajamiseen tarvittavat taidot [TRAINER D.2.1]

TRAINER -tutkimuksen mukaan taulukon kahden alimman tason asioita harjoitellaan yleisesti autokouluissa Euroopassa sekä teoriassa, että käytännössä. Näitä asioita voidaan hyvin harjoitella myös simulaattorissa. Kahden ylimmän tason asioita harjoitellaan pääasiassa teoriatasolla, joten niiden harjoittelu simulaattorilla on

hankalampaa. Joitain näistäkin asioista voidaan kuitenkin harjoitella myös simulaattorilla.

Alimmalla tasolla harjoitellaan ajoneuvon käsittelyä. Visualisoinnin kannalta oleellista tämän tason harjoituksille on oman ajoneuvon, sekä käsittelyharjoituksiin sopivien ympäristöjen esittäminen. Toisella tasolla harjoitellaan liikenneympäristössä ajamista, joten myös visualisoinnissa pitää pystyä esittämään monipuolinen liikenneympäristö, joka sisältää muun muassa tieverkoston, liikennemerkkit ja muut ajoneuvot.

Koulutussimulaattoria tulisi käyttää perinteisten opetusmenetelmien rinnalla sellaisten asioiden harjoitteluun, joihin simulaattori parhaiten soveltuu. Tietyissä tilanteissa koulutussimulaattorin käytöllä voidaan saavuttaa huomattavia etuja verrattuna oikealla ajoneuvolla harjoitteluun [TRAINER D.4.1]:

- Vaarallisia tilanteita voidaan harjoitella turvallisessa ympäristössä.
- Harjoittelijan suoritusta voidaan seurata ja mitata tarkemmin.
- Samaa harjoitusta voidaan toistaa useita kertoja samanlaisissa olosuhteissa. Näin taataan tasapuolinen oppimisympäristö eri harjoittelijoille.
- Simulaattorissa voidaan harjoitella sellaisissa olosuhteissa, jotka eivät muuten olisi mahdollisia, esimerkiksi liukkaankelin ajoharjoittelu kesällä.

Simulaattorin käytöllä voi olla myös haittapuolia, jotka saattavat vaikuttaa opiskelijan taitoihin siirryttäessä oikean laitteen käyttöön [TRAINER D.4.1]:

- Simulaattori ei vastaa täysin oikeaa laitetta
- Siirtovaikutus ei ole yhtä hyvä kuin oikealla laitteella harjoiteltaessa
- Simulaattorin käyttö voi johtaa väärin toimintatapojen kehittymiseen
- Simulaattorilla voidaan opettaa vain tiettyjä työprosessin osia

Haittapuolien vaikutusta voidaan kuitenkin pienentää riittävällä ohjauksella ja täydentämällä koulutusta muilla keinoin. Asiantunteva kouluttaja on tärkeä tekijä onnistunutta simulaattorikoulutusta ajatellen.

## **2.2. Esimerkkejä koulutussimulaattoreista ja visualisointijärjestelmistä**

Tässä luvussa on esitelty muutamia eri toimialojen koulutussimulaattoreita ja visualisointijärjestelmiä. Esitellyt simulaattorit ovat yleisesti käytössä toimialojen opiskelijoiden, sekä myös kokeneempien ammatinharjoittajien koulutuksessa.

### **John Deere metsäkonesimulaattori**

John Deere on kehittänyt metsäkonesimulaattorin, jolla on mahdollista harjoitella virtuaalisesti harvesterin ja kuormakoneen käyttöä. Simulaattori sisältää oikean metsäkoneen ohjauslaitteet ja yhden näyttölaitteen. Simulaattori esittää 3D-visualisoinnin avulla metsäkoneet, sekä työskentelykohteena olevan metsän. Simulaattoriin sisältyy myös tehtäväeditori, jolla on mahdollista luoda uusia harjoitustehtäviä ja korjuukohteita. Editorissa on mahdollista muokata virtuaalimaailmaa monipuolisesti. Editorilla voi muokata muun muassa maaston muotoja ja puiden sijoittelua. [John Deere]



**Kuva 2.3 John Deere metsäkonesimulaattori [John Deere]**

### **Faros pimeäajosimulaattori**

Pimeäajosimulaattori on tarkoitettu pimeäajon harjoitteluun autokouluissa ja muissa koulutustilaisuuksissa. Simulaattori perustuu Faros F2300 -laitteistoon, johon Suomen Autokoululiitto on yhdessä Faroksen kanssa kehittänyt pimeäajo-ohjelmiston. Simulaattoriin on laadittu autokoulun opetussuunnitelman mukainen opetussisältö. Opetusohjelma jakaantuu esittelyyn näyttöön ja harjoitteluun. Esittelyssä kerrotaan mitä valitussa tehtävässä on tarkoitus harjoitella. Näytössä käydään yksityiskohtaisesti läpi, mitä kuljettajan pitää harjoituksen aikana tehdä. Harjoittelussa kuljettaja suorittaa

harjoituksen aiemmissa vaiheissa opittujen ohjeiden mukaisesti. Simulaattori osaa antaa myös palautetta suorituksesta.

Simulaattori sisältää normaalin henkilöauton keskeisimmät hallintalaitteet. Simulaattorissa käytetään kolmea näyttöä, joilla esitetään etu- ja sivunäkymät. Kolmen näytön ratkaisulla saavutetaan riittävän leveä näkökenttä pimeääjon tehokkaaseen harjoitteluun. Simulaattori on käytössä useissa autokouluissa Suomessa. [Autokoululiitto]



**Kuva 2.4 Faros pimeääjosimulaattori [Faros]**

### **Simrac bussisimulaattori**

Simrac valmistaa koulutuskäyttöön tarkoitettuja bussisimulaattoreita. Simulaattori sisältää kaikki oikean bussin hallintalaitteet ja perustuu myös oikean bussin koriin. Simulaattorissa on myös liikealusta, joka mahdollistaa todentuntuisen ajokokemuksen. Simulaattorilla pystyy harjoittelemaan ajamista laajassa liikenneympäristössä ja vaihtelevissa sääolosuhteissa. Simulaattorin visualisointi on toteutettu usealla videotykillä, ja näin saavutetaan laaja, jopa 240 asteen näkökenttä. Simulaattori sisältää myös harjoituksen arviointi- ja läpikäyntitoiminnot. [Simrac]



Kuva 2.5 Simrac bussisimulaattori [Simrac]

### **Insta SimCore visualisointijärjestelmä**

Insta on kehittänyt SimCore -nimisen visualisointijärjestelmän, jonka avulla voidaan tehokkaasti visualisoida laajoja alueita. SimCore on kehitetty erityisesti lentokonesimulaattoreita varten, mutta sitä voidaan käyttää myös monissa muissa sovelluksissa. SimCore käyttää visualisointiin visualisointitietokantaa, joka on rakennettu automaattisesti käyttäen pohjana paikkatietomateriaalia. Automaattinen rakennusprosessi mahdollistaa suurtenkin alueiden luomisen visualisointitietokantaan kohtuullisessa ajassa. Järjestelmällä on mahdollista visualisoida esimerkiksi koko Suomen kattava alue. [Insta]

### **2.3. Skenaariot**

Skenaariolla tarkoitetaan koulutussimulaattorilla suoritettavaa koulutuksellista kokonaisuutta. Skenaario sisältää harjoituksen lähtötilanteen, tavoitteet, harjoituksen kulun, sekä arvioinnin ja palautteen. Lähtötilanne ja harjoituksen tarkoitus ovat tärkeässä asemassa oppimisen kannalta, sillä ilman näitä tietoja opiskelija ei välttämättä osaa yhdistää simulaattorilla harjoiteltavia asioita todellisen ympäristön tilanteisiin. Skenaario kuvaa tyypillisesti yhden tai useamman toisiinsa liittyvän tosi elämän tilanteen, jotka tulee suorittaa simulaattorilla. Skenaarioiden suunnittelu on tärkeässä roolissa simulaattorikoulutuksessa, sillä skenaarioiden sisältö ratkaisee kuinka hyvin

oppimistuloksiin simulaattorin käytöllä päästään. Skenaariot suunnitellaan yleensä opetussuunnitelman pohjalta. [Alessi & Trollip 1985]

Skenaariossa oppimistehtävän eteneminen kuvataan usein tapahtumien (event) avulla. Ajoneuvosimulaattorissa tapahtumat vaikuttavat harjoituksen etenemiseen esimerkiksi muuttamalla sääolosuhteita tai ohjailemalla skenaariossa olevia muita ajoneuvoja. Tapahtumien lisäksi skenaarion tapahtumia ohjaa yleensä tekoäly, joka ohjaa muuta liikennettä tiettyjen sääntöjen mukaisesti. Pelkkien tapahtumien käyttäminen skenaarion ohjailuun on työlästä, sillä tapahtumia tarvitaan usein paljon, jotta skenaario vastaisi todellista tilannetta.

Skenaario on yhteydessä myös visualisointiin, sillä skenaario määrää millaisessa virtuaaliympäristössä harjoitus suoritetaan. Ajoneuvosimulaattorissa on tärkeää, että virtuaaliympäristöstä löytyy monipuolisesti paikkoja erilaisten liikennetilanteiden harjoitteluun. Myös skenaarion olosuhteet määrittellään skenaarioita suunniteltaessa.

Itse harjoituksen suorittamisen lisäksi skenaarioon sisältyy usein ennen harjoitusta esitettävä ohjeistus (briefing) ja harjoituksen läpikäynti jälkikäteen (debriefing). Ennako-ohjeistuksessa kerrotaan opiskelijalle mitä harjoituksessa on tarkoitus tehdä ja mitä tavoitteita siinä on.

Harjoituksen jälkeen suoritettava läpikäynti on tärkeä osa oppimisprosessia, sillä siinä opiskelijalle selviää, mitkä osat harjoituksesta sujuivat hyvin ja missä on parantamisen varaa. Läpikäynnissä opiskelijalle voi myös selvittää asioita, jotka harjoitusta suoritettaessa ovat jääneet huomaamatta. Läpikäynti suoritetaan yleensä yhdessä opettajan kanssa, sillä ihmisen kanssa keskustelu harjoituksesta on tehokkaampaa oppimisen kannalta, kuin pelkästään koneen tuottamien raporttien lukeminen. Läpikäynnin tukena on usein koneellisesti tuotettu yhteenveto suorituksesta. Lisäksi monissa opetussimulaattoreissa on mahdollista nauhoittaa suoritus ja katsoa se jälkikäteen uudelleen. Nauhoitusta katsottaessa suoritusta voi tarkastella eri kuvakulmista ja haluttaessa sen voi myös pysäyttää tärkeään paikkaan ja tarkastella tilannetta tarkemmin.

## **2.4. Laitteisto**

Simulaattoriohjelmistossa erityisesti fysiikan mallinnus ja 3D-visualisointi ovat tyypillisesti varsin raskaita osa-alueita ja vaativat tehokkaan tietokoneen. Nykyään tavallisten PC-tietokoneiden suorituskyky on kuitenkin kasvanut niin paljon, että simulaattoriohjelmistoa pystytään suorittamaan niillä. Mikäli yhden tietokoneen suorituskyky ei riitä koko simulaattoriohjelmiston ajamiseen, voidaan suoritusta hajauttaa useammalle tietokoneelle, jotka ovat yhteydessä toisiinsa verkon välityksellä. Yleensä tietokoneita tarvitaan useampia, jotta päästään riittävän hyvään päivitysnopeuteen. Hajautus voidaan tehdä esimerkiksi siten, että yksi kone laskee fysiikkaa, yksi tai useampi laskee visualisointia ja yksi hallinnoi muita koneita ja suorittaa loput toiminnot, kuten äänen toistamisen.

Tärkeimmät osat tietokoneissa suorituskyvyn kannalta ovat keskusprosessori ja näyttöohjain. Lisäksi keskusmuistia tulee olla riittävästi, nykyään yleinen määrä on 4 gigatavua. Prosessorit ovat nykyään moniydinprosessoreita, joten niillä pystytään suorittamaan tehokkaasti useita ohjelmia yhtäaikaaisesti. Tällä hetkellä suosituimmat prosessorit ovat kaksi- tai neliytimisiä, mutta tulevaisuudessa ytimien määrä tulee kasvamaan, sillä kehitteillä on jo prosessoreita, joissa on useita kymmeniä ytimiä. Moniydinprosessorin tehokas käyttö vaatii tukea myös simulaattoriohjelmistolta, joka tulee suunnitella siten, että se tukee monen säikeen käyttöä. Nykyään prosessorimarkkinoita hallitsevat Intel ja AMD, joilta kummaltakin on saatavissa tehokkaita moniydinprosessoreita, jotka soveltuvat hyvin myös simulaattorikäyttöön.

Näyttöohjaimet ovat kehittyneet nopeasti niin suorituskykynsä, kuin ominaisuuksiensa puolesta. Näyttöohjainmarkkinoille on tällä hetkellä kaksi suurta grafiikkapiirien valmistajaa: Nvidia ja AMD. Kummaltakin valmistajalta on saatavilla sekä pelikäyttöön suunnattuja näyttöohjaimia, että myös erikseen ammattikäyttöön tarkoitettuja malleja. Pelikäyttöön suunnatut näyttöohjaimet soveltuvat hyvin myös simulaattorin visualisointiin, sillä se muistuttaa varsin paljon pelien visualisointeja. Ammattikäyttöön tarkoitettujen mallien suunnattu lähinnä 3D-mallinnukseen ja CAD (Computer Aided Design) -suunnitteluun ja ne on optimoitu OpenGL-grafiikkarajapinnalle. Ammattikäyttöön suunnatut näyttöohjaimet ovat tietyissä käyttökohteissa hieman nopeampia, mutta myös huomattavasti kalliimpia, kuin pelinäyttöohjaimet.

Nvidian GeForce näyttöohjaimet ovat suosittuja pelikäytössä, mutta niitä voi hyvin käyttää myös simulaattorin visualisoinnissa. Nvidian ammattikäyttöön suunnattu näyttöohjainmallisto on nimeltään Quadro. Nvidian näyttöohjainmallistoon voi tutustua tarkemmin Nvidian kotisivuilla [Nvidia]. AMD:n Radeon näyttöohjaimet ovat myös suosittuja pelikäytössä, ja ne ovat suorituskyvyltään samaa tasoa, kuin GeForcet. AMD:n ammattikäyttöön suunnattu näyttöohjainmallisto on nimeltään FireGL. AMD:n näyttöohjainmallistoon voi tutustua tarkemmin AMD:n kotisivuilla [AMD]. Kaikista mainituista näyttöohjainmallistoista on saatavilla useita eri versioita, jotka eroavat suorituskyvyltään ja hinnaltaan. Kalliimmissa malleissa on myös usein enemmän muistia. Erityisesti pelinäyttöohjaimissa kehitys on nopeaa, ja niistä julkaistaan uusia malleja jopa kaksi kertaa vuodessa. Ammattikäyttöön suunnatuissa näyttöohjaimissa julkaisuja tulee harvemmin, ja niille luvataan myös pidempi tuki.

Näyttöohjaimissa rinnakkaista laskentaa hyödynnetään huomattavasti enemmän, kuin keskusprosessoreissa. Uusimmissa näyttöohjaimissa on satoja rinnakkaisia suoritusyksiköitä, joille laskentatyö voidaan jakaa. Esimerkiksi Nvidian GeForce GTX 285 näyttöohjaimessa on 240 rinnakkaista suoritinta [Nvidia].

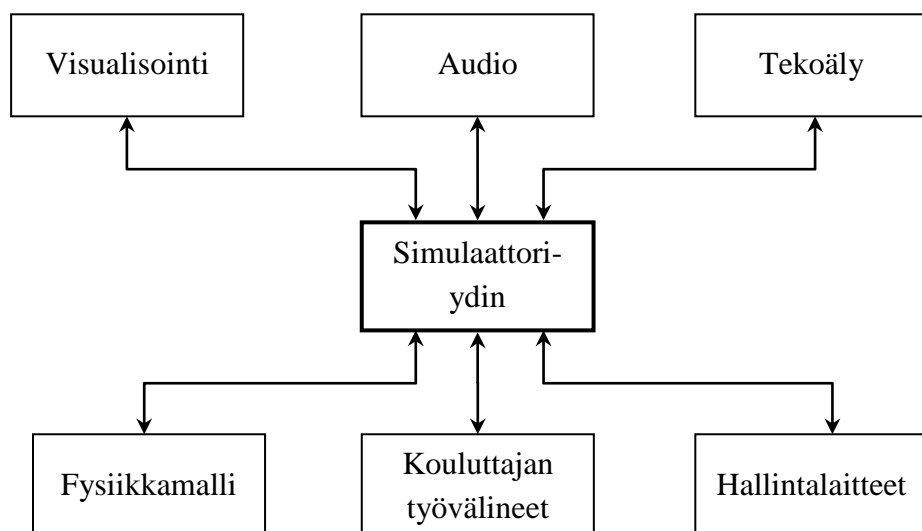
Näyttöohjaimissa rinnakkaista laskentaa voidaan hyödyntää tehokkaammin kuin keskussuorittimessa, koska grafiikan piirto jakaantuu paremmin useaan osaan, joita voidaan laskea irrallaan toisistaan. Grafiikan laskennassa käytettävä liukuhihna-rakenne helpottaa rinnakkaisten grafiikkasuorittimien käyttöä, koska liukuhihnan eri vaiheet voidaan laskea eri suorittimilla. Lisäksi kokonaisia liukuhihnoja, tai liukuhihnan osia

voi olla useita rinnakkain, joilla jokaisella on omat suorittimensa. [Akenine-Möller *et al.* 2008]

## 2.5. Ohjelmisto

Koulutussimulaattorit ovat tyypillisesti varsin laajoja ja monimutkaisia järjestelmiä, joten ohjelmistoarkkitehtuurin suunnittelu on tärkeässä asemassa niiden suunnittelussa. Ohjelmistoarkkitehtuurin rakenne ja ominaisuudet vaikuttavat suuresti koko simulaattorin toimivuuteen ja ylläpidettävyyteen. Kattava kuvaus koulutussimulaattorin arkkitehtuurin suunnittelusta löytyy Ville Isännäisen tekemästä diplomityöstä [Isännäinen 2010].

Simulaattoriohjelmisto on suuri kokonaisuus, ja se kannattaa jakaa osiin hallittavuuden ja päivitettävyyden parantamiseksi. Ohjelmiston osia kutsutaan usein moduuleiksi. Moduuli tarkoittaa tässä yhteydessä ohjelmiston osaa, jonka vastuulla on hoitaa tietty osa simulaattorin toiminnallisuudesta. Esimerkki simulaattorin moduulirakenteesta on esitetty kuvassa 2.6. Esitetyn kaltaista rakennetta on käytetty muun muassa Iowa ajoneuvosimulaattorissa [Kuhl & al. 1995]. Jokaisessa simulaattorissa on moduuli joka huolehtii simulaattorin ajamisesta ja tiedonsiirrosta, sekä hallinnoi muita moduuleja. Tätä moduulia kutsutaan simulaattoriyttimeksi. Ytimen lisäksi lähes kaikissa simulaattoreissa on fysiikan laskenta -moduuli ja tiedon esittämiseen liittyvät moduulit, kuten visualisointi ja audio. Edellä mainittujen moduulien lisäksi simulaattorissa voi olla muun muassa hallintalaitte- ja tekoälymoduulit. Koulutussimulaattoreissa on myös aina kouluttajan käyttöliittymä, jonka kautta kouluttaja voi ohjata ja seurata opiskelijan suoritusta. Ohjelmisto kannattaa suunnitella moduulipohjaiseksi, jolloin siihen voidaan helposti lisätä tai poistaa osia. Moduulipohjainen arkkitehtuuri mahdollistaa teoriassa myös tietyn moduulin vaihtamisen ilman, että se vaikuttaa muuhun ohjelmistoon. Tämä on hyvä ominaisuus, jos halutaan esimerkiksi siirtää käyttämään erilaista visualisointimoottoria.



**Kuva 2.6** Esimerkki simulaattoriohjelmiston moduulijaosta. Pohjautuu Iowa ajoneuvosimulaattorin ohjelmistoarkkitehtuuriin [Kuhl & al. 1995].

Visualisointimoduulin vastuulla on esittää simulaattorissa käytettävä maailma simulaattorin käyttäjälle. Simulaattoreissa käytettävä virtuaalimaailma on yleensä pyritty tekemään samankaltaiseksi, kuin oikea ympäristö, jossa simuloitavaa laitetta, tai asiaa käytetään. Ajoneuvosimulaattoreissa käytettävä maailma sisältää yleensä erikokoisia teitä niin kaupungissa, kuin sen ulkopuolellakin. Teiden lisäksi maailmassa on rakennuksia, kasvillisuutta, sekä muita asioita, jolla maisemasta saadaan aidon tuntuista. Lisäksi maailmassa on usein muita ajoneuvoja, tai muita liikkuvia kohteita.

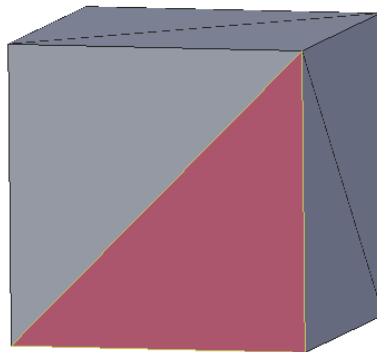
Ajoneuvosimulaattorissa fysiikkamoduuli laskee ajoneuvon fysiikkaa hallintalaitteilta saatujen arvojen perusteella. Lopputuloksena fysiikkamoduulilta saadaan muun muassa ajoneuvon paikka ja orientaatio. Tätä laskentaa suoritetaan reaaliaikaisesti, ja se vaatii runsaasti laskentatehoa. Visualisointimoduuli esittää ajoneuvon liikkeit käyttäjälle fysiikkamoduulilta saatujen tietojen perusteella. Visualisointimoduuli on siis riippuvainen fysiikkamoduulin toiminnasta, ja tästä syystä fysiikkamoduulin on toimittava vähintään yhtä suurella päivitystaajuudella, kuin visualisoinninkin. Jos fysiikanlaskenta toimii hitaammalla taajuudella kuin visualisointi, ei visualisoinnille tule riittävän usein uutta tietoa ajoneuvon paikasta. Tällöin visualisointi piirtää ajoneuvon samaan paikkaan usean päivitysjakson ajan, joka näkyy käyttäjälle kuvan nykimisenä. Tämä on monella tavalla haitallista, sillä se heikentää vauhdintuntua ja voi aiheuttaa myös simulaattorisairautta eli käyttäjälle aiheutuvaa pahoinvointia.

## 3. 3D-VISUALISOINNIN TEORIA

### 3.1. 3D-mallit

3D-visualisoinnilla tarkoitetaan tietokoneella esitettävää kolmiulotteista grafiikkaa. 3D-visualisoinnissa asiat esitetään 3D-mallien avulla, jotka voivat esittää reaali maailman esineitä tai olla täysin mielikuvituksellisia. 3D-malleja voidaan tehdä monella eri menetelmällä. Niitä voidaan esimerkiksi tehdä 3D-mallinnusohjelmalla käsityönä, skannata 3D-skannerilla tai rakentaa malli matemaattisten funktioiden perusteella. Mallinnustekniikan valintaan vaikuttaa mallin käyttötarkoitus, mutta 3D-visualisoinnissa yleisin tapa on käyttää 3D-mallinnusohjelmaa. Ammattikäytössä yleisin 3D-mallinnusohjelma on Autodeskin 3ds Max. Myös monia ilmaisia mallinnusohjelmia on saatavilla.

3D-mallin sisäinen rakenne voidaan tallentaa usealla eri tavalla. 3D-visualisoinnissa yleisimmäksi 3D-mallien rakenteeksi on vakiintunut polygonimalli, joka koostuu kolmiulotteisesta pisteverkosta. Pisteverkoston pisteet (vertice) muodostavat pisteiden välille viivoja (edge). Kolme pistettä, ja niiden väliset viivat, rajaavat polygonin (polygon). Polygoni voi olla myös useamman, kuin kolmen pisteen rajaama alue. Yleisimmät polygonit 3D-grafiikassa ovat kolme- tai neljäkulmaisia. Kuvassa 3.1 on esitetty yksinkertainen kuution 3D-malli, jonka jokainen sivu koostuu kahdesta polygonista. Kuvassa yksi polygoni on korostettu punaisella värillä.



**Kuva 3.1** Kuution 3D-malli, jonka jokainen sivu koostuu kahdesta polygonista. Kuvankaappaus Blender 3D-mallinnusohjelmasta.

Polygonin tärkein ominaisuus ja samalla myös rajoite on se, että polygoni on tasainen, eli sen kaikki pisteet ovat samassa tasossa. Tästä seuraa se, että polygonimallilla ei pysty esittämään täysin tarkasti monimutkaisempia muotoja sisältäviä esineitä. Esimerkiksi palloa ei pysty esittämään polygoneilla tarkasti, koska pallon pinta on kaareva, mutta polygonit ovat aina suoria. Tätä ongelmaa voidaan vähentää pienentämällä polygonin kokoa, jolloin mallin muodot saadaan näyttämään

pyöreämmiltä. Tästä seuraa kuitenkin polygonimäärän kasvu, joka heikentää suorituskykyä.

Mallin rungon lisäksi tarvitaan materiaalitieto, jonka avulla mallin pinnan ulkonäköä voidaan muokata. Materiaalissa voidaan määritellä muun muassa pinnan väri, valon heijastavuus ja läpinäkyvyys. Materiaalien lisäksi mallin pinnalla käytetään usein tekstuureja. Tekstuurit ovat kuvia, jotka kiinnitetään mallin pintaan sopivaan asentoon. Tekstuurien avulla saadaan mallista tehtyä yksityiskohtaisemman näköinen.

3D-mallien tarkkuus ja samalla myös polygonimäärät ovat kasvaneet voimakkaasti laitteiston kehittyessä yhä tehokkaammaksi. Mallien polygonimäärät vaihtelevat suuresti mallin käyttötarkoituksen mukaan, mutta esimerkiksi 2000 luvun alun tietokonepeleissä yksittäinen malli rakentui tyypillisesti sadoista tai maksimissaan muutamasta tuhannesta polygonista. Nykyisissä peleissä yksittäinen malli saattaa sisältää jopa satoja tuhansia polygoneja ja koko maisema miljoonia polygoneja. Tarkkuuden lisääntymisen ovat mahdollistaneet laitteiston suorituskyvyn kasvun lisäksi kehittyneet optimointimenetelmät.

## 3.2. Geometriset muunnokset

Geometriset muunnokset ovat perustyökaluja 3D-objektien liikuttamiseen virtuaalimaailmassa. Geometrisella muunnoksella tarkoitetaan sitä, että 3D-objektia muokataan matemaattisen funktion mukaan. Näitä matemaattisia funktioita ovat muun muassa 3D-objektin siirto, kierto ja skaalaus eli koon muuttaminen. Näiden kolmen funktion lisäksi on olemassa suuri määrä muitakin muunnoksia, mutta jo näitä kolmea yhdistelemällä pystyy 3D-objekteja liikuttamaan monipuolisesti.

Kaikki geometriset muunnokset toimivat samalla periaatteella: 3D-objektin jokaisen pisteen sijaintia muutetaan matemaattisen funktion määräämällä tavalla. Muunnosfunktiossa voidaan määritellä jokaiselle kolmesta akselista ( $x,y,z$ ) omat muunnosarvonsa, joka mahdollistaa objektin muokkaamisen eri verran eri akseleiden suhteen. Kolmiulotteisia objekteja käsittelevät muunnosfunktioit voidaan kuvata  $3 \times 3$  matriiseina, mutta yleensä ne esitetään  $4 \times 4$  matriiseina. Syynä tähän on se, että  $4 \times 4$  kokoiseen matriisiin saadaan mukaan myös tieto objektin paikan siirrosta. Näin on mahdollista esittää samassa matriisissa esim. rotaatio ja paikan siirto. [Watt 2000]

### 3.2.1. Siirto

Siirto (translaatio) muunnos siirtää kappaleen haluttuun paikkaan. Siirtoa käytetään hyvin yleisesti, sillä kappaleiden paikkaa täytyy muuttaa lähes kaikissa 3D-sovelluksissa. Muunnos esitetään translaatiomatriisina  $\mathbf{T}$ . Tämä matriisi siirtää kappaletta vektorin  $\mathbf{t} = (t_x, t_y, t_z)$  mukaisesti.  $\mathbf{T}$  on kuvattu kaavassa 3.1. [Akenine-Möller *et al.* 2008]

$$\mathbf{T}(t) = \mathbf{T}(t_x, t_y, t_z) = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.1)$$

### 3.2.2. Kierto

Rotaatio (rotation) muunnos kääntää kappaletta origon läpi kulkevan akselin ympäri. Rotaatiomuunnos esitetään rotaatiomatriiseina  $\mathbf{R}$ , jotka kuvataan kullekin akselille erikseen. Rotaatiomatriisit x, y ja z akselin suhteen on esitetty kaavoissa 3.2, 3.3, ja 3.4. [Akenine-Möller *et al.* 2008]

$$\mathbf{R}_x(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.2)$$

$$\mathbf{R}_y(\theta) = \begin{pmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.3)$$

$$\mathbf{R}_z(\theta) = \begin{pmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.4)$$

### 3.2.3. Skaalaus

Skaalaus muuttaa kappaleen kokoa siirtämällä kappaleen pisteitä kauemmaksi toisistaan x, y ja z – suunnissa. Skaalaus matriisi,  $\mathbf{S}(s) = \mathbf{S}(s_x, s_y, s_z)$ , on esitetty kaavassa 3.5. [Akenine-Möller *et al.* 2008]

$$\mathbf{S}(s) = \mathbf{S}(s_x, s_y, s_z) = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.5)$$

Mikäli  $s_x$ ,  $s_y$  tai  $s_z$  ovat arvoltaan alle 1, niin kappale pienenee kyseisen akselin suunnassa ja jos yli 1, niin kappale suurenee.

## 3.3. 3D-maailman rakenne

3D-maailma koostuu objekteista, jotka ovat 3D-malleja. Objektit sijaitsevat kolmiulotteisessa avaruudessa ja niille on määritelty geometrisiä ominaisuuksia, kuten paikka ja suunta. Näitä ominaisuuksia muuttamalla saadaan 3D-maailman objektit liikkumaan halutulla tavalla. Jotta objektien tiedot voidaan löytää, pitää ne tallentaa

tietokoneen muistiin jonkinlaiseen tietorakenteeseen. Ensimmäisissä 3D-sovelluksissa tämä tietorakenne saattoi olla esimerkiksi yksinkertainen lista johon objektit vain tallennettiin ilman tarkempaa logiikkaa. Yksinkertaisenkin tietorakenne oli toimiva ratkaisu, kun objekteja oli pieni määrä. Sen sijaan suuremmissa 3D-maailmoissa, joissa suorituskyky on tärkeä, oli keksittävä tehokkaampia ratkaisuja objektien hallintaan.

### 3.3.1. Scene graph tietorakenne

Scene graph on tietorakenne, joka on suunniteltu 3D-maailman tilan tallentamiseen. Scene graphit ovat nykyään hyvin yleisesti käytössä 3D-grafiikkaa sisältävissä sovelluksissa, kuten tietokonepeleissä ja graafisissa suunnitteluohjelmissa. Scene graphien tarkoitus on nopeuttaa maiseman piirtämistä järjestämällä objektit piirtämisen kannalta optimaaliseen järjestykseen.

Scene graph on hierarkkinen rakenne, eli siihen tallennetut alkiot on ryhmitelty jonkin tietyn ominaisuuden mukaan. Tämä ominaisuus voi olla esimerkiksi objektien looginen yhteys, paikka kolmiulotteisessa avaruudessa tai objektin pinnan materiaali. Scene graphien sisäinen rakenne vaihtelee hieman eri sovelluksissa käyttötarpeen mukaan, mutta yleisesti ne perustuvat puurakenteeseen, johon tiedot tallennetaan ryhmiteltynä tietyn ominaisuuden mukaan. Puuhun tallennettuja alkioita kutsutaan solmuiksi. Scene graphin tapauksessa solmu tarkoittaa objektia tai sen osaa. Solmuilla on tyypillisesti vain yksi vanhempi, mutta niillä voi olla useita lapsisolmuja. Koko puun isäsolmuja kutsutaan juureksi. Scene graphin käytön etuna on, että monia objekteja voidaan muokata yhdellä käskyllä, joka kohdistetaan niiden isäsolmuun. Isäsolmuun kohdistettu toimenpide välittyy automaattisesti kaikkiin sen lapsisolmuihin. [Akenine-Möller *et al.* 2008]

Perinteisiin tietorakenteisiin on tyypillisesti tallennettu vain objektien geometriset tiedot. Kolmiulotteisen maiseman esittämiseen tarvitaan kuitenkin paljon muutakin, kuin pelkästään geometria tietoa. Siihen tarvitaan muun muassa tietoa objektien animoinnista ja näkyvyydestä. Scene graphiin on mahdollista tallentaa myös näiden ominaisuuksien hallitsemiseen tarvittavat tiedot. [Akenine-Möller *et al.* 2008]

Kun scene graphiin lisätään paljon muita tietoja geometriatietojen lisäksi, on vaarana se, että rakenne muuttuu turhan raskaaksi ja suoritusajat kasvavat. Tässä tapauksessa kannattaa harkita tietojen ryhmittelyä eri tavalla, tai kokonaan toisen rakenteen tekemistä muille tiedoille.

## 3.4. 3D-piirron liukuhihna

3D-grafiikan piirto kuvataan usein liukuhihnan (pipeline) avulla. Liukuhihna nimitys tulee siitä, että jokaisen kuvan (frame) piirron aikana käydään tietyssä järjestyksessä läpi useita vaiheita. Grafiikan piirtoon tarvittavat tiedot etenevät liukuhihnalla vaiheelta toiselle ja jokaisessa vaiheessa tietoja käsitellään eri tavoin. Vaiheiden järjestys ei ole mitenkään yksikäsitteinen, vaan se riippuu muun muassa käytettävästä sovelluksesta ja näytönohjaimesta. Karkealla tasolla liukuhihnalla on kolme vaihetta, jotka suoritetaan

yleensä samassa järjestyksessä. Nämä vaiheet ovat sovellus-, geometria- ja rasterointivaihe. Liukuhinnarakenteen seurauksena koko järjestelmän suorituskyky määrytyy hitaimman vaiheen mukaan, koska seuraava vaihe ei voi alkaa, ennen kuin edellinen on saanut laskennan valmiiksi. Tästä syystä vaiheet kannattaa rakentaa siten, että kaikki suorittavat laskennan samassa ajassa. Vaiheet voidaan myös jakaa pienempiin palasiin, joita voidaan suorittaa rinnakkaisesti ja saavuttaa näin parempi suorituskyky. [Akenine-Möller *et al.* 2008]

### 3.4.1. Sovellusvaihe

Sovellusvaiheen tavoitteena on luoda 3D-malleista ja niiden oheistiedosta koostuva malli, joka kuvaa halutun virtuaalimaailman [Puhakka 2006]. Tämä malli lähetetään edelleen käsiteltäväksi seuraaviin vaiheisiin. Sovellusvaihe on ainoa vaihe, joka suoritetaan kokonaisuudessaan tietokoneen keskussuorittimella [Akenine-Möller *et al.* 2008]. Sovellusvaiheen toteutus on täysin ohjelmoijan päätettävissä ja sen ratkaisuilla voi vaikuttaa suuresti myös koko järjestelmän ruudunpäivitysnopeuteen. Sovellusvaiheessa tehdään mallin kokoamisen lisäksi myös optimointitoimenpiteitä, joilla voidaan saavuttaa huomattava suorituskyvyn parannus. Sovellusvaiheessa suoritettavia optimointitoimenpiteitä ovat näkökentän ulkopuolelle jäävien objektien rajausta ja mallien tarkkuustason valinta (LOD). Lisäksi sovellusvaiheessa suoritetaan tyypillisesti myös törmäystarkastelu. Kaikki visualisointikirjaston funktioiden avulla tehtävät toimenpiteet kuuluvat sovellusvaiheeseen.

### 3.4.2. Geometriavaihe

Geometriavaiheessa malleja käsitellään lähinnä niiden kulmapisteiden paikkoja muuttamalla, eli malleille tehdään erilaisia geometrisia muunnoksia, jotta ne saadaan asemoitua siten, että ne voidaan esittää näytöllä [Puhakka 2006]. Geometriavaihe suoritetaan yleensä näytönohjaimen suorittimella (GPU) [Akenine-Möller *et al.* 2008]. Aiemmin geometriavaiheen muunnostoimenpiteet ovat olleet kiinteästi ohjelmituna näytönohjaimeen, eikä niiden toteutukseen ole voinut vaikuttaa. Nykyaikaisissa näytönohjaimissa kiinteät vaiheet on korvattu vapaasti ohjelmitavalla verteksivarjostimella (vertex shader), jotka mahdollistavat kiinteitä operaatioita monipuolisemmat efektit [Puhakka 2006]. Verteksivarjostimilla voidaan nimensä mukaisesti vaikuttaa 3D-mallin pisteisiin (vertex). Verteksivarjostimella voidaan esimerkiksi luoda aaltoilevaa vettä siirtämällä mallin pisteitä aaltofunktion mukaisesti.

### 3.4.3. Rasterointivaihe

Liukuhinnan viimeisenä vaiheena on rasterointi, jonka tarkoituksena on muuntaa kolmiulotteisena tallennettu malli sellaiseen muotoon, että se voidaan esittää näyttölaitteilla, jotka pystyvät esittämään vain kaksiulotteista kuvaa. Rasterointivaiheen lopputuloksena 3D-malleilla esitetty maailma on muunnettu näytöllä näkyviksi pikseleiksi. Nykyaikaisissa näytönohjaimissa on ohjelmitavia pikselivarjostimia (pixel

shader), joilla voidaan vaikuttaa yksittäisen pikselin ulkonäköön [Puhakka 2006]. Pikselivarjostimella on esimerkiksi mahdollista luoda kuvaan sumuefekti lisäämällä jokaiseen pikseliin sumun väriä.

### 3.5. Grafiikkarajapinnat

Grafiikkarajapinta mahdollistaa näytönohjaimen toimintojen käytön ohjelmakoodista. Grafiikkarajapinnan perusajatuksena on tarjota ohjelmoijalle samat piirtokäskyt riippumatta käytetystä laitteistosta. Osa grafiikkarajapinnoista tarjoaa myös tuen useille eri käyttöjärjestelmille. Grafiikkarajapinnat voidaan jakaa matalan ja korkean tason rajapintoihin. Nykyään käytössä on yleisesti kaksi matalan tason grafiikkarajapintaa: OpenGL ja Direct3D. Korkean tason grafiikkarajapinnasta käytetään usein myös nimitystä grafiikkamoottori. Grafiikkamoottorit on rakennettu matalan tason grafiikkarajapinnan päälle ja ne tarjoavat paljon valmiita toiminnallisuuksia usein toistuvien asioiden tekemiseen.

#### 3.5.1. OpenGL

OpenGL (Open Graphic Library) on alun perin Silicon Graphics Inc nimisen yrityksen julkaisema grafiikkakirjasto. Ensimmäinen versio OpenGL:stä julkaistiin jo vuonna 1992 ja sen jälkeen siitä on julkaistu useita versioita, joissa on uusia ominaisuuksia. OpenGL:n uusin versio 3.0 julkaistiin vuonna 2008. OpenGL on yleisesti käytössä teollisuudessa ja lähes kaikki CAD suunnitteluohjelmat perustuvat OpenGL:ään. OpenGL suunniteltiin alun perin juuri teollisuuden tarpeisiin, mutta sitä käytetään jonkin verran myös pelien ja muiden 3D-ohjelmien toteutuksessa.

OpenGL toimii sekä Windowsissa, että Unixissa, joten se on hyvä vaihtoehto sellaisten sovellusten tekemiseen, joiden pitää toimia kummassakin käyttöjärjestelmässä. Windows Vista ei kuitenkaan suoraan tue natiivia OpenGL:ää, mutta Vistaan sisältyy tulkki, joka muuntaa OpenGL käskyt Direct3D:lle ja mahdollistaa näin myös OpenGL ohjelmien käytön. Tämä ratkaisu ei kaikissa tilanteissa ole kuitenkaan yhtä nopea ja vakaa, kuin natiivin OpenGL:n käyttö.

#### 3.5.2. Direct3D

Direct3D on Microsoftin kehittämä grafiikkakirjasto, joka on osa Microsoftin laajempaa DirectX kirjastoa. Ensimmäinen versio Direct3D:stä julkaistiin vuonna 1995 ja sen jälkeen siitä on julkaistu useita uusia versioita. Nykyinen versio on 11. Direct3D suunniteltiin alun perin juuri pelien grafiikkamoottoriksi ja siinä käytössä se toimiikin varsin hyvin. Ensimmäiset versiot Direct3D:stä olivat ohjelmoijien mielestä liian monimutkaisia ja hankalia käyttää, joten aluksi se ei saavuttanut suurta suosiota. Sen sijaan versiosta 7 eteenpäin se on ollut suosituin grafiikkarajapinta Windows pelien tekemiseen.

Rajoittava tekijä Direct3D:n käytössä on se, että se toimii vain Windows käyttöjärjestelmissä. Tämä on luonnollisesti suuri rajoite ja tästä syystä Direct3D:tä ei juurikaan käytetä suunnitteluohjelmissä tai muissa teollisuuden järjestelmissä. Tietokonepelien kohdalla tilanne on kuitenkin toinen, koska nykyään lähes kaikki PC pelit on suunniteltu juuri Windowsille ja tästä syystä myös lähes kaikissa peleissä käytetään Direct3D:tä. Direct3D:stä löytyy kuitenkin samat ominaisuudet kuin OpenGL:stä, joten se on toimiva kirjasto myös muunlaisiin sovelluksiin, kuten simulaattoreiden visualisointiin.

### 3.6. 3D-visualisointikirjastot

Matalan tason grafiikkarajapinnat, OpenGL ja Direct3D, tarjoavat kaikki tarvittavat työkalut 3D-sovellusten tekemiseen ja monet ohjelmat onkin tehty suoraan niillä. Ongelmana niiden käytössä on se, että yksinkertaisenkin ohjelman tekemiseen menee paljon aikaa, koska matalan tason rajapinnoista ei löydy valmiita funktioita monimutkaisempien asioiden tekemiseen, vaan kaikki täytyy ohjelmoida itse. Tästä syystä on kehitetty korkeamman tason grafiikkakirjastoja, jotka tarjoavat enemmän valmiita toimintoja. Korkean tason kirjastot on rakennettu matalan tason rajapintojen päälle, joten pohjimmiltaan niillä on samat ominaisuudet ja rajoitteet, kuin matalan tason kirjastoilla.

3D-visualisointikirjastoja on nykyään tarjolla runsaasti. Saatavilla on sekä ilmaisia että maksullisia kirjastoja. Ilmaisia avoimen lähdekoodin 3D visualisointikirjastoja ovat muun muassa Irrlicht, OGRE ja OpenSceneGraph. Suuri osa kirjastoista on tehty alun perin pelien kehittämistä varten, mutta useimmat niistä soveltuvat hyvin myös muiden sovellusten, kuten ajoneuvosimulaattoreiden tekemiseen. Kirjastoa valittaessa on tärkeää hahmottaa millaisia ominaisuuksia toteutettavassa sovelluksessa tarvitaan ja tehdä valinta niiden pohjalta. Korkean tason kirjastot ovat yleisesti helpompia käyttää, kuin matalan tason kirjastot, mutta niidenkin käytettävyydessä ja ominaisuuksissa on suuria eroavaisuuksia. Mitä korkeammalle tasolle kirjasto on tehty, eli mitä enemmän siinä on valmiita toimintoja, sitä nopeammin sillä saa tehtyä toimivan sovelluksen. Haittapuolena korkean tason kirjastoissa voi olla se, että ne eivät välttämättä ole riittävän joustavia erikoisempien toiminnallisuuksien toteuttamiseen. Esimerkiksi jonkun tietyn lajityypin pelien tekemiseen tarkoitettu grafiikkamoottori ei välttämättä mahdollista toisen lajityypin pelin kunnollista toteuttamista. Useimmat pelikäyttöön tarkoitettut grafiikkamoottorit on kuitenkin tehty yleisemmälle tasolle, jolloin ne mahdollistavat minkä tahansa 3D-sovelluksen tekemisen.

### 3.7. Optimointi

Valmiit grafiikkamoottorit sisältävät optimointitekniikoita, jotka nopeuttavat grafiikan piirtoa. Nämä optimointitekniikat ovat erityisen tärkeitä silloin, kun visualisoinnissa

esitettävä virtuaalimaailma on pinta-alaltaan suuri. Tällöin myös piirtoetäisyyksien täytyy olla pitkiä, jotta maisema näyttää hyvältä. Tämä tarkoittaa sitä, että näkyvissä saattaa olla yhtä aikaa satoja 3D-objekteja. Tällöin piirrettäväksi tulee helposti useita miljoonia kolmioita, ja näin suurista määristä edes nopeimmat näytönohjaimet eivät selviydy riittävällä päivitysnopeudella ilman optimointitekniikoiden käyttöä. Seuraavissa luvuissa on esitelty kaksi yleistä optimointimenetelmää: näkymättömien objektien rajausta ja level of detail.

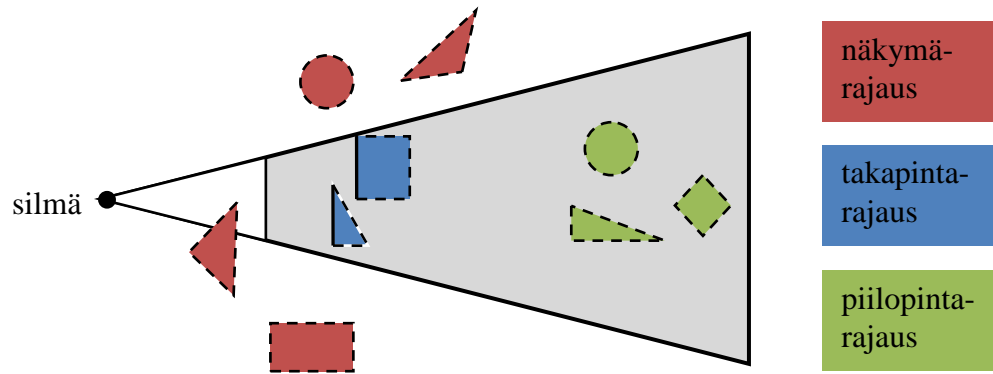
### 3.7.1. Näkymättömien objektien rajausta

Virtuaalinen ympäristö voi koostua tuhansista 3D-objekteista, joiden yhteenlaskettu kolmiomäärä on niin suuri, että kaikkia niitä ei pystytä piirtämään koko aikaa. Tästä syystä on kehitetty erilaisia rajausten menetelmiä, joiden avulla lopullisessa kuvassa näkymättömiin jäävät objektit ja objektien osat voidaan jättää piirtämättä. Rajausten menetelmiä on useita erilaisia riippuen siitä mitä osia halutaan rajata pois. Eri rajausten menetelmät on havainnollistettu kuvassa 3.2.

Objektin takapintojen rajausta (backface culling) rajaa pois objektin ne kolmiot, jotka eivät näy lopullisessa kuvassa, koska ne ovat objektin takapuolella. Esimerkiksi pallon muotoisesta objektista on keralla näkyvissä vain noin puolet kolmioista, joten tällä menetelmällä voidaan joidenkin objektien tapauksessa saada rajattua pois jopa puolet kolmiomäärästä. Kokonaisuudessa saavutettava hyöty ei kuitenkaan ole näin iso, koska läheskään kaikissa objekteissa ei ole paljoa taakse jääviä kolmioita. Esimerkiksi tasaisessa maastossa kaikki kolmiota ovat näkyvissä. Jos lopputuloksesta halutaan täysin aidon näköinen, pitää rajauksessa huomioida geometrisen laskennan lisäksi myös objektien mahdollinen läpinäkyvyys. Jos objektissa on läpinäkyviä osia, pitää niiden läpi näkyä objektin takana oleva maisema. Takapintojen rajausta on rajausten menetelmistä yksinkertaisin, sillä siinä suoritetaan laskentaa yksittäin kolmiotasolla. Takapintojen rajauksessa hyödynnetään polygonien normaalivektoreita, jotka kertovat polygonin suunnan avaruudessa. Normaalivektorin avulla voidaan laskea, onko polygoni näkyvissä. [Akenine-Möller *et al.* 2008]

Näkymärajausta (view frustum culling) rajataan pois ne objektit, jotka jäävät lopullisessa kuvassa näkyvän alueen ulkopuolelle. Näkökentän laajuus määrittää sen, kuinka paljon kuvassa näkyy objekteja. Näkökentän ulkopuolelle jäävät objektit rajataan pois. Näkökentällä on rajat myös etäisyysuunnassa, joten myös liian lähellä ja liian kaukana olevat objektit rajataan pois. Kuvassa 3.2 näkökentän alue on värjätty harmaalla. Suorituskyvyn kannalta tämä on tärkein rajausten menetelmä, sillä yleensä lopullisessa visualisoinnissa näkyvässä alueessa on mukana vain pieni osa koko maiseman objekteista. [Akenine-Möller *et al.* 2008]

Piilopintarajausta (occlusion culling) rajaa pois ne objektit jotka jäävät piiloon toisten objektien taakse. Tämä rajausten menetelmä vaatii paljon laskentaa, koska siinä pitää huomioida objektien vaikutus toisiinsa. Tästä syystä siitä ei välttämättä kaikissa tilanteissa saada kovin suurta suorituskykyä. [Akenine-Möller *et al.* 2008]



**Kuva 3.2 3D-visualisoinnin optimointiin käytettävät rajausmenetelmät. Kuvassa Rajatut pinnat on piirretty katkoviivalla. (Mukailtu [Akine-Möller et al. 2008]).**

Rajaus voidaan suorittaa missä tahansa osassa 3D-piirron liukuhihnaa. Suorituskyvyn

kannalta on sitä parempi, mitä aikaisemmassa vaiheessa liukuhihnaa rajaus tehdään, koska tällöin saadaan karsittua mahdollisimman paljon turhaa laskentaa myöhemmistä vaiheista. Mikäli rajaus suoritetaan jo sovellusvaiheessa, joka tehdään keskusprosessorilla, eivät rajatut objektit päädy ollenkaan näytönohjaimelle asti. Optimintapauksessa näytönohjaimelle menisivät vain lopulliseen näkymään kuuluvat objektit.

Rajausmenetelmät löytyvät kaikista moderneista grafiikkamoottoreista sisäänrakennettuna. Ohjelmoija voi yleensä päättää onko tietty menetelmä päällä vai ei ja mahdollisesti antaa sille muutamia parametreja, mutta muuten ne toimivat käytännössä automaattisesti. Joissain näytönohjaimissa osa rajausmenetelmistä on rakennettu suoraan rautatasolle, jolloin ne ovat hyvin nopeita.

### 3.7.2. LOD (level of detail)

Level of detail on optimointitekniikka, jossa 3D-mallista käytetään eri tarkkuudella tehtyjä versioita riippuen siitä, kuinka kaukaa sitä katsotaan. Läheltä katsottaessa käytetään tarkinta versiota, joka sisältää paljon kolmioita. Kauempaa katsottaessa 3D-objekti näkyy näytöllä pienempänä, jolloin siitä ei pysty erottamaan tarkempia yksityiskohtia. Tällöin voidaan käyttää yksinkertaisempia, vähemmistä kolmioista koostuvia, malleja ilman, että eroa huomaa. Yksinkertaisempien mallien käyttäminen voi tuoda suurenkin parannuksen suorituskykyyn, koska kolmioiden määrää saadaan vähennettyä huomattavasti. Esimerkiksi alkuperäisessä mallissa voi olla kolmioita 10 000 ja yksinkertaisimmassa LOD -tasossa vain 100. 3D-mallien geometrian lisäksi eri LOD -tasoilla voidaan käyttää myös erilaisia tekstuureita ja varjostimia (shader), jolloin saadaan vielä lisää suorituskykyä ja säästetään muistia.

Yleisesti LOD algoritmit koostuvat kolmesta osasta: luonti, valinta ja vaihtaminen. LOD:en luominen tarkoittaa käytännössä eri tarkkuuksisten versioiden tekemistä 3D-malleista. Tämä tapahtuu yleensä automaattisesti grafiikkamoottorin

toimesta siten, että alkuperäisestä mallista luodaan yksinkertaisempia malleja kolmioita vähentämällä. Haluttaessa mallien LOD:t voidaan tehdä myös käsin. Valinta tarkoittaa oikean LOD:n valitsemista tilanteen mukaan. Valinta perustuu tiettyihin kriteereihin, kuten siihen, kuinka suurena malli näkyy ruudulla ja kameran etäisyyteen mallista. Vaihtaminen tarkoittaa käytössä olevan LOD tason vaihtamista valinnan perusteella. [Akenine-Möller *et al.* 2008]

Ongelmana LOD:en käytössä on, että vaihdos eri LOD -tasojen välillä on usein havaittavissa lopullisessa visualisoinnissa välkkymisenä. Tähän ongelmaan on kehitetty useita ratkaisutapoja, jotka vähentävät välkkymistä. Eräs tapa on tehdä vaihdos siten, että vaihdoksessa käytetään hetken aikaa kumpaakin LOD:ia, ja vanha häivytetään pois muuttamalla se läpinäkyväksi [Akenine-Möller *et al.* 2008]. Välkkymistä voidaan vähentää myös tekemällä vaihto hieman eri kohdissa riippuen siitä ollaanko siirtymässä kauemmaksi vai lähemmäksi objektia [Puhakka 2006]. Myös LOD -tasojen lukumäärä vaikuttaa vaihdon sulavuuteen, sillä mitä enemmän tasoja on, sitä pienempi muutos tasojen välillä on, jolloin sitä ei myöskään huomaa niin selkeästi. Tasoja ei kuitenkaan kannata olla kovin paljoa, koska liian suuri tasojen määrä heikentää suorituskykyä. LOD -tasojen määrä vaihtelee tapauskohtaisesti, mutta yleinen määrä on kolmesta viiteen tasoa, jolloin käytössä on siis kolmesta viiteen eri tarkkuusversioita 3D-malleista.

Yleensä LOD:n valinta-algoritmi rakennetaan siten, että sillä saavutetaan mahdollisimman suuri suorituskykyetu, samalla kuitenkin pitäen visuaalinen ulkoasu mahdollisimman tarkkana. Algoritmi voidaan suunnitella myös siten, että se pyrkii pitämään ruudunpäivitysnopeuden vakiona. Tällaisesta algoritmista on hyötyä sellaisissa järjestelmissä, joissa on ehdottoman tärkeää, että päivitysnopeus on koko ajan sama. Ruudunpäivitysnopeuteen vaikuttaa moni muukin asia, joten pelkkä LOD -algoritmi ei yksinään pysty pitämään sitä vakiona, vaan myös muu visualisointi ja koko järjestelmä pitää suunnitella toimimaan vakionopeudella.

## 4. AJONEUVOSIMULAATTORIN VISUALISOINNIN ERITYISPIIRTEET

### 4.1. Suorituskykyvaatimukset

Simulaattorikäytössä on tärkeää, että visualisointi tapahtuu riittävän nopeasti ja tasaisesti ilman nykimistä tai muunlaista häiriötä. Visualisoinnin suorituskykyä mitataan ruudunpäivitysnopeudella (frames per second, fps), joka kertoo kuinka monta kertaa sekunnissa näytönohjain piirtää kuvan tietokoneen näytölle. Ruudunpäivitysnopeuteen vaikuttaa eniten näytönohjaimen suorituskyky, mutta myös moni muu asia, kuten ohjelmiston rakenne, grafiikan yksityiskohtaisuus, fysiikkamallin päivitysnopeus ja näyttölaitteen päivitysnopeus.

Samoin kuin videokuva, myös 3D-visualisoinnin grafiikka koostuu ruudulla esitettävistä yksittäisistä kuvista. Kun näitä kuvia vaihdetaan riittävän nopeasti, alkaa ihminen hahmottaa yksittäisten kuvien sijaan liikkuvaa kuvaa. On hieman henkilöstä kiinni kuinka nopeasti kuvia täytyy päivittää, jotta alkaa nähdä liikkuvaa kuvaa, mutta yleisesti tämä raja on noin 15 kuvaa sekunnissa. Tämä ei kuitenkaan vielä tarkoita, että kuva olisi tällä päivitysnopeudella sulavaa. Elokuville ja televisio-ohjelmissa ruudunpäivitysnopeus on 24 kuvaa sekunnissa ja tämä näyttää varsin sulavalta. Tietokoneella esitettävässä kuvassa 24 fps ei kuitenkaan näytä yhtä sulavalta. Suurin syy tähän on se, että elokuvissa yksittäiset kuvat ovat hieman sumeita (motion blur efekti) ja tästä syystä ihminen ei hahmota niiden vaihtumista niin selkeästi. Sen sijaan tietokoneella esitettävässä 3D-visualisoinnissa yksittäiset kuvat ovat täysin tarkkoja. Tietokonegrafiikassa yleisenä alarajana ruudunpäivitysnopeudelle pidetään 30 kuvaa sekunnissa. Ihmissilmä pystyy kuitenkin erottamaan paljon tätä korkeampia päivitystaajuuksia ja suositeltava taajuus on 60 kuvaa sekunnissa. Tällä taajuudella kuva näyttää useimpien ihmisten mielestä täysin sulavalta. Tämä taajuus on myös useimpien LCD näyttöjen suurin päivitysnopeus, joten niitä käytettäessä tätä suuremmasta ruudunpäivitysnopeudesta ei enää ole hyötyä, koska näyttö ei kuitenkaan päivitä sitä nopeammin. [Walrath 1999]

Kuvan piirtämisestä 3D-visualisoinnissa vastaa pääosin näytönohjain, joten se on myös suurin yksittäinen suorituskykyyn vaikuttava tekijä. Näytönohjainten nopea kehitystahti on mahdollistanut varsin yksityiskohtaisen kuvan esittämisen riittävällä päivitysnopeudella. Reaaliaikaisesti suoritettavassa visualisoinnissa kasvanutta suorituskykyä voidaan hyödyntää usealla eri tavalla: voidaan kasvattaa ruudunpäivitysnopeutta, käyttää tarkempaa resoluutiota, tehdä 3D-malleista tarkempia tai käyttää enemmän erikoisefektejä, kuten valoja ja varjoja. Nykyaikaisissa

tietokonepeleissä visualisoinnissa käytetään monenlaisia efektejä, jotka kuluttavat suuren osan näytönohjaimen resursseista. Sen sijaan nykyisissä simulaattoreissa ei ole yleensä käytössä lisäefektejä, vaan grafiikka on yksinkertaista ja selkeää. Simulaattoreissakin tietyillä efekteillä voisi parantaa visualisoinnin laatua. Esimerkiksi nykyistä paremmilla valaistusefekteillä voitaisiin ajosimulaattoreissa luoda huomattavasti realistisemmat pimeäajo-olosuhteet. Nykyiset näytönohjaimet mahdollistavat myös kahden eri kuvan piirtämisen yhdellä näytönohjaimella, joka mahdollistaa tietokoneiden määrän vähentämisen.

Simulaattoreissa on yleensä taustalla simulointimalli, jolla mallinnetaan reaali maailman ilmiötä. Tämän mallin tuottaman tiedot esitetään käyttäjälle mm. visualisoinnin avulla. Näin ollen myös simulointimallin nopeus vaikuttaa visualisoinnin sulavuuteen. Simulointimallin on siis toimittava vähintään yhtä suurella taajuudella, kuin visualisoinnin halutaan toimivan.

Ajoneuvosimulaattoreissa virtuaalinen harjoitteluala on yleensä pinta-alaltaan laaja, jotta siinä voidaan harjoitella monipuolisesti erilaisia ajotilanteita. Tämä tarkoittaa sitä, että myös visualisoinnin on kyettävä esittämään suuri alue, jossa maastoa näkyy kilometrien päähän ja näkyvissä voi olla jopa satoja 3D-objekteja. Tämä johtaa siihen, että näkyvillä oleva alue voi sisältää niin ison määrän polygoneja, että edes tehokkaimmat näytönohjaimet eivät sitä pysty sulavasti esittämään, ellei käytetä tehokkaita optimointimenetelmiä. Valmiisiin grafiikkamoottoreihin on useimmiten sisäänrakennettu varsin tehokkaat optimointimenetelmät, joten niiden avulla pystytään esittämään myös laajoja virtuaalialueita. Vaikka nämä optimointimenetelmät ovatkin sisäänrakennettuna grafiikkamoottoriin, niin se ei kuitenkaan tarkoita sitä, että suorituskyky olisi automaattisesti riittävän hyvä. Ohjelmoijan täytyy ymmärtää miten optimointimenetelmät toimivat, jotta niitä osaa käyttää tehokkaasti. Myös 3D-mallien rakenne ja tarkkuus pitää olla käyttötarkoitukseen sopiva.

## 4.2. Realistinen grafiikka

Simulaattoreissa on perinteisesti pyritty tekemään graafisesta ulkoasusta realistisen näköinen. Realistisuudella tarkoitetaan visualisoinnissa sitä, että visualisoitu näkymä pyritään tekemään mahdollisimman samanlaiseksi, kuin oikea maailma. Suurimpana rajoitteena realistisuudelle on ollut tietokoneiden laskentateho. Laskentatehon lisääntyessä graafinen ulkoasu on kehittynyt entistä paremmaksi ja nykyään visualisoinnissa pystytään luomaan varsin aidon näköisiä maisemia. Täysin fotorealistiseen ulkoasuun ei visualisoinnissa vieläkään päästä, eikä siihen välttämättä aina edes pyritä. Koulutussimulaattoreissa täysin realistista ulkoasua tärkeämpää on visualisoinnin selkeys, ja että sen avulla pystyy tekemään oppimisen kannalta tärkeät päätökset.

Eräs tärkeä tekijä visualisoinnin realistisuudessa on käytetty resoluutio. Resoluutio kertoo näytöllä esitettävien kuvapisteen määrän. Esimerkiksi 1280x1024

resoluutio tarkoittaa, että näytöllä on vaakatasossa 1280 kuvapistettä ja pystytasossa 1024 kuvapistettä. Suurempi resoluutio on yleensä parempi, koska siinä yksittäisen kuvapisteen koko on pienempi, joka mahdollistaa tarkempien yksityiskohtien esittämisen. Ajoneuvosimulaattoreissa esimerkiksi kaukana olevia liikennemerkkejä voi olla vaikea tunnistaa, jos käytetään liian pientä resoluutiota. Resoluution merkitys korostuu erityisesti suurilla näyttölaitteilla, koska tällöin yksittäinen kuvapiste on niin iso, että ihmissilmä pystyy sen erottamaan. Esimerkiksi jos 1280x1024 resoluution kuva esitetään kolme metriä leveällä pinnalla, niin kuvapisteen leveys on 2.3 mm. Käytettävän resoluution määrää lopulta se, kuinka suurta resoluutiota käytettävä näyttölaite tukee.

### **4.3. Lisätty todellisuus**

Lisätty todellisuus (Augmented Reality) yhdistää todelliseen maailmaan virtuaalisia objekteja. Tyypillinen lisättyä todellisuutta hyödyntävä sovellus on datalasit, joiden läpi käyttäjä näkee todellisen maailman, mutta näkymään lisätään virtuaalisia objekteja. Tällaista sovellusta voidaan käyttää esimerkiksi opastenuolten ja tekstien lisääminen todellisen maailman näkymään.

Simulaattorikäytössä lisätty todellisuus esiintyy hieman eri muodossa, koska simulaattoreissa ei yleensä näy todellista maailmaa, vaan sitä edustaa virtuaalimaisema, joka on tyypillisesti pyritty tekemään todellisen maailman kaltaiseksi. Tietokonepeleissä on jo pitkään käytetty lisättyä todellisuutta erilaisten lisätietojen kertomiseen pelaajalle. Esimerkiksi hahmon terveydentila kerrotaan monessa pelissä käyttäjälle piirtämällä näkymään palkki, jonka pituus kertoo terveydentilan. Vastaavanlaisia visualisointikeinoja voidaan hyödyntää myös simulaattorikäytössä. Myös ajoneuvosimulaattoreissa eräs käyttökohde lisätylle todellisuudelle on opastenuolten lisääminen näkymään. Lisätyn todellisuuden avulla voidaan myös pyrkiä tuomaan paremmin esille esimerkiksi liikennemerkkejä tai muita tärkeitä kohteita. Tietyn asian korostaminen voidaan toteuttaa esimerkiksi piirtämällä kohteen ympärille sitä korostava kehys.

### **4.4. Monikanavaisuus**

Ajoneuvosimulaattoreissa visualisoinnin esittämistavalla on suuri merkitys siihen, kuinka luonnolliseksi simulaattorin käyttäjä kokee ajokokemuksen. Tähän vaikuttavat muun muassa visualisoinnin päivitystaajuus, näkökentän laajuus ja visualisoinnin yksityiskohtaisuus.

Simulaattorin visualisoinnin esitystapaa ja laitteistoa mietittäessä kannattaa huomioida myös ihmisen näköaistin ominaisuudet ja rajoitteet. Ihmissilmän näkökenttä on laajuudeltaan vaakasuunnassa noin 150 astetta ja pystysuunnassa noin 120 astetta. Kahden silmän yhteenlaskettu näkökenttä vaakasuunnassa on hieman laajempi, noin 200 astetta. Päästä ja silmiä liikuttamalla kokonaisnäkökenttä on vielä huomattavasti

laajempi. Kahdella silmällä katsottaessa muodostuu näkökentän keskelle stereonäköalue, joka on vaakatasossa noin 30 astetta leveä. [Kalawsky 1993]

Simulaattorikäytössä vähintään stereonäön alueen on hyvä näkyä visualisoinnissa, mutta myös tätä laajempi visualisointialue lisää realismia, sillä ihminen tekee paljon havaintoja myös näkökentän reuna-alueilla. Simulaattoreissa laajempi visualisointialue auttaa myös lisäämään immersiota, eli tunnetta siitä, että on todella sisällä virtuaalimaailmassa.

Näkökentän laajuuden vaikutusta on tutkinut muun muassa Jerrold D. Prothero [Prothero 1998]. Tässä tutkimuksessa on todettu, että laajempi näkökenttä lisää useimpien henkilöiden kohdalla immersioita. Tässä on kuitenkin eroavaisuuksia ihmisten välillä, sillä kyseisessä tutkimuksessa kolmasosa ihmisistä koki kapeamman näkökentän toden tuntuiseemmaksi. Huonona puolena laajemmassa näkökentässä on se, että se aiheuttaa suuremmalla todennäköisyydellä simulaattorisairautta, kuin kapeampi näkökenttä.

Ajoneuvosimulaattoreissa tarvitaan ainakin yksi näkymä, jossa esitetään tyypillisesti ajoneuvon edessä oleva maailma. Tämä riittää ajamiseen sellaisissa tapauksissa, jossa ei ole tarvetta nähdä kovin laajalle alueelle sivusuunnassa. Ajoneuvosimulaattoreissa on kuitenkin usein tarve nähdä myös mitä ajoneuvon sivuilla tapahtuu. Tämä korostuu erityisesti harjoiteltaessa risteysajoa, jolloin on tarve havainnoida sivulta lähestyviä ajoneuvoja. Autopeleissä tämä ongelma on usein ratkaistu siten, että käyttäjä voi tiettyä näppäintä painamalla kääntää näkymää sivulle päin, jolloin hän näkee paremmin sivulla tapahtuvat asiat. Tämä ei kuitenkaan vastaa oikean maailman tilannetta ja on lisäksi hankala käyttää, koska käyttäjän on aina painettava nappia halutessaan katsoa sivulle. Lisäksi sivulle päin katsottaessa etunäkymä katoaa osittain hankaloittaen kokonaiskuvan hahmottamista. Realistiseen esitystapaan pyrkivissä simulaattoreissa onkin yleensä käytössä erilliset sivunäkymät. Sivunäkymät voidaan sijoittaa etunäkymän reunoille joko suoraan kulmaan, tai vinosti sivulle päin. Etu- ja sivunäkymien lisäksi ajoneuvosimulaattoreihin halutaan usein myös peilinäkymät, joista voi seurata ajoneuvon takana tapahtuvaa liikennettä.

## 4.5. Näyttölaitteet

Näyttölaitteiden valinta on tärkeässä roolissa tarkoituksenmukaisen visuaalisen ulkoasun luomisessa. Näyttölaitteiden valintaan vaikuttavat suuresti simulaattorin käyttötarkoitus, visualisoinnin haluttu kuvapinta-ala, sekä käytettävissä olevat tilat.

### 4.5.1. LCD näyttö

Yksinkertaisin ja myös halvin ratkaisu on käyttää yhtä LCD -näyttöä, jossa esitetään etunäkymä. Mikäli halutaan myös sivunäkymät, voidaan myös ne toteuttaa LCD -näyttöillä. Haittapuolena tässä ratkaisussa on se, että näyttöjen väliin jää aina tilaa johtuen näyttöjen reunuksista, jolloin kuva-alue ei ole yhtenäinen. Nykyään on tosin saatavilla myös hyvin ohuilla reunuksilla varustettuja näyttöjä, jolloin näkymien väliin

jää vain muutaman sentin levyinen alue. LCD -näyttöjen etuna on kompakti koko, sillä ne vievät tilaa syvyysuunnassa vain noin 10 senttiä. Nykyään on saatavilla kohtuuhintaisia LCD -näyttöjä kokoluokassa 20–52 tuumaa. Yleisin virkistystaajuus LCD -näyttöissä on 60Hz. Näyttöjen resoluutiot ovat kasvaneet viime aikoina ja nykyään suurempien näyttöjen resoluutio voi olla jopa 2560x1600. Yhdellä LCD -näytöllä voidaan koosta ja sijoittelusta riippuen saavuttaa 15–40 asteen näkökenttä vaakasuunnassa ja 10–25 asteen pystysuunnassa.

#### **4.5.2. Projektori**

Mikäli tarvitaan yli 52 tuuman kokoista kuva-alaa, on järkevää käyttää projektoria ja valkokangasta. Projektorien etuna on, että niillä voidaan tuottaa hyvinkin suurta kuvaa. Haittapuolena ne tarvitsevat paljon tilaa, koska suurta kuvaa esitettäessä myös projektorin heijastusmatka on pitkä. Projektorin heijastusmatka riippuu linssin kuvasuhteesta, joka kertoo projisointietäisyyden suhteessa saavutettuun kuvakokoon. Esimerkiksi 1:1 linssillä saadaan kahden metrin heijastusetäisyydeltä kaksi metriä leveä kuva, kun taas 2:1 linssillä tarvitaan samankokoisen kuvan tuottamiseen neljän metrin heijastusmatka.

Projektoria lisäksi tarvitaan valkokangas, johon kuva heijastetaan. Valkokankaat voidaan jakaa kahteen eri kategoriaan: etuprojisio- ja taustaprojisiovalkokankaat. Etuprojisiovalkokankaalle kuva heijastetaan samalta puolelta, josta sitä katsotaan. Taustaprojisiovalkokankaalle kuva heijastetaan takaa päin, joka mahdollistaa projektorin paremman sijoittelun. Etuprojisiota käytettäessä projektori pitää sijoittaa samalle alueelle, josta simulaattorin käyttäjä katselee valkokankaita. Monissa simulaattoreissa on suuret hallintalaitteet ja mahdollisesti myös oikeaa ajoneuvoa muistuttava kori, joka tekee videotykin sijoittamisesta hankalaa. Taustaprojisiota käytettäessä näitä ongelmia ei ole, koska projektori sijoitetaan kankaan taakse. Haittapuolena taustaprojisiossa on se, että kankaan takana täytyy olla riittävästi tilaa projektorille.

Tilan tarvetta voidaan vähentää käyttämällä peiliä, jonka kautta kuva heijastetaan kankaalle. Peilinä kannattaa käyttää pintaheijastuspeiliä, jossa heijastava kerros on peilin pinnassa. Tavallisessa peilissä heijastava kerros on lasin alla, joka voi aiheuttaa kuvaan pieniä haamukuvia.

#### **4.5.3. Virtuaalikypärä**

Virtuaalikypärä eroaa muista näyttölaitteista siten, että siinä näyttö on kiinnitettynä kypäriin, jonka käyttäjä asettaa päähänsä. Tästä seuraa se, että virtuaalikypärä liikkuu pään mukana.

Virtuaalikypärä on hyvä vaihtoehto näyttölaitteeksi, mikäli halutaan tukea täysin vapaata liikkumista virtuaalimaailmassa. Virtuaalikypärissä on yleensä mukana liikkeentunnistussensorit, jotka mahdollistavat pään asennon seuraamisen. Tämä

mahdollistaa sen, että virtuaalinäkymää voidaan käänellä ohjelmallisesti aitojen pään liikkeiden mukaan ja näin voidaan saavuttaa parempi immersio.

Virtuaalikyypärässä kummallekin silmälle on oma näyttö, ja ne ovat noin 5-10 senttimetrin etäisyydellä silmästä. Virtuaalikyypärän näytön näkökenttä on tyypillisesti noin 30-45 astetta vaakasuunnassa, joka on melko vähän verrattuna useasta näytöstä, tai projektorista koostuvaan visualisointiratkaisuun. Kapeahko näkökenttä voi aiheuttaa ongelmia joissain sovelluksissa, mutta toisaalta virtuaalikyypärän avulla voidaan liikkeentunnistusta hyödyntäen luoda täysi 360 asteen näkymä, jossa käyttäjä voi katsoa mihin suuntaan tahansa päätään kääntämällä. Aikaisemmin virtuaalikyypärät ovat olleet epämukavia käyttää pidempiä aikoja painonsa takia. Tässä on kuitenkin tapahtunut kehitystä viime vuosien aikana, ja nykyään kevyimmät virtuaalikyypärät painavat vain noin 600 grammaa. Simulaattorikäytössä ongelmia voi aiheuttaa se, että uuden käyttäjän pitää aina säätää virtuaalikyypärän säädöt omaan päähän sopiviksi. Lisäksi silmälasien käyttö virtuaalikyypärän kanssa on hankalaa.

#### **4.6. Simulaattorisairaus**

Simuloiduissa ympäristöissä käyttäjälle voi aiheutua simulaattorisairautta, joka voi ilmetä muun muassa päänsärkynä, pahoinvointina ja huimauksena. Simulaattorisairaus muistuttaa oireidensa osalta liikesairautta, jota monet ihmiset kärsivät esimerkiksi ollessaan laivassa, joka on kovassa merenkäynnissä. Yhtäläisyyksistä huolimatta simulaattorisairaus on kuitenkin eri asia, kuin liikesairaus, sillä sitä voi aiheutua myös ilman varsinaista liikettä, esimerkiksi vain katsomalla liikkuvaa kuvaa simuloidusta ympäristöstä. Simulaattorisairaus voi aiheuttaa myös jälkioireita, jotka voivat jatkua jopa useita tunteja varsinaisen simulaattorin käytön jälkeen. Jälkioireita ovat muun muassa huimaus, huojunta ja asennon epävarmuus. [Kolasinski 1995]

Simulaattorisairautta on tutkittu paljon jo vuosikymmenten ajan. Simulaattorisairauden tutkimista hankaloittaa eri ihmisten erilainen reagoiminen simuloituihin ympäristöihin. Toisilla ihmisillä ei tule minkäänlaisia oireita, vaikka he käyttäisivät simulaattoria pitkiäkin aikoja, kun taas toisilla saattaa seurata niin vakavia oireita, että he eivät pysty käyttämään simulaattoria lainkaan. Joka tapauksessa on selvää, että simulaattorikoulutuksen yleistyessä simulaattorisairaus on rajoite simulaattoreiden käytölle. Tästä syystä on tärkeää, että simulaattorit pyritään suunnittelemaan siten, että ne ovat mahdollisimman vähän simulaattorisairautta aiheuttavia.

Tutkimusten perusteella ei ole saatu täysin yksikäsitteisiä tuloksia simulaattorisairauden aiheuttajista, mutta suurimpana aiheuttajana pidetään yleisesti vihjekonfliktia, joka seuraa siitä, että ihmisen eri aistit saavat simuloidussa ympäristössä ristiriitaisia vihteitä. Esimerkiksi simulaattorin visualisoinnista aivot voivat saada näköaistin kautta tiedon, että ollaan liikkeessä, mutta tasapainoaisesti kertoo aivoille, että ollaankin paikallaan. Toinen syy simulaattorisairauteen voi olla se, että yhdenkin aistin

kautta saadut vihjeet ovat simuloitussa ympäristössä hieman erilaisia, kuin mihin on oikeassa ympäristössä totuttu. [Kolasinski 1995]

#### **4.6.1. Simulaattorisairauden torjuminen**

Simulaattorin rakenne vaikuttaa suuresti siihen kuinka paljon sen käyttäjillä esiintyy simulaattorisairautta. Vaikuttavia tekijöitä ovat muun muassa käytetty visualisointitapa, liikealustan käyttö ja liikkeiden voimakkuus. Myös simulaattorin käyttöympäristöllä on vaikutusta simulaattorisairauden yleisyyteen. Simulaattorikoulutukseen käytettävässä tilassa pitää olla hyvä ilmanvaihto ja valaistus pitää olla sellainen, että visualisoinnin ja muun valaistuksen välille ei muodostu liian suurta kontrastieroja, eli huone ei saisi olla täysin pimeä. Videoprojektoreiden käyttö tosin vaatii melko pimeän tilan.

Visualisoinnin katselukentän laajuus vaikuttaa suuresti simulaattorisairauden yleisyyteen. Laaja katselukenttä lisää tunnetta liikkeestä, ja aiheuttaa samalla todennäköisemmin myös simulaattorisairautta. Ohjelmiston osalta visualisoinnin tulee olla mahdollisimman sulavaa, eikä siinä saa esiintyä välkkymistä tai muuta häiriötä. Visualisoinnissa käytettävä yksityiskohtien määrä voi myös vaikuttaa simulaattorisairauteen, sillä enemmän yksityiskohtia sisältävän ympäristön on todettu aiheuttavan enemmän simulaattorisairautta.

Monissa simulaattoreissa on käytössä liikealusta, jonka avulla pyritään mallintamaan oikean laitteen liikkeitä. Liikealustan käyttö voi vähentää vihjekonfliktia, koska visualisoinnin lisäksi liikkeen voi aistia myös tasapainoelimillä. Toisaalta liikealustalla ei yleensä pystytä mallintamaan täysin oikeiden liikkeiden kaltaisia liikeratoja, jolloin vihjeet eivät vastaa todellisen maailman tilanteita. Liikealustan käyttö voi aiheuttaa simulaattorisairauden lisäksi myös liikesairautta. Liikealustaa käytettäessä kannattaa liikkeiden nopeus ja liikematka säätää maltillisiksi, koska nopeat liikkeet aiheuttavat todennäköisemmin simulaattori- tai liikesairautta.

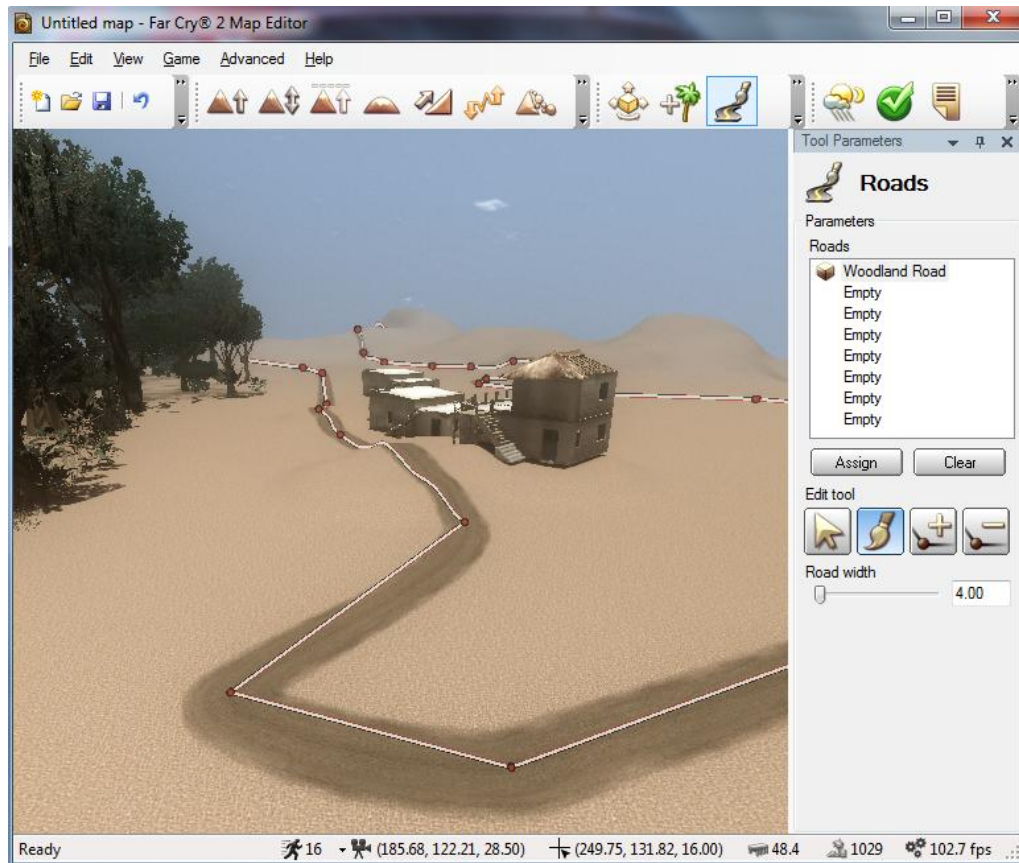
Eräs toimivaksi havaittu keino välttää simulaattorisairautta on totutella simulaattorin käyttöön pikku hiljaa, jolloin elimistö ehtii sopeutua simulaattorin toimintaan. Ensimmäisellä kerralla simulaattoria kannattaa käyttää vain vähän aikaa, esimerkiksi 5 minuuttia. Seuraavilla kerroilla aikaa voidaan pidentää. Ensimmäisellä kerralla kannattaa myös välttää sellaisten harjoitteiden suorittamista, jotka sisältävät paljon äkkinäisiä liikkeitä ja liikesuunnan muutoksia.

## 5. VIRTUAALIMAAILMAN LUOMINEN

Nykyaikaisissa simulaattoreissa on yleensä varsin laaja virtuaalimaailma, joka esitetään käyttäjälle 3D-visualisoinnin avulla. Opetuskäyttöön tehdyissä simulaattoreissa on tärkeää, että virtuaalimaailma tarjoaa monipuoliset mahdollisuudet oppimistehtävien suorittamiseen. Virtuaalimaailman luomisessa suurimmat ja eniten aikaa vievät vaiheet ovat tarkoitukseen sopivan maailman suunnitteleminen ja tarvittavien 3D-mallien tekeminen.

Virtuaalimaailman suunnitteleminen koulutussimulaattoriin on vaikea ja aikaa vievä tehtävä, koska maailman pitää olla toimiva kokonaisuus myös opetuksellisesta näkökulmasta katsottuna. Suunnitteluprosessia voidaan keventää tarjoamalla suunnittelijalle sopivat apuvälineet virtuaalimaailman tekemiseen. Yksi tapa nopeuttaa suunnittelua on käyttää pohjana jotain oikeasta maailmasta löytyvää paikkaa. Tällöin voidaan hyödyntää paikkatietoa (GIS, geographical information system), johon on tallennettu oikeassa maailmassa olevan paikan sijainti- ja ominaisuustiedot. Ongelmana tässä lähestymistavassa on se, että oikeasta maailmasta ei välttämättä löydy sellaista aluetta, joka mahdollistaisi kaikkien opetustehtävien tehokkaan opetuksen. Tähän ongelmaan voidaan hakea ratkaisua siten, että yhdistetään useita eri paikoista löytyviä todellisen maailman kohteita samaan virtuaalimaailmaan. Toinen ongelma on se, että riittävän tarkkojen kuvailutietojen hankkiminen oikean maailman kohteesta voi olla vaikeaa, sillä kaikkia maailman kohteita ei ole vielä mallinnettu paikkatiedoksi. Yksi paikkatietoa hyödyntävä lähestymistapa virtuaalimaailman luomiseen on esitetty Mayank Sharman tekemässä diplomityössä [Sharma 2009].

Toinen tapa nopeuttaa maailman suunnittelua on tehdä editori, jossa voidaan helposti luoda ja muokata virtuaalimaailman kohteita ja niiden ominaisuuksia. Editorin avulla on mahdollista suunnitella virtuaalimaailma alusta loppuun huomattavasti nopeammin verrattuna esimerkiksi 3D-mallinnusohjelmiin. Virtuaalimaailmaeditoreja on jonkin verran käytössä erilaisissa sovelluksissa. Monissa tietokonepeleissä on mahdollista suunnitella omia tehtäviä, ja usein niissä on mukana myös mahdollisuus muokata tai luoda kokonaan uusi virtuaalimaailma. Esimerkiksi Far Cry 2 pelissä on mukana monipuolinen editori (kuva 5.1), jolla pystyy muu muassa muokkaamaan maaston muotoja ja lisäämään rakennuksia, teitä ja kasvillisuutta. Editorin käytön lopputuloksena saadaan kuvailutiedosto, joka kattaa koko luodun virtuaalimaailman tiedot.



**Kuva 5.1** Kuvaruutukaappaus Far Cry 2 -pelin tehtäväeditorista

Sekä paikkatietoa että editoria käytettäessä maaston 3D-mallien tuottaminen voidaan ainakin osittain automatisoida käyttäen kuvaustietoja mallien luomiseen. Paikkatietoa käytettäessä 3D-mallit voidaan generoida suoraan paikkatiedon perusteella. Editoria käytettäessä 3D-mallit voidaan generoida editorin tuottaman kuvaustiedon pohjalta.

Ajoneuvosimulaattoriin soveltuva virtuaalimaailma voidaan sisältönsä puolesta jakaa kolmeen osaan: maasto, tiestö ja muut objektit. Maastolla tarkoitetaan tässä yhteydessä maan pintaa. Tiestö seuraa suurelta osin maaston pinnan muotoja, mutta varsinkin ajoneuvosimulaattoreissa tiet on mallinnettava tarkemmin kuin muu maasto. Muut objektit, kuten talot, puut ja liikennemerkkit sijoitetaan maastoon haluttuun paikkaan siten, että ne ovat samalla korkeudella kyseisen paikan maaston kanssa.

## 5.1. Paikkatieto

Paikkatiedolla tarkoitetaan sellaista tietoa, joka kertoo kohteen maantieteellisen sijainnin. Suomen julkisen hallinnon tietohallinnon neuvottelukunta määrittelee paikkatiedon seuraavasti: ”Paikkatieto on kokonaisuus, johon kuuluvat paikannettua todellisuuden kohdetta kuvaavat sijainti-, ominaisuus- ja yhteystiedot sekä näiden laatua koskevat tiedot.” Paikkatieto koostuu yleensä ominaisuus-, sijainti- ja yhteystiedosta (kuva 5.2).



**Kuva 5.2 Paikkatiedon osatekijät [Huttunen 1998]**

Ominaisuustiedot ovat kohdetta määritteleviä tietoja. Niiden avulla kuvaillaan kohteen yksilöiviä, paikantavia, ajoittavia tai kuvailevia tietoja. Ominaisuustietojen avulla paikkatietoon voidaan liittää sellaista tietoa, jota ei voi laskea pelkästään geometriatietojen perusteella. [Huttunen 1998]

Sijaintitieto tarkoittaa kohteen sijainnin kertovaa koordinaatti-, geometria- ja topologiatietoa. Koordinaattitieto kertoo kohteen sijainnin koordinaattien avulla, sekä käytetyn koordinaatistojärjestelmän ja koordinaattilukuarvojen epävarmuuden. Geometriatiedossa kerrotaan millä geometrisella yksilötyypillä todellisuuden kohde on kuvattu (esimerkiksi viiva, alue tai piste) ja kuvaillaan kohteen muoto valitun yksilötyypin avulla. Topologiatieto tarkoittaa geometrinen kohteiden välisiä suhteita kuvaavaa tietoa. Esimerkiksi katuverkostoa kuvattaessa katujen risteyskohdat muodostavat solmuja ja topologiatiedolla kerrotaan, mitkä kadut missäkin solmussa kohtaavat. [Huttunen 1998]

Yhteystieto sisältää todellisten kohteiden suhteita kuvaavia tietoja. Suhteiden kuvaamiseen käytetään kuitenkin yleensä topologiatietoja, jolloin yhteystietoja ei erikseen tallenneta. [Huttunen 1998]

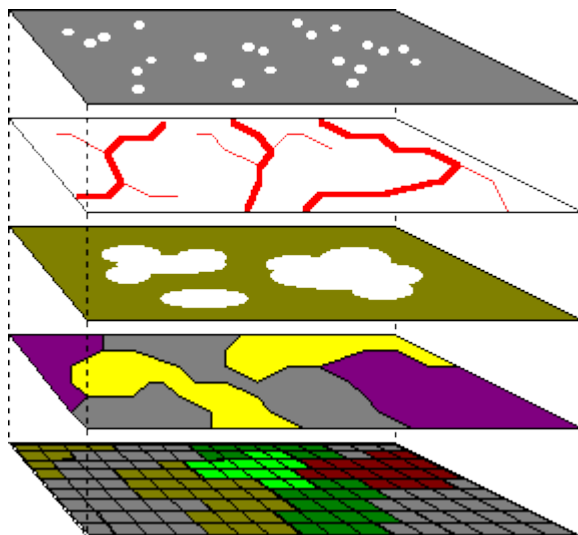
Paikkatietoa on jo pitkään hyödynnetty erilaisissa kartta- ja navigointisovelluksissa. Näihin tarkoituksiin riittää yleensä kaksiulotteinen paikkatieto, jossa ei ole kerrottu korkeustietoa. 2D paikkatiedolle onkin muodostunut yleisesti käytössä olevat tallennusformaatit ja sitä on myös hyvin saatavilla suurelta osalta maapalloa. Kolmas ulottuvuus, eli korkeustieto, on nykyään tulossa vahvasti mukaan paikkatiedon käyttöön, sillä se mahdollistaa huomattavasti monipuolisempien visualisointien luomisen muun muassa maaston muodoista, rakennuksista ja teistä.

Kolmiulotteisen paikkatiedon tallennukseen ei ole vielä muodostunut yhtä vahvoja standardeja, kuin 2D puolella, ja tästä syystä myös tiedon saatavuus ja

hyödynnettävyys on heikompaa. Standardin puutteesta johtuen 3D-paikkatiedon käsittelyyn käytettävät ohjelmat ja tiedostoformaatit vaihtelevat paljon eri projekteissa. CAD-ohjelmia käytetään melko paljon myös 3D-paikkatiedon luomiseen ja käsittelyyn. CAD-ohjelmat on tarkoitettu alun perin erilaisten suunnitelmien tekemiseen sekä kaksi-että kolmiulotteisena, joten periaatteessa ne soveltuvat myös paikkatiedon käsittelyyn. CAD-ohjelmin käytössä on kuitenkin muutamia rajoitteita, koska niitä ei ole suunniteltu paikkatiedon käsittelyyn. Suurin rajoite liittyy paikkatiedon osien välisten suhteiden ja lisätietojen säilyttämiseen, sillä CAD-ohjelmat eivät yleensä sisällä tällaisten tietojen tallentamiseen soveltuvia toimintoja. [Abdul-Rahman & Pilouk 2008]

Paikkatieto voi olla joko rasteri- tai vektorimuotoista. Rasterimuotoinen paikkatieto on käytännössä pikselimuotoinen kuva. Pikselikuvassa kuva on jaettu ruudukkoon ja jokaista ruutua vastaa yksi pikseli, joka kertoo ruudun värin. Kuvan avulla voidaan esittää esimerkiksi tavallinen karttalehti. Rasterimuotoinen paikkatieto soveltuu hyvin laajojen yhtenäisten alueiden esittämiseen. Esimerkiksi maaston pinnanmuodot voidaan hyvin esittää rasterikuvan avulla, jossa eri korkeudella olevat alueet on kuvattu eri väreillä.

Vektorimuotoinen paikkatieto koostuu pisteistä. Vektorimuotoinen paikkatieto voi sisältää esitettävän kohteen mukaan joko yksittäisiä pisteitä, pisteistä muodostuvia viivoja, tai pisteiden rajaamia alueita. Vektorimuodossa jokaisen pisteen koordinaatit on tallennettu numeerisina arvoina. Vektorimuotoiset paikkatiedot sisältävät usein tiedon paikasta vain kahdessa ulottuvuudessa, eli kohteen sijaintitieto korkeussuunnassa puuttuu. Tämä aiheuttaa ongelmia haluttaessa hyödyntää paikkatietoa 3D-mallinnetussa maailmassa, jossa myös korkeustieto on tiedettävä. Vektorimuotoinen paikkatieto sopii hyvin yksittäisten, toisistaan irrallaan olevien kohteiden esittämiseen.



**Kuva 5.3 Paikkatiedon geometriset yksilötyypit kerroksittain ylhäältä alas: piste, viiva, alue, aluejako ja ruudukko [Huttunen 1998]**

Paikkatieto jaetaan usein kerroksiin esitettävän tiedon mukaan (kuva 5.3). Esimerkiksi talot, tiet, järvet ja maaston muodot voivat olla omina kerroksinaan. Eri

kerrokset tallennetaan erillisiin tiedostoihin, jolloin voidaan helposti valita, mitä kerroksia halutaan esittää.

Virtuaalimaailman automaattisessa generoinnissa on useita erityyppisiä osaluoteita, joiden mallintamiseen kannattaa käyttää erilaisia lähtötietoja. Kuten aiemmin on mainittu, rasterikuva soveltuu hyvin maaston pinnanmuotojen kuvailemiseen. Teiden kuvaamiseen taas soveltuu paremmin vektorimuotoinen tieto.

## 5.2. Paikkatietoformaatit

Paikkatietoa hyödyntäviä sovelluksia on nykyään olemassa varsin paljon, ja moniin näistä sovelluksista on kehitetty myös oma paikkatietoformaatti. Eri formaatit eivät aina ole yhteensopivia keskenään, ja tämä hankaloittaa niiden käyttöä. Tähän lukuun on koottu vain muutamien yleisimpien paikkatietoformaattien ominaisuuksia. Kattava katsaus eri paikkatietoformaatteihin ja niiden ominaisuuksiin on esitetty Mayank Sharman diplomityössä [Sharma 2009].

Vektorimuotoisen paikkatiedon tallennuksessa yleisin formaatti on shapefile, joka on alun perin ESRI:n kehittämä, mutta nykyään se on käytössä varsin monessa paikkatietoa käsittelevässä ohjelmassa. Shapefile koostuu useasta eri tiedostopäätteen omaavasta tiedostosta. Seuraavassa listauksessa on kuvattu shapefilen pakolliset tiedostot [ESRI]:

- .shp – Shapefile päätiedosto, sisältää geometriatiedot tallennettavista kohteista
- .shx – Indeksitiedosto, nopeuttaa tietyn kohteen etsimistä shapefilen sisällä
- .dbf – Tietokantatiedosto, sisältää kohteiden attribuuttitiedot

Rasterimuotoisen paikkatiedotuksen tallennukseen käytetään yleisiä kuvaformaatteja, kuten tiff, png ja jpg. Itse kuvan lisäksi tarvitaan tiedot ainakin kuvassa käytetystä koordinaatistojärjestelmästä ja siitä, mihin koordinaatteihin se sijoittuu.

Korkeuskartta (heightmap) on erityistapaus rasterimuotoisesta paikkatiedosta. Siitä käytetään myös nimityksiä digital elevation model (DEM) ja digital terrain model (DTM). Korkeuskartan avulla kuvataan maaston pinnanmuotoja. Kuten kaikissa rasterikartoissa myös korkeuskartassa kuvan yksi pikseli vastaa tietyn kokoista neliötä maastossa. Korkeuskartassa pikselin värillä kerrotaan kyseisen kohdan korkeus. Korkeusarvo on keskiarvo kyseisen pikselin alueelle osuvalta maaston osalta. Korkeuskartat ovat yleensä harmaasävykuvia, joissa värejä käytetään siten, että valkoisella kuvataan korkeinta kohtaa ja mustalla matalinta.

Korkeuskartan tarkkuus riippuu siitä, kuinka suuren alueen yksi kuvan pikseli kattaa. Tyypillisesti pikselin koko vaihtelee välillä 10x10-1000x1000 metriä. Mitä pienempää aluetta yksi pikseli vastaa oikeassa maastossa, sitä tarkemmin korkeuskartalla pystytään esittämään pinnanmuodot. Toinen korkeuskartan tarkkuuteen vaikuttava tekijä on korkeustiedon tarkkuus, joka riippuu siitä, kuinka tarkasti maastoa kuvaava satelliitti on arvon saanut laskettua.

Korkeuskartoilla on muutamia rajoitteita, jotka estävät niiden käytön joissakin tapauksissa. Korkeuskartan rakenteesta johtuen sen avulla ei pystytä esittämään täysin pystysuoria pintoja. Tämä johtuu siitä, että korkeuskartta ei voi sisältää kahta päällekkäin olevaa pistettä. Samasta syystä myöskään maanalaisten rakenteiden, kuten esimerkiksi tunnelien esittäminen ei ole mahdollista korkeuskartan avulla.

### 5.3. Automaattinen 3D-mallien generointi

3D-mallien tekeminen käsityönä 3D-mallinnusohjelmaa käyttäen on hidasta, ja tästä syystä on alettu tutkia mahdollisuutta generoida 3D-malleja automaattisesti kuvailutiedon pohjalta. Käytettävän kuvailutiedon tulee sisältää tiedot maastosta ja kaikista objekteista, jotka maailmaan halutaan sijoittaa. Tämän tyyppisiä kuvailutietoja käytetään yleisesti paikkatiedon yhteydessä, joten niitä voidaan soveltaa myös virtuaalimaailman luonnin yhteydessä.

3D-mallin automaattisella generoinnilla tarkoitetaan prosessia, jossa kuvailutiedon pohjalta rakennetaan tietokoneellisesti ja automaattisesti 3D-polygonimalli. Jotta tämä olisi mahdollista, tulee käytettävän kuvailutiedon sisältää tieto mallinnettavan kohteen geometrisistä ominaisuuksista kolmessa ulottuvuudessa. Käytettävä aineisto täytyy muuntaa sellaiseen muotoon, että siitä voidaan rakentaa polygonimalli. Käytännössä aineiston pohjalta täytyy siis pystyä laskemaan 3D-mallin jokaisen pisteen koordinaatit kolmessa ulottuvuudessa. Tämä aiheuttaa rajoitteita käytettävän aineiston valinnassa, sillä kaikkien paikkatietoformaattien avulla muunnosprosessi 3D-malliksi ei ole mahdollista, koska ne eivät sisällä kaikkea tarvittavaa tietoa.

Muunnosprosessi paikkatiedosta 3D-polygonimalliksi voidaan toteuttaa joko käyttäen erillistä muunnosohjelmaa tai hyödyntämällä visualisointimoottorin tarjoamia ominaisuuksia mallien rakentamiseen.

### 5.4. Maaston luominen

Maaston luomisella tarkoitetaan maan pinnanmuotojen mallintamista. Pinnanmuotojen lisäksi maaston ominaisuuksia ovat pinnan koostumus, joka voi olla esimerkiksi ruoho tai hiekka. Erilaiset pinnat voidaan esittää käyttämällä erilaisia tekstuureja. Mikäli halutaan mallintaa oikeasta maailmasta löytyvää maastoa, voidaan lähdetietona käyttää paikkatietoaineistoa. Käytettävästä paikkatietoaineistosta on löydyttävä riittävällä tarkkuudella maaston muodot, eli käytännössä korkeustieto. Tämä rajaa käyttökelpoisia paikkatietoformaatteja, sillä läheskään kaikissa ei ole mukana maaston korkeustietoa sellaisessa muodossa, että sitä voitaisiin helposti hyödyntää automaattisessa maaston generoinnissa. Satelliitista tai lentokoneesta otetuista ilmakuvista on periaatteessa mahdollista hahmottaa karkeat maaston pinnanmuodot, mutta niiden tunnistaminen koneellisesti on varsin haastavaa ja aikaa vievää. Tästä syystä ilmakuvien käyttö ei ole

tämän opinnäytetyön puitteissa järkevä vaihtoehto automaattisen maaston generoinnin aineistoksi. Vaihtoehtoisiksi jää lähinnä pikselimuotoiset korkeuskartat ja vektorimuotoinen korkeuskäyriin perustuva aineisto.

Korkeuskartta on hyvä vaihtoehto maaston 3D-mallin luomiseen, koska siitä on suoraan saatavissa tieto 3D-mallin pisteiden koordinaateiksi. Leveys ja pituussuuntaiset koordinaatit saadaan yksittäisen pikselin sijainnista kuvassa. Korkeustieto saadaan pikselin väriarvon perusteella, joka kuvaa maaston korkeuden kyseisessä paikassa. Korkeuskartat tehdään yleensä satelliitista kuvatun aineiston perusteella. Korkeuskarttoja on saatavilla myös ilmaiseksi suuresta osasta maailmaa. Niitä on saatavilla useilla eri tarkkuustasoilla. Parhaiten ilmaista paikkatietoaineistoa on saatavilla USA:n alueelta.

Korkeuskarttoja ja myös muun muotoista paikkatietoa on saatavilla myös ilmaiseksi useista palveluista. Eräs koko maailman kattava palvelu on GeoComm ([www.geocomm.com](http://www.geocomm.com)), joka tarjoaa ilmaiseksi lähes koko maailman kattavat korkeuskartat ja muita paikkatietoaineistoja. Saatavilla olevien aineistojen tarkkuus ja formaatit vaihtelevat aluekohtaisesti, mutta esimerkiksi USA:n alueelta on saatavilla varsin tarkat 30x30 metrin ja jopa 10x10 metrin tarkkuudella olevat korkeuskartat. USA:n alueelta on muutenkin hyvin saatavilla paikkatietodataa, sillä U.S. Geological Survey (USGS) tarjoaa suuren osan aineistosta saataville myös ilmaiseksi.

Maaston sopiva tarkkuus riippuu suuresti simulaattorin käyttötarkoituksesta. Esimerkiksi opetuskäyttöön tehdyissä ajoneuvosimulaattoreissa maasto, eli teiden ulkopuolinen alue ei tyypillisesti ole mallinnettu kovin suurella tarkkuudella, koska sillä ei ole opetuksen kannalta suurta merkitystä. Simulaattoreissa maastoalueet voivat myös olla hyvin suuria, jolloin suurella tarkkuudella tehdyt maastot voivat aiheuttaa suorituskykyongelmia. Korkeuskartoissakin yleinen 10x10 tai 30x30 metrin tarkkuus on yleensä sopiva myös maaston polygonimallin pisteiden väliseksi etäisyydeksi.

Vektorikarttoihin perustuva maaston korkeusmalli on yleensä esitetty käyttäen samanlaisia korkeuskäyriä, joita käytetään perinteisissä paperikartoissa. Korkeuskäyrien ongelmana on se, että kahden käyrän välillä tapahtuu diskreetti hyppäys korkeudessa. Tämä aiheuttaa sen, että porrastus näkyy myös luotavassa 3D-mallissa, mikäli korkeuskäyrästä arvoja käytetään sellaisenaan mallin luomiseen. Käytännössä käyttökelpoisen maastomallin luomiseksi pitääkin kehittää sellainen algoritmi, joka tasoittaa korkeuskäyrien välisiä eroja maaston muodoissa.

## 5.5. Tiestön luominen

Tiestö on erityisesti ajoneuvosimulaattoreissa huomattavasti tärkeämmässä roolissa, kuin muu maasto, ja tästä syystä se täytyy myös mallintaa tarkemmin. Myös tiet on mahdollista mallintaa käyttäen samoja korkeuskarttoja, joilla maasto mallinnetaan, mutta niiden resoluutio ei yleensä riitä käyttökelpoisen tien luomiseen. Tästä syystä tiet tallennetaan usein erillisenä tiedostona ja eri muodossa, kuin muu maasto. Paikkatietoa voidaan hyödyntää myös tiestön mallinnuksessa käyttäen esimerkiksi

vektorimuodossa tallennettuja tiekuvauksia. Ongelmana on se, että nykyään suurin osa tiestön kuvailutiedoista on tallennettu vain 2D:nä, jolloin niistä puuttuu korkeustieto. Tätä ongelmaa voidaan kiertää siten, että käytetään mallintamiseen saatavilla olevia 2D malleja, ja lisätään korkeustieto toisessa formaatissa, kuten esimerkiksi maaston korkeuskarttojen avulla.

Vektorikuvausta käytettäessä tie muodostuu pisteistä, joka aiheuttaa myös ongelmia, koska pisteistä muodostetusta reitistä muodostuu kulmikas. Ajoneuvosimulaattorissa teiden mutkien pitäisi olla sulavan muotoisia, jotta ajokokemus on todellisuutta vastaava. Tästä syystä 3D-malleja ei voida muodostaa suoraan vektorimuotoisen kuvauksen perusteella. Ratkaisu tähän ongelmaan on muodostaa vektorikuvauksen perusteella reitistä bezier-käyrästä, tai jokin muu vastaava matemaattinen malli, joka on muodoltaan jatkuva. Bezier-käyrä on matemaattinen malli, joka muodostaa kahden tai useamman pisteen kautta kulkevan kaartuvan viivan [Akenine-Möller *et al.* 2008]. Bezier-käyrän avulla voidaan muodostaa tien varsinainen 3D-malli. Polygonimuotoinen 3D-malli koostuu myös pisteistä, joten käyrän jatkuvuus menetetään muodostettaessa polygonimalli. 3D-mallissa pisteitä voi kuitenkin olla hyvinkin taajassa, jolloin siitä saadaan riittävän sulavan muotoinen.

### 5.5.1. Tiesuunnitelmien hyödyntäminen

Nykyään uusien teiden suunnittelu tehdään suurelta osin tietokoneen avulla käyttäen erilaisia suunnittelu- ja mallinnusohjelmia. Näihin suunnitelmiin mallinnetaan tien muoto, kaistamerkinät ja risteysalueiden suunnittelu. Itse tien lisäksi suunnitelmissa on usein myös tien piennaralueet ja liikennemerkit. Suunnitelmat tehdään useimmiten CAD-pohjaisilla suunnitteluohjelmilla. Suunnitelmat sisältävät ainakin 2D:nä ylhäältä päin kuvatun suunnitelman ja nykyään monesti myös 3D-mallin.

Tiesuunnitelmissa on periaatteessa valmiina kaikki tarvittava tieto virtuaalimaailman tekemiseen, joten näiden suunnitelmien hyödyntäminen simulaattorin virtuaalimaailmaa tehdessä on houkutteleva vaihtoehto. Etuna näiden suunnitelmien käytöllä olisi se, että tiet on jo valmiiksi suunniteltu toimiviksi kokonaisuuksiksi, joissa on mukana myös liikennemerkit ja muut toimivalta liikenneympäristöltä edellytettävät ominaisuudet. Riippuen suunnitelmien muodosta, niiden perusteella voisi joko suoraan rakentaa 3D-mallin liikenneympäristöstä, tai lukea vain liikenne-elementtien sijainnit ja sijoittaa valmiit mallit niiden perusteella oikeisiin paikkoihin.

Suunnitelmien hyödyntämistä hankaloittaa kuitenkin muutama asia. Suurin ongelma on se, että Suomessa ei vielä ole käytössä yhtenäistä suositusta tai standardia teiden suunnitteluun, vaan eri hankkeiden suunnitteluprosessit eroavat suuresti toisistaan [Laakkonen 2009]. Jopa saman hankkeen sisällä saatetaan käyttää useita keskenään epäyhteensopivia ohjelmia ja tiedostoformaatteja [Laakkonen 2009]. Tämä tarkoittaa sitä, että eri hankkeiden suunnitelmat eivät ole keskenään yhteensopivia, joten niiden laajempi ohjelmallinen hyödyntäminen on teknisesti hyvin haastavaa. Toinen ongelma on se, että suunnitelmissa yleisimmin käytetyt CAD-formaatit ovat suljettuja formaatteja, eikä niiden sisäistä rakennetta ole julkistettu. Tällaisen suunnitelman

hyödyntäminen ohjelmallisesti on käytännössä mahdotonta. Ainakin Autocadin dwg- ja MicroStationin dgn -tiedostoformaattit ovat suunnittelussa yleisesti käytettyjä formaatteja, jotka kumpikin ovat suljettuja. Muutamat yritykset ovat kuitenkin onnistuneet selvittämään tiedostoformaattien rakenteen reverse-engineeringin avulla ja tarjoavat ohjelmistokirjastoja, joilla pystyy lukemaan kyseisiä tiedostoja. Esimerkiksi Open Design Alliance [Open Design] tarjoaa dwg - ja dgn – ohjelmistokirjastoja. Nämä ohjelmistokirjastot eivät kuitenkaan välttämättä tue kaikkia formaatin mahdollistamia ominaisuuksia.

Suomessa on tutkittu kannattaisiko infra-alalla siirtyä tuotetietomallin käyttöön. Tuotetietomalli tarkoittaa spesifikaatiota, jonka pohjalta voidaan toteuttaa tuotemallia käsitteleviä ohjelmistoja ja niiden välisiä rajapintoja. Tuotetietomalli tarkoittaa käytännössä laajaa järjestelmää, joka tarjoaisi yhteensopivat työvälineet infra-alan eri työvaiheisiin ja -tehtäviin. Infra2010 -kehittämishjelmassa on tutkittu muun muassa Quadri-mallin soveltuvuutta suomen infra-alalle. Quadri-malli on alun perin Norjassa kehitetty malli infra-suunnittelun yhtenäistämiseksi ja tiedonsiirron helpottamiseksi. Infra2010-hankkeessa on tutkittu myös LandXML –pohjaista tiedonsiirtoformaattia, jota voitaisiin hyödyntää suunnitelmien siirtämiseen yhteensopivassa muodossa eri suunnitteluvaiheiden ja eri projektien välillä. [Hyvärinen *et al.* 2006]

Tuotetietomallin käyttöönotto mahdollistaisi suunnitelmien helpomman hyödyntämisen myös tiesuunnittelun ulkopuolella. Esimerkiksi LandXML on avoin formaatti, joten sen hyödyntäminen olisi mahdollista myös simulaattorin virtuaalimaailman luomisessa.

## 5.6. Objektien luominen

Virtuaalimaailmaan tarvitaan maaston ja tiestön lisäksi suuri määrä erilaisia objekteja, jotta maailmasta saadaan luotua riittävän aidon tuntuinen. Näitä objekteja ovat muun muassa rakennukset, puut, liikennemerkkit ja ajoneuvot. Nämä objektit ovat usein muodoltaan niin monimutkaisia, että niiden 3D-malleja ei pysty tuottamaan automatisoidusti, vaan ne kannattaa tehdä käsityönä. Hyvänä puolena yhtä mallia voidaan monistaa ja käyttää useissa paikoissa virtuaalimaailmassa. Mallin ulkonäköä voi hieman muuttaa vaihtamalla tekstuuria tai skaalaamalla malli isommaksi tai pienemmäksi.

Maaston pinnalle sijoitettavat objektit voi periaatteessa mallintaa myös kiinteästi osaksi maastoa, mutta tämä tarkoittaisi sitä, että niiden uudelleen sijoittelu on hyvin vaikeaa. Tästä syystä ne on käytännöllisempää tehdä erillisiksi malleiksi, jotka sijoitetaan haluttuun paikkaan maastossa. Objektien sijoittaminen maastoon sopiville paikoille vaatii tarkkaa suunnittelua, mutta se pitää joka tapauksessa tehdä jossain vaiheessa virtuaalimaailman rakentamista. Sijoittelu helpottuu oleellisesti, mikäli käytössä on editori, jonka avulla objektit voi sijoittaa halutuille paikoille. Editorista tulee löytyä ominaisuudet ainakin objektin tyyppin valinnalle, objektin sijoittamiselle

maastoon ja objektin kääntelemiseksi eri asentoihin. Objektien tiedot kannattaa tallentaa XML-muotoiseen kuvaustiedostoon. Jokaisesta objektista tulee tallentaa ainakin seuraavat tiedot:

- objektin tunniste
- sijainti (x-, y- ja z-koordinaatit)
- asento (kierto x-, y- ja z-akselien suhteen)

Objektin pituus- ja leveysuuntainen sijainti suhteessa maastoon kerrotaan x- ja y-koordinaattien avulla. Objektin sijainti korkeussuunnassa kerrotaan z-koordinaatin avulla. Korkeustieto täytyy täsmätä maaston korkeuden kanssa, jotta objekti sijoittuisi maaston pinnalle. Tästä syystä korkeustieto on kätevä laskea maaston pinnasta ohjelmallisesti objektien lisäyksen yhteydessä, jolloin sitä ei tarvitse erikseen kertoa XML-kuvaustiedostossa.

## 5.7. Semanttisen tiedon hyödyntäminen

Toimivan koulutussimulaattorin tekemiseen tarvitaan paljon erilaista tietoa virtuaalimaailmasta ja sen ominaisuuksista. Tiedot kannattaa tallentaa semanttisena tietona, jolloin niitä on tehokasta käsitellä tietokoneella. Semanttisella tiedolla tarkoitetaan rakenteellisessa muodossa tallennettua tietoa, jota tietokone pystyy lukemaan ja käsittelemään [IJSC]. Ajoneuvosimulaattorin tapauksessa eniten tietoa tarvitaan tiestöstä ja muusta liikenneympäristöstä. Visualisointiin käytettävät 3D-mallit sisältävät lähinnä geometrista tietoa, joten niiden avulla pystytään laskemaan kohteiden etäisyyksiä ja muita geometrisia ominaisuuksia. Sen sijaan monia tärkeitä ominaisuuksia, kuten tien ajosuuntaa ei ole mahdollista päätellä pelkästään 3D-mallin tietojen perusteella. Muut tiedot voidaan tallentaa esimerkiksi XML muodossa erilliseen tiedostoon, josta niitä on helppo hyödyntää simulaattorin eri moduuleissa.

Tiestöstä tarvitaan esimerkiksi seuraavaa semanttista tietoa:

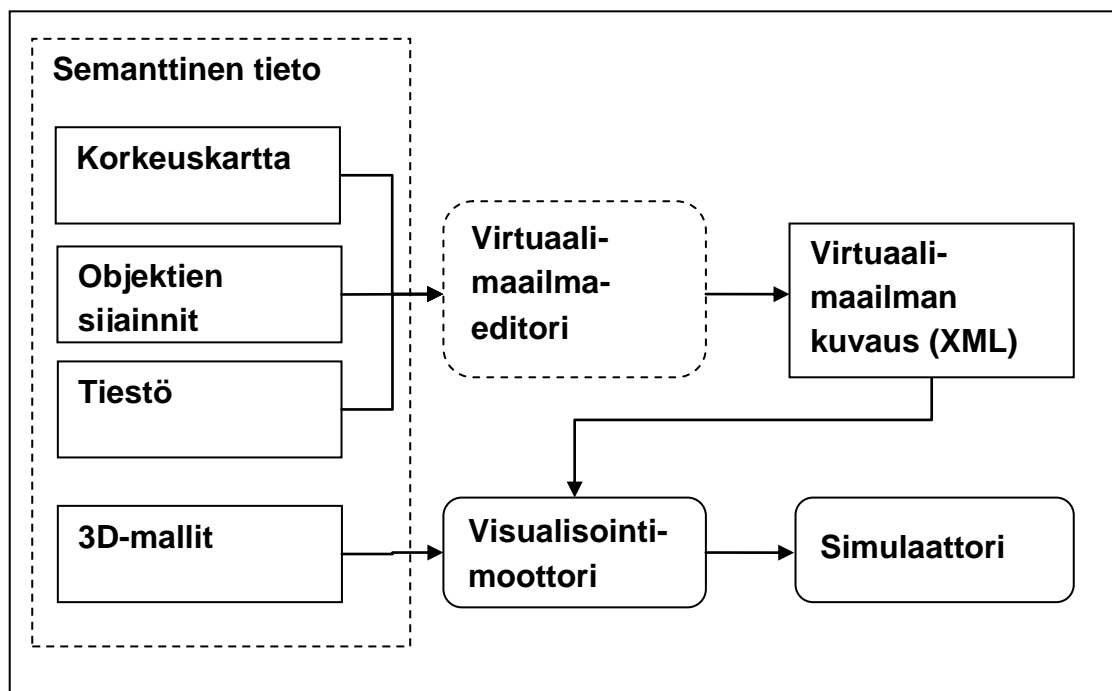
- reittiverkosto
- kaistojen määrä
- ajosuunta
- nopeusrajoitus
- liikennemerkit
  - o sijainti, vaikutusalue
- tien tyyppi
  - o moottoritie, valtatie, pyörätie...
- risteyksen tyyppi
  - o liikenneympyrä, tasa-arvoinen, eriarvoinen

Semanttista tietoa voidaan hyödyntää ajoneuvosimulaattorissa useissa eri ohjelmistomoduuleissa. Visualisoinnissa rikkaan semanttisen kuvauksen avulla on mahdollista toteuttaa monipuolisempia visualisointeja. Semanttinen tieto mahdollistaa monipuoliset lisätyn todellisuuden visualisoinnit, joita voidaan käyttää esimerkiksi reitin esittämiseen tai etäisyyden arvioinnin apuna.

## 6. KÄYTÄNNÖN TOTEUTUS

Tämän opinnäytetyön yhteydessä toteutettiin yksinkertainen ajoneuvosimulaattorin 3D-visualisointiohjelmisto ja virtuaalimaailman rakennusprosessi. Ohjelmiston toteuttamisessa hyödynnetään paikkatietoa ja 3D-mallien automaattista generointia.

Kuvassa 6.1 on esitetty virtuaalimaailman luominen prosessina. Kuvan vasemmassa reunassa ovat lähtötietoina käytettävät aineistot, joiden pohjalta virtuaalimaailma luodaan. Näiden aineistojen muokkaamiseen ja objektien sijoitteluun voitaisiin käyttää editoria, joka nopeuttaisi virtuaalimaailman luomista. Tämän työn yhteydessä editoria ei kuitenkaan toteuteta, vaan virtuaalimaailman kuvaava XML-tiedosto tuotetaan käsin. Tämä on mahdollista, koska virtuaalimaailma pidetään yksinkertaisena ja se sisältää vain muutamia objekteja. Laajempaa virtuaalimaailmaa tehtäessä editorista olisi suuri hyöty ja se toteuttamiseen kannattaisi käyttää aikaa. Varsinainen visualisointi tuotetaan visualisointimoottorin avulla, joka lukee virtuaalimaailman kuvaustiedostot ja luo niiden perusteella maaston, objektit ja tiestön.



Kuva 6.1 Virtuaalimaailman tuottaminen paikkatiedon ja muun semanttisen tiedon avulla

## 6.1. Hyödynnetyt ohjelmistot

3D-visualisoinnissa tärkein yksittäinen ohjelma on visualisointimoottori. Tämän työn visualisointimoottorina käytettiin OGRE:a (Object-Oriented Graphics Rendering Engine), joka on avoimen lähdekoodin visualisointikirjasto. OGRE käyttää scene graph tietorakennetta 3D-mallien rakenteen tallentamiseen ja soveltuu hyvin laajojenkin 3D-visualisointien tekemiseen. Se tarjoaa monipuoliset työkalut 3D-grafiikan käsittelyyn, jotka on myös hyvin dokumentoitu. OGRE on myös yhteensopivuudeltaan hyvä, sillä se tarjoaa OpenGL- ja Direct3D-rajapinnat ja se toimii Windowsissa, Linuxissa ja Macissa.

Paikkatietojen käsittelyyn käytettiin avoimen lähdekoodin MapWindow GIS ohjelmaa, joka osaa yleisen ESRI:n shape-formaatin lisäksi lukea suurta määrää muita paikkatietoformaatteja.

XML-tiedostojen lukemiseen käytettiin avoimen lähdekoodin tinyXML ohjelmistokirjastoa, joka tarjoaa tarvittavat perustoiminnot XML-tiedostojen käsittelyyn.

## 6.2. Tallennusformaatit

Visualisoinnissa käytettävien objektien paikat tallennetaan XML-tiedostoon, josta ne luetaan ja tallennetaan visualisointimoottorin sisäiseen scene graph tietorakenteeseen. Listauksen 6.2 esimerkissä on kahden puun ja yhden talon koordinaatit XML-muotoon tallennettuna. X- ja z-koordinaatit kuvaavat objektin leveys- ja pituus -suuntaista paikkaa. Objekteille voidaan määrittellä myös kiertokulma (yaw), joka kertoo kuinka paljon objektia tulee kääntää korkeussuuntaisen akselin ympäri.

```
<terrainobjects>
  <object>
    <type>tree</type>
    <x>50</x>
    <z>100</z>
  </object>
  <object>
    <type>tree</type>
    <x>70</x>
    <z>120</z>
  </object>
  <object>
    <type>building</type>
    <x>170</x>
```

```

<z>125</z>
<yaw>1.57</yaw>
</object>
</terrainobjects>

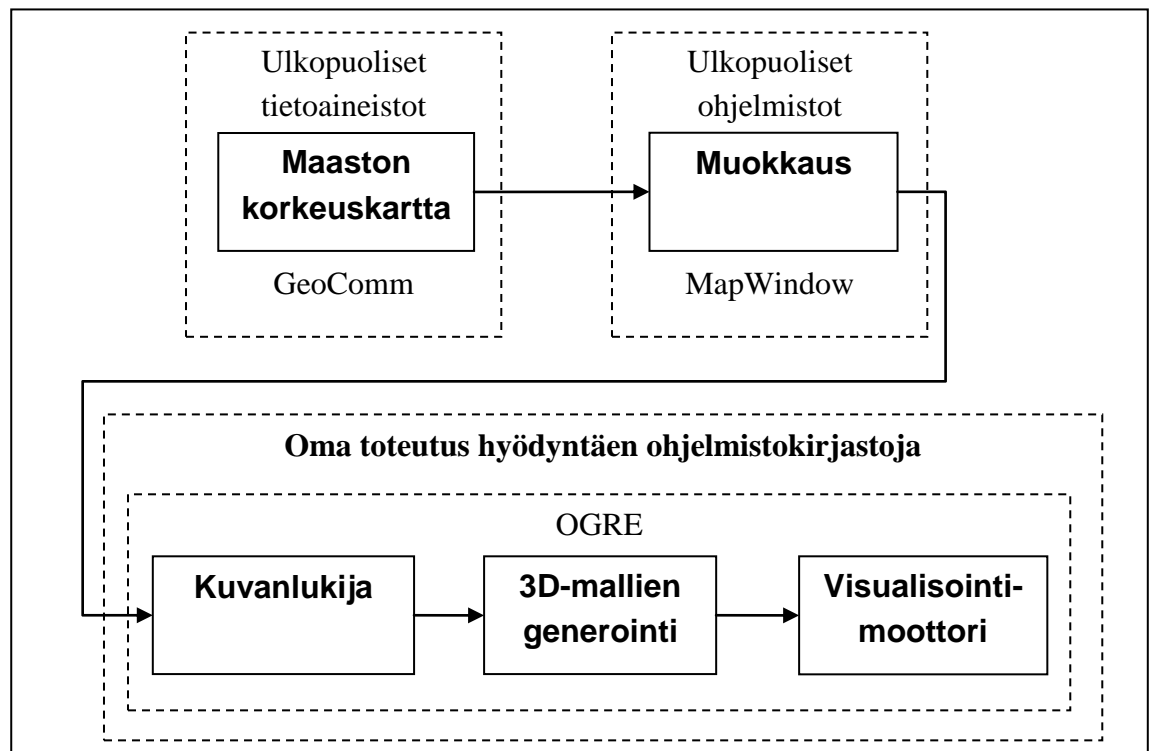
```

Listaus 6.2 Esimerkki objektien tiedot sisältävästä XML-tiedostosta

Valmiita 3D-malleja käyttävien objektien lisäksi myös tiet tallennetaan XML-tiedostona pistemuodossa. Tien jokaisesta pisteestä tallennetaan x- ja y-koordinaatit.

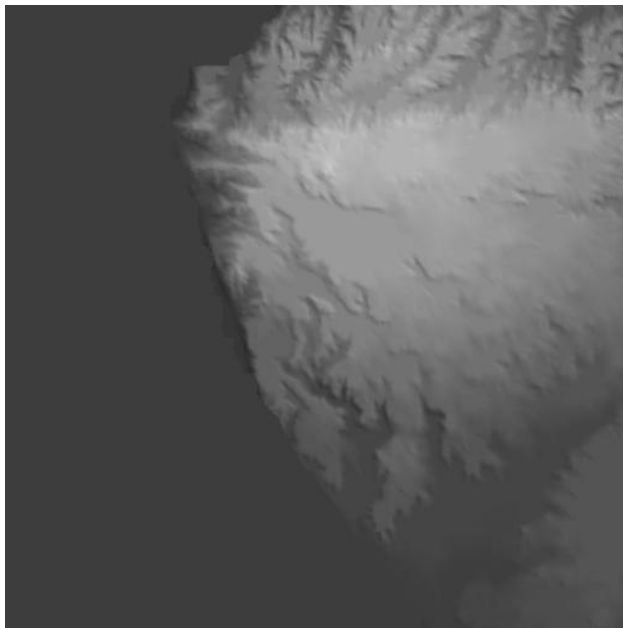
### 6.3. Maaston visualisointi

Maastomalli luodaan automaattisen generoinnin avulla. Maaston generoinnin päävaiheet on esitetty kuvassa 6.3. Maaston visualisoinnin pohjatietona käytetään korkeuskarttaa, joka on hankittu GeoComm-palvelusta. Käytetyt korkeuskartat ovat palvelussa nimellä Digital Elevation Models (DEM) – 24K. Näitä korkeuskarttoja löytyy suurelta osalta maailmaa. Yksi kartta kattaa maastossa 24x24km kokoisen alueen. Kartan resoluutio on 800x800 pikseliä, joten yksi pikseli vastaa 30x30 metrin aluetta maastossa. Korkeuskartat ovat ASCII Grid muodossa. Visualisointimoottorina käytettävä OGRE ei osaa suoraan lukea ascii muotoista kuvaa, joten korkeuskartta täytyy muuntaa OGRE:n ymmärtämään muotoon. Muuntaminen tehtiin MapWindow -ohjelmalla. Kun ASCII Grid tyyppisen korkeuskartan avaa MapWindowssa, se generoi siitä automaattisesti bmp-muotoisen kuvan, jota on helppo käsitellä muilla ohjelmilla.



Kuva 6.3 Maaston 3D-mallin generoiminen korkeuskartan perusteella

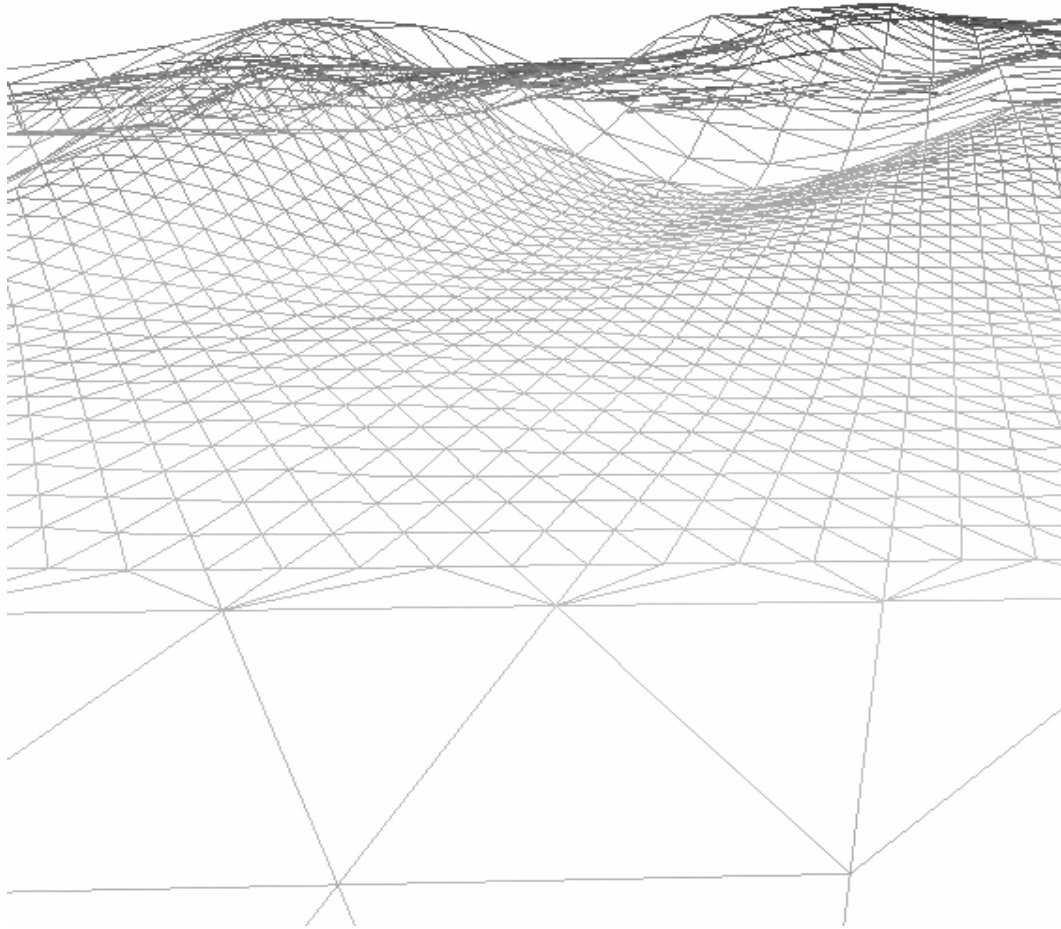
OGRE sisältää useita erilaisia Scene Managereita, jotka tallentavat 3D-maailman tilan ja rakenteen scene graph rakenteeseen. Scene Manager valitaan sen mukaan, millaista ympäristöä ollaan visualisoimassa. Laajan ulkomaisen visualisointiin sopiva vaihtoehto on Terrain Scene Manager, joka osaa suoraan lukea korkeuskartan ja luoda siitä 3D-mallin. Terrain Scene Manager vaatii korkeuskartan harmaasävyisenä png-kuvana. Kuvan täytyy myös olla neliön muotoinen ja sen sivun pituus pitää olla  $(2^n)+1$  (esimerkiksi 257 tai 513). MapWindow luoma bmp kuva täytyy muuntaa sopivaan muotoon. Muuntaminen voidaan tehdä millä tahansa kuvankäsittelyohjelmalla, joka osaa tallentaa kuvan png-muodossa. Esimerkki korkeuskartasta Terrain Scene Managerin hyväksymässä muodossa on esitetty kuvassa 6.4.



**Kuva 6.4 Esimerkki maastomallin generointiin käytettävästä korkeuskartasta harmaasävykuvana**

Terrain Scene Manager lukee korkeuskartan tiedot konfiguraatitiedoston perusteella. Tärkeimmät konfiguraatitiedostossa annettavat tiedot ovat korkeuskartan nimi, tekstuurin nimi, korkeuskartan koko ja korkeuskartan korkein mahdollinen arvo.

Terrain Scene Manager luo korkeuskartan perusteella automaattisesti maaston 3D-mallin. Periaate mallin luomisessa on yksinkertainen ja haluttaessa se olisi melko helppo tehdä myös omana toteutuksena. 3D-maastomalli luodaan siten, että luetaan korkeuskartan jokaisen pikselin arvo ja lisätään maaston 3D-malliin piste pikselin arvon mukaiselle korkeudelle. Pisteet sijoitetaan leveys- ja pituussuunnassa siten, että ne muodostavat yhtä ison alueen, kuin konfiguraatitiedostossa on määritelty korkeuskartan kooksi. Kuvassa 6.5 on esitetty Terrain Scene Managerilla luodun maaston 3D-mallin kolmioverkko.



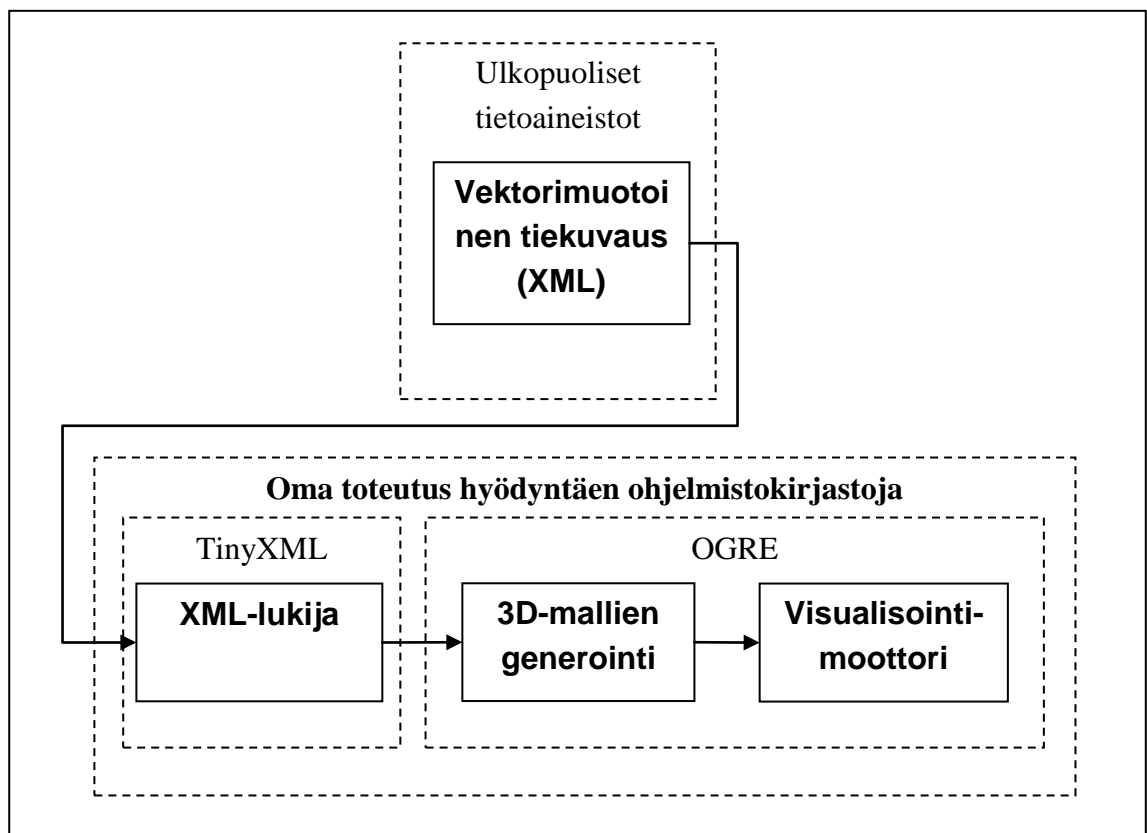
**Kuva 6.5 Terrain Scene Managerin avulla luotu maaston 3D-malli, jossa on käytössä useita LOD-tasoja.**

Valmiin toteutuksen käytöstä korkeuskartan lukemiseen on sekä hyötyjä, että haittoja. Suurena etuna valmiin toteutuksen käyttämisessä on se, että Terrain Scene Manager sisältää optimointeja, joiden avulla suurikin alue pystytään visualisoimaan tehokkaasti. Se jakaa maaston pienempiin palasiin, joista jokaisesta luodaan useita Level of Detail tasoja. Terrain Scene Manager osaa automaattisesti vaihtaa tietyn alueen mallin LOD-tasoa tarkempaan tai epätarkempaan katseluetaisyyden ja maaston pinnanmuotojen perusteella. Kuvasta 6.5 on nähtävissä LOD-tasojen käyttö optimointiin. Kuvan alareunassa näkyy suuria kolmioita, koska maasto on tällä alueella tasaista, eikä suuremmasta kolmiomäärästä olisi hyötyä. Kuvan keskiosassa maasto alkaa kohoamaan, jolloin tarvitaan tarkempaa esitystä. Kuvan yläosassa kolmiot ovat jälleen hieman isompia, koska tämä alue on niin kaukana katselupisteestä, että tarkkuutta voidaan pienentää katsojan sitä huomaamatta.

Haittapuolena Terrain Scene Managerin käytössä on se, että sen luomaa maastomallia ei pysty muokkaamaan luomisen jälkeen. Tämä voi aiheuttaa ongelmia tietyissä tilanteissa, sillä esimerkiksi maaston tasoittaminen tietyltä kohtaa ei ole mahdollista.

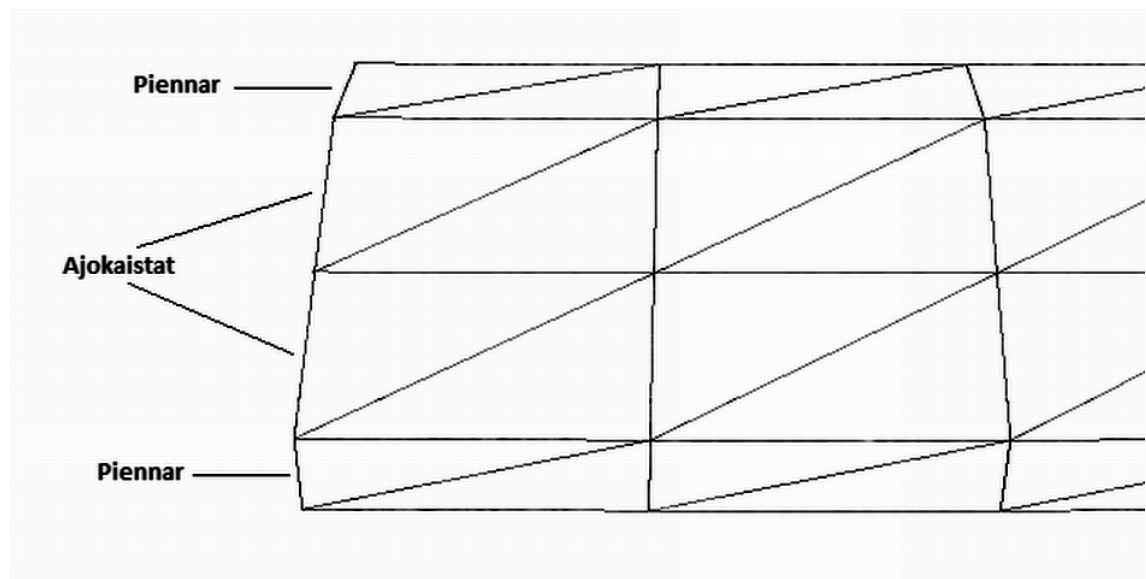
## 6.4. Tiestön visualisointi

Tiestöä kuvaavat 3D-mallit luodaan ajonaikaisesti tiestön kuvaavan tiedon pohjalta. Tiestön generointiprosessin vaiheet on esitetty kuvassa 6.6. Tiestön kuvaava tieto on esitetty pisteinä, jotka kuvaavat tien kulkureitin kaksiulotteisina vektoreina. Tien korkeustieto lasketaan polygonien luonnin yhteydessä maaston korkeuden mukaan. Tien pisteet on tallennettu XML-tiedostoon, josta ne luetaan XML-lukijan avulla. Lukija on ohjelmoitu hyödyntäen TinyXML-ohjelmistokirjaston tarjoamia toimintoja. Pisteiden avulla esitetyn tiestön ongelma on se, että mutkissa pisteitä pitää olla taajassa, tai muuten mutkasta tulee epätasainen ja kulmikas. Tämä ongelma on ratkaistu siten, että kuvailutiedossa kerrottujen pisteiden avulla luodaan aluksi käyrät jotka kuvaavat tien kulkureitin pelkkiä pisteitä tarkemmin. Nämä käyrät mahdollistavat sen, että kuvailutiedossa pisteitä voi olla harvassa, mutta tien muodoista saadaan silti sulavat. OGRE tarjoaa useita valmiita käyrätyyppejä. Teiden muodostamiseen valittiin SimpleSpline niminen käyrämuoto. SimpleSpline käyrä muodostetaan antamalla sille pisteet, jonka kautta käyrän halutaan kulkevan. Pisteet voidaan antaa joko kaksi- tai kolmiulotteisessa muodossa. Tässä tapauksessa halutaan luoda vain kaksiulotteisia käyriä, joten korkeuskoordinaatit asetetaan nollassi.



Kuva 6.6 Tiestön 3D-mallin generoiminen vektorimuotoisen aineiston perusteella

Varsinainen tien 3D-malli muodostetaan luomalla polygoneja tien kuvaavan käyrän muodon mukaisesti. Kuten luvussa 3 on kerrottu, polygonit muodostuvat kolmesta pisteestä. Pisteiden välimatka vaikuttaa mallin tarkkuuteen, joten sulavan muotoisessa tiessä niitä pitää olla melko taajassa. Tien pituussuunnassa pisteiden etäisyydeksi valittiin viisi metriä. Tiestön kuvaavassa kuvailutiedossa tie muodostuu yksittäisistä pisteistä, jotka eivät kuvaile mitenkään tien leveyttä, eikä leveyssuuntaista muotoa. Nämä tiedot pitää hankkia muualta, tai päättää ne itse. Tien leveys ja tien leveyssuuntainen pisteiden määrä on tallennettu ohjelmakoodissa muuttujiin, joiden arvoja voidaan muuttaa. Näin mahdollistetaan useiden erikokoisten teiden luominen. Esimerkkivisualisoinnissa tien leveydeksi on valittu 7,5 metriä ja tie muodostuu leveyssuunnassa viidestä pisteestä. Pisteet on asemoitu siten, että keskimmainen piste kuvaa tien keskikohdan ja sen viereiset pisteet ovat ajokaistojen reunoilla. Reunimmaisista pisteistä kuvaavat pientareiden reunoja. Tien 3D-mallin rakenne on esitetty kuvassa 6.7.



**Kuva 6.7 Tien rakenne. Kuvassa esitetään, miten tie jakaantuu ajokaista- ja piennar-alueisiin. Kuvasta selviää myös, miten tie rakentuu pisteistä, niiden välisistä viivoista ja viivojen rajaamista kolmioiden muotoisista polygoneista.**

Tiestö muodostetaan seuraavan algoritmin mukaisesti:

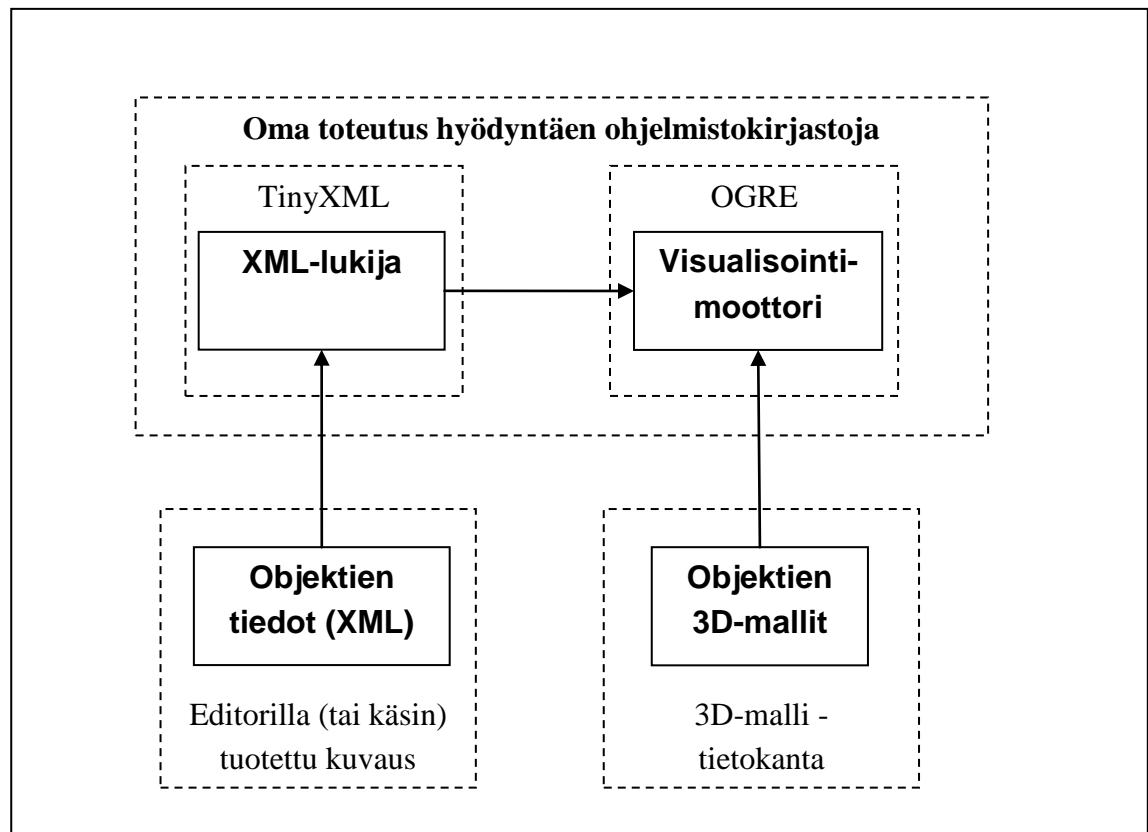
1. Hae tien keskipisteen sijainti tiestön kuvaavalta käyrältä
2. Luo tien keskipiste ja muut leveyssuuntaiset pisteet
3. Laske pisteiden korkeus maaston mukaan
4. Luo pisteiden mukaiset polygonit
5. Toista jokaiselle tien pituussuuntaiselle pisteelle

Pisteiden leveys- ja pituussuuntainen sijainti saadaan kysymällä tien kuvaavalta käyrältä sijaintia halutulta kohtaa käyrää. Käyrän palauttama piste kuvaa tien

keskikohdan pistettä. Loput tien leveysuunnassa olevat pisteet luodaan siten, että tien leveydeksi muodostuu 7,5 metriä. Pisteiden korkeus lasketaan maaston pinnanmuotojen mukaan. OGRE:ssa maaston korkeus voidaan selvittää RaySceneQueryn avulla. RaySceneQuery lähettää säteen tietyistä pisteistä haluttuun suuntaan ja kertoo kun se törmää maastoon, tai johonkin muuhun objektiin. RaySceneQuery osaa tämän jälkeen kertoa minkä tyyppiseen objektiin se törmäsi, sekä törmäyspisteen koordinaatit, joista saadaan maaston pinnan korkeus. Tien ajokaistojen alue luodaan esimerkiksi leveysuunnassa tasaiseksi ja tien korkeudeksi valitaan sen leveysuuntaisen pisteen korkeus, joka on korkeimmalla. Tiehen voitaisiin haluttaessa luoda kallistuksia asettamalla pisteet eri tasoon. Piennaralueiden reunapisteiden korkeudeksi asetetaan maaston korkeus näissä pisteissä, jolloin tien pientareet mukailevat maaston muotoja.

## 6.5. Objektien visualisointi

Objekteilla tarkoitetaan tässä yhteydessä kaikkia muita 3D-objekteja maaston ja tiestön lisäksi. Näitä voivat olla esimerkiksi erilaiset rakennukset, puut ja ajoneuvot. Objekteille on olemassa valmiit 3D-mallit, jotka asetetaan oikeaan paikkaan virtuaalimaailmassa kuvaustiedoston perusteella. Objektien luomisen vaiheet on esitetty kuvassa 6.8. Objektien paikat ja muut ominaisuustiedot luetaan XML-tiedostosta TinyXML-lukijan avulla. Objektien tietojen lukemista varten on ohjelmoitu lukija, joka lukee jokaisen XML-tiedostoon tallennetun objektin tiedot ja tallentaa ne listarakenteeseen. Jokaisesta objektista luetaan tyyppi ja koordinaatit. Objektien tiedot sisältävä lista käydään läpi alkio kerrallaan ja luodaan tietojen perusteella vastaava objekti virtuaalimaailmaan. Aluksi tarkastellaan minkä tyyppinen objekti on kyseessä ja kutsutaan tämän perusteella objektin luovaa funktiota. Erilaisten objektien luomiseen on omat funktionsa, koska eri tyyppin objektit luodaan eri tavoin. Esimerkiksi puut tehdään billboardien avulla, jotka ovat kaksiulotteisia kuvia, jotka kääntyvät aina kameran suuntaan. Sen sijaan esimerkiksi rakennukset ja ajoneuvot ovat 3D-malleja.



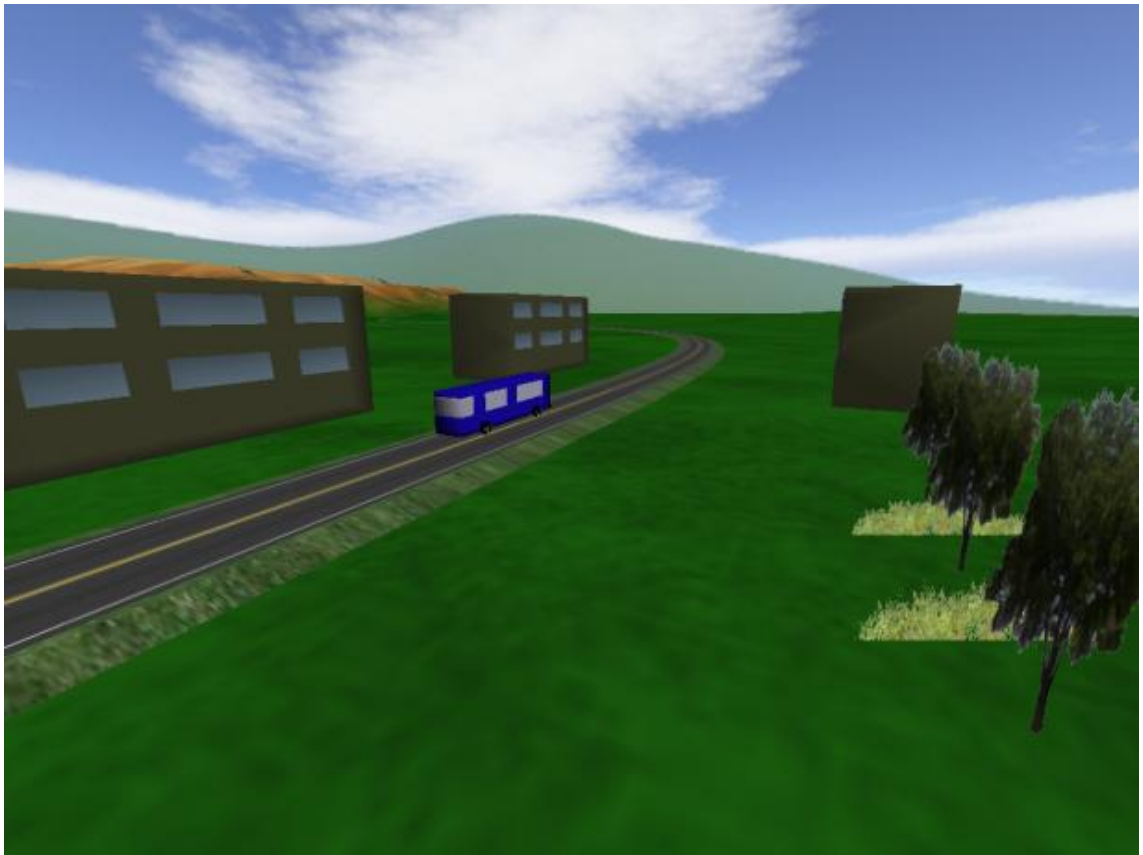
**Kuva 6.8 Objektien luominen XML-muotoisen kuvauksen perusteella**

Objektin luominen OGRE:ssä aloitetaan luomalla entity-tyyppinen muuttuja, johon luetaan objektin 3D-malli. Entity on instanssi 3D-objektista. OGRE:ssä jokaiselle virtuaalimaailmassa olevalle objektille luodaan oma entity. Jokaiselle XML-tiedostossa määritellylle objektin tyyppiä löytyy valmis 3D-malli tai billboard, joka ladataan OGRE:een entityn luomisen yhteydessä. Jokainen entity pitää nimetä yksilöllisesti, joten nimenä käytetään objektin tyyppiä, jonka perään on lisätty juokseva numerointi. Entityn luomisen jälkeen asetetaan entityn ominaisuudet. Ominaisuuksia ovat muun muassa materiaali, paikka ja orientaatio. Kaikkia entityn ominaisuuksia ei ole pakko erikseen asettaa, vaan voidaan käyttää oletusarvoja. Objektin paikka asetetaan XML-tiedostoon tallennettujen tietojen perusteella. Pituus- ja leveys-suuntaiset koordinaatit saadaan suoraan tiedoista, mutta korkeustieto pitää laskea maaston pinnanmuotojen perusteella.

Kun entity on luotu ja halutut ominaisuudet asetettu, pitää se vielä liittää osaksi OGRE:n scenegraph rakennetta. Tämä tehdään luomalla sceneNode, eli solmu, joka on osa scene graphia. SceneNode luodaan lapsisolmuksi joko maailman juurisolmulle tai jollekin muulle aiemmin luodulle solmulle. Lopuksi entity liitetään luotuun sceneNodeen.

## 6.6. Lopputulos

Visualisoinnin lopputuloksena saadaan virtuaalimaailma, jota voisi periaatteessa käyttää ajoneuvosimulaattorissa. Kuvassa 6.9 on esitetty ruudunkaappaus rakennetusta virtuaalimaailmasta. Käytännössä toteutuksessa on monia puutteita, jotka rajoittavat sen käyttökelpoisuutta todellisessa ympäristössä. Esimerkiksi teiden risteykset ovat ongelmallisia, sillä nykyisessä toteutuksessa tiet menevät risteyksessä osittain päällekkäin, joka aiheuttaa ongelmia. Automatisoinnin ansioista virtuaalimaailman rakennetta ja objektien sijainteja voidaan kuitenkin helposti muuttaa ja luoda näin erilaisia liikenneympäristöjä.



Kuva 6.9 Ruudunkaappaus automatisoidusti luodusta virtuaalimaailmasta

## 7. ARVIOINTI JA JATKOKEHITYS

Koulutussimulaattorikäyttöön soveltuvan 3D-visualisoinnin tekeminen on suuri ja työläs tehtävä. Nykyään on kuitenkin saatavilla runsaasti sekä kaupallisia, että ilmaisia 3D-visualisointikirjastoja, joita käyttämällä työ nopeutuu. Visualisointikirjastot ovat nykyään käytettävyydeltään hyviä ja niiden avulla on mahdollista saada melko nopeasti tehtyä yksinkertainen visualisointi. Visualisointikirjastot tarjoavat suuren määrän valmiita toimintoja, jotka on ennen tarvinnut ohjelmoida itse alusta alkaen. Valmiissa visualisointikirjastoissa on usein myös sisäänrakennettuja optimointimenetelmiä, jotka mahdollistavat laajojenkin visualisointien tekemisen tehokkaasti.

Visualisointikirjastot helpottavat visualisoinnin esittämistä, mutta 3D-visualisointiin tarvittavien 3D-mallien tekemistä ne eivät suoranaisesti nopeuta. Tässä työssä tutkittu 3D-mallien automaattinen generointi perustuu semanttisen tiedon käyttöön. Käytännössä hyödynnetty tieto on erilaisissa formaateissa olevaa paikkatietoa. Tämä on periaatteessa hyvin toimiva ratkaisu, sillä paikkatietoaineistoa on saatavilla lähes koko maapallon alueelta. Käytännössä paikkatiedon hyödyntäminen osoittautui hieman hankalammaksi, koska paikkatietoa voidaan tallentaa hyvin monessa eri formaatissa, jotka eivät useinkaan ole keskenään yhteensopivia. Myös paikkatiedon tarkkuus vaihtelee paljon eri alueilla. Esimerkiksi tiestön generointiin käytettävän paikkatiedon pitää olla riittävän tarkkaa, tai muuten generoidusta tiestä ei saada tarpeeksi tarkkaa simulaattorikäyttöön.

3D-mallien generoinnissa tarvitaan myös korkeustieto, mutta suuri osa paikkatietoaineistoista on vielä kaksiulotteisia, eli ne eivät sisällä korkeustietoa. Erityisesti tiestöä kuvaavaa kolmiulotteista paikkatietoaineistoa on yllättävän heikosti saatavilla. Tilanne on kuitenkin paranemassa, sillä nykyään tehtävissä uusissa paikkatietoaineistoissa on lähes poikkeuksetta mukana myös korkeustieto. Myös paikkatiedon tarkkuus paranee jatkuvasti kehittyneiden mittausmenetelmien myötä.

Paikkatietoaineisto sisältää alun perin koordinaattitiedot, joista selviää tarkalleen, mistä kohtaa maapalloa aineisto on peräisin. Luvun kuusi esimerkkitutetuksessa on käytössä vain grafiikkamoottorin paikallinen koordinaatisto, joka ei huomio paikkatiedon alkuperäisiä koordinaattitietoja. Paikallisen koordinaatiston käyttö oli riittävä ratkaisu esimerkkitutetuksen pieneen virtuaalimaailmaan, mutta jos haluttaisiin rakentaa suuri virtuaalimaailma, kannattaisi toteuttaa järjestelmä, joka huomio alkuperäiset koordinaattitiedot ja osaa sijoittaa paikkatiedon pohjalta luodut 3D-mallit automaattisesti oikeaan paikkaan.

Tässä työssä käytettiin kaksiulotteista vektorimuotoista paikkatiedodataa tiestön generoimiseen. Tarvittava korkeustieto tiestölle haettiin maastomallista, joka puolestaan

generoitiin korkeuskartan perusteella. Tällä tavalla saatiin kierrettyä kolmiulotteisen paikkatietoaineiston puuttumista tiestön generoinnissa. Tämä oli ihan hyvin toimiva ratkaisu, mutta maastomalli aiheuttaa tiettyjä rajoitteita tarkkuutensa puolesta. Mikäli käytössä olisi kolmiulotteinen kuvaus tiestöstä, olisi järkevää luoda aluksi tiestö ja muodostaa maasto tien ympärille jälkikäteen. Näin tiestö voitaisiin luoda suurella tarkkuudella ja maastossa puolestaan voitaisiin käyttää huomattavasti pienempää tarkkuutta.

Tiestön 3D-mallin muodostaminen vektorimuotoisen, pisteistä koostuvan, kuvailutiedon pohjalta on toimiva ratkaisu, mutta se sisältää myös muutamia ongelmia. Yksi ongelma oli se, että tien mutkat eivät ole riittävän sulavia pelkkien pisteiden perusteella luotuna. Tien tarkkuutta parannettiin muodostamalla aluksi pisteiden pohjalta käyrä, joka tasoittaa tien muotoja. Tämän käyrän avulla muodostettiin lopulta tien 3D-malli. Tien korkeussuuntainen muoto muodostettiin maaston pinnanmuotojen mukaan. Maaston muodot ovat kuitenkin epätarkempia ja tästä syystä tien korkeussuuntaiset muodot ovat tietyissä kohtaa liian jyrkkiä. Myös korkeussuuntaiset muodot voitaisiin muodostaa käyrän avulla, jolloin niistäkin saataisiin sulavampia. Ongelmaksi muodostuisi tällöin se, että tie ei enää seuraisi maaston muotoja, vaan voisi välillä vajota maaston sisään ja välillä taas olla maaston yläpuolella.

Teiden risteyskohdat ovat nykyisellä toteutuksella ongelmallisia, sillä niissä tiet menevät osittain päällekkäin. Tämä saattaa aiheuttaa muun muassa välkkymistä päällekkäin menevissä osissa. Ratkaisuna tähän ongelmaan olisi luoda risteysalueiden polygonit vasta sitten, kun kaikki risteykseen tulevat tiet on luotu. Tällöin olisi mahdollista rakentaa risteysalueen 3D-malli siten, että siinä ei ole päällekkäin meneviä osia. Oikeaa ajoneuvosimulaattoria varten risteyksissä pitäisi käyttää muutenkin erilaisia tiemalleja, joissa esimerkiksi ryhmittymiskaistat ja ajokaistamerkinnot olisivat oikein aseteltuja. Risteyskohtia tehtäessä olisikin suuri apu, mikäli käytettävissä olisi valmis liikennesuunnitelma, jossa tällaiset asiat on jo valmiiksi suunniteltu. Liikennesuunnitelmien käyttöä 3D-mallien generoinnissa tutkittiin, mutta ongelmaksi muodostui se, että suunnittelussa ei ole vielä yhtenäisiä standardeja, vaan eri suunnitelmissa käytetään hyvin erilaisia tallennusformaatteja. Osa formaateista on myös suljettuja, eli niiden sisäisen rakenteen kertovaa spesifikaatiota ei ole julkisesti saatavilla. Ilman tietoa tallennusformaatin muodosta, niitä ei pysty käyttämään omilla ohjelmissa. Mikäli tulevaisuudessa otetaan käyttöön yhtenäinen standardi liikennesuunnitelmien tekemisessä, kannattaa niiden hyödyntämistä tutkia uudelleen.

Laajemman virtuaalimaailman rakentamisessa kannattaisi hyödyntää editoria, jonka avulla käsityön määrää voitaisiin vähentää ja samalla nopeuttaa rakentamista. Monipuolisen editorin tekeminen toisi paljon mahdollisuuksia virtuaalimaailman rakentamiseen. Editorin avulla voisi asetella virtuaalimaailman esineet haluttuihin paikkoihin ja muokata niiden ominaisuuksia. Editori voisi tarjota myös vaihtoehdon valmiin paikkatiedon käytölle siten, että sillä voisi luoda teitä ja maastoa tyhjästä ilman paikkatietoa. Eräs vaihtoehto olisi irrottaa paikkatiedosta pienempiä osia, joita voisi editorin avulla yhdistellä ja rakentaa niistä toimivan virtuaalimaailman.

## 8. YHTEENVETO

Nykyaikaisissa koulutussimulaattoreissa 3D-visualisointi on yleisesti käytetty työväline virtuaalimaailman havainnollistamiseen. Nopeasti kehittyvät visualisointimenetelmät ja -laitteisto mahdollistavat näyttävien visualisointien esittämisen, mutta samalla visualisoinnin tekemiseen käytettävä aika on kasvanut. Tämän työn tarkoituksena oli tutkia, mitä ajoneuvosimulaattoriin soveltuvalta 3D-visualisoinnilta vaaditaan ja miten se voidaan toteuttaa. Toinen aihealue oli tutkia, miten 3D-visualisointiin vaadittava virtuaalimaailma voidaan rakentaa tehokkaasti ja miten rakennusprosessia voisi automatisoida.

Nykyään tarjolla on paljon 3D-visualisointikirjastoja, jotka soveltuvat hyvin myös koulutuskäyttöön tarkoitetun ajoneuvosimulaattorin visualisoinnin toteuttamiseen. Valmiin visualisointikirjaston käyttö nopeuttaa oleellisesti työtä verrattuna siihen, että ohjelmoisi kaikki työvälineet itse. Simulaattorikäytössä graafisen ulkoasun on tärkeää olla selkeä ja ruudunpäivitysnopeuden tulee olla tasainen. Nykyaikainen laitteisto mahdollistaa hyvinkin tarkkojen visualisointien esittämisen reaaliaikaisesti ilman, että ruudunpäivitysnopeus siitä kärsii.

Virtuaalimaailma koostuu 3D-malleista, joiden tekeminen käsityönä mallinnusohjelmalla on hidasta. Ajoneuvosimulaattorissa tärkeimpiä kohteita ovat tiet ja tien ympärillä oleva maasto. Nämä kohteet ovat myös varsin laajoja, joten niiden tekeminen käsin on hyvin työlästä. Tästä syystä tutkittiin miten näiden kohteiden luomista voisi automatisoida. Automatisoitu luominen perustettiin semanttisten kuvausten käyttöön. Semanttinen aineisto oli tässä tapauksessa paikkatietoaineistoa, jossa kuvaillaan tiestön ja maaston muotoja. 3D-mallien muodostamiseen käytettävän aineiston tulee olla kolmiulotteista, ja tämä aiheutti ongelmia, sillä suuri osa nykyisestä paikkatiedosta on vain kaksiulotteista. Uudet paikkatietoaineistot tehdään nykyään kuitenkin lähes aina kolmiulotteisena, joten tulevaisuudessa paikkatietoa voidaan käyttää yhä monipuolisemmin myös 3D-visualisoinnin tarpeisiin.

Luvussa 6 esitetty käytännön toteutus ajoneuvosimulaattorin 3D-visualisoinnista ja virtuaalimaailman rakentamisesta automaattisen generoinnin keinoin kuvaa yhden lähestymistavan tärkeimmät periaatteet, mutta se ei pyrikään olemaan täydellinen kuvaus. Näin ollen se ei myöskään sellaisenaan sovellu käytettäväksi oikeassa simulaattorissa. Se kuitenkin todistaa, että automaattinen generointi on hyödyllinen työväline 3D-visualisoinnissa ja sen avulla virtuaalimaailman rakennusprosessia voidaan nopeuttaa.

## LÄHTEET

[Abdul-Rahman & Pilouk 2008] Abdul-Rahman A., Pilouk M. 2007. Spatial Data Modelling for 3D GIS. Springer

[Alessi & Trollip 1985] Alessi M., Trollip R. 1985. Computer-Based Instruction Methods and Development. Prentice-Hall

[Akenine-Möller *et al.* 2008] Akenine-Möller T., Haines E. & Hoffman N. 2008. Real-Time Rendering. 3. painos. A K Peters.

[AMD] Suoritin- ja näytönohjainvalmistaja AMD. <http://www.amd.com>

[Autokoululiitto] Autokoululiiton ja Faroksen pimeäajosimulaattori. Viitattu: 24.2.2010. <http://www.autokoululiitto.fi/sivu.php?uid=13&id=10>

[ESRI] ESRI Shapefile technical description. Saatavilla: <http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf>

[Faros] Eca Faros ajoneuvosimulaattori. [http://www.ecafaros.com/en/liste-produit.htm?\\_rub=1&\\_srub=3](http://www.ecafaros.com/en/liste-produit.htm?_rub=1&_srub=3)

[Huttunen 1998] Huttunen T. 1998. Paikkatietojen hyödyntäminen tiensuunnittelun eri vaiheissa, Tielaitoksen selvitys 51/1998. Oy Edita Ab.

[Hyvärinen *et al.* 2006] Hyvärinen J, Porkka J, Pienimäki M, Korkiala-Tanttu L, Mäkeläinen T & Kiviniemi A. 2006. Infra 2010-hanke, Infra-alan tuotetietomalliselvitys Yhteenvetoraportti. Saatavilla: <http://www.infra2010.fi/Aineisto/tuotemalliselvitys.pdf>

[IJSC] International Journal of Semantic Computing (IJSC). <http://www.worldscinet.com/ijsc/>

[Insta] Instan SimCore visualisointijärjestelmä. [http://www.insta.fi/insta\\_defsec/puolustusratkaisut/simcore-maisemavisualisointi/](http://www.insta.fi/insta_defsec/puolustusratkaisut/simcore-maisemavisualisointi/)

[Isännäinen 2010] Isännäinen V. 2010. Koulutussimulaattorin arkkitehtuurin suunnittelu. Diplomityö. Tampereen teknillinen yliopisto.

[John Deere] John Deere metsäkonesimulaattori. [http://www.deere.com/fi\\_FI/equipment/forestry/virtual/index.html](http://www.deere.com/fi_FI/equipment/forestry/virtual/index.html)

[Kalawsky 1993] Kalawsky R. S. 1993. The Science of Virtual Reality and virtual environments. Addison Wesley.

[Kolasinski 1995] Kolasinski E. M., Simulator Sickness in Virtual Environments, U.S. Army Research Institute Technical Report 1027, 1995

[Kuhl & al. 1995] Kuhl J., Evans D., Papelis Y., Romano R., Watson G. 1995. The Iowa Driving Simulator: An Immersive Research Environment. Computer Volume 28, Issue 7, July 1995

[Laakkonen 2009] Laakkonen M. 2009. Tiemittauksen asettamat vaatimukset suunnittelulle. Opinnäytetyö, Metropolia Ammattikorkeakoulu.

[Nvidia] Näytönohjainvalmistaja Nvidia. <http://www.nvidia.com>

[Open Design] Open Design Alliance. <http://www.opendesign.com>

[Peltoniemi 2001] Peltoniemi R. 2001. Maavoimien simulaattoriavusteisen koulutuksen optimointi – maavoimien simulaattoristrategia. Maanpuolustuskorkeakoulu, Koulutustaidon laitos. Julkaisusarja 2, No 9. Helsinki. Edita ab.

[Prothero 1998] Prothero J.D. The Role of Rest Frames in Vection, Presence and Motion Sickness. University of Washington. 1998. Saatavilla: <http://www.hitl.washington.edu/publications//r-98-11/r-98-11.pdf>

[Puhakka 2006] Puhakka A. Tietokonegrafiikka-luentokalvot. 2006

[Sharma 2009] Sharma M. 2009. Constructing Virtual Environments Based on Semantic information Sources into Training Simulator. Master of Science Thesis. Tampere University of Technology

[Simrac] Team Simrac bussisimulaattori. <http://www.simrac.com/>

[Teikari & Vartiainen 1985] Teikari V., Vartiainen M. 1985. Simulaatio työtaidon kehittäjänä. Teknillinen korkeakoulu.

[TRAINER D.2.1] TRAINER, Inventory of driver training needs and major gaps in the relevant training procedures. Deliverable 2.1.

[TRAINER D.4.1] TRAINER, Driving simulator scenarios and requirements. Deliverable 4.1.

[van Emmerik 2004] van Emmerik M. 2004. Beyond the Simulator, Instruction for high-performance tasks. Doctoral Thesis. University of Twente.

[Walrath 1999] Walrath J. 1999. 30 Frames per Second vs. 60 Frames per Second. Viitattu: 12.10.2009. Saatavilla: [http://www.daniele.ch/school/30vs60/30vs60\\_1.html](http://www.daniele.ch/school/30vs60/30vs60_1.html)

[Watt 2000] Watt, A. 2000. 3D Computer Graphics. 3. painos. Addison Wesley.