3D-PELIEN KENTTÄSUUNNITTELU

LAHDEN AMMATTIKORKEAKOULU
Mediatekniikan koulutusohjelma
Teknisen visualisoinnin suuntautumisvaihtoehto
Opinnäytetyö
5.5.2008
Markus Kallio

Lahden ammattikorkeakoulu

Mediatekniikan koulutusohjelma

KALLIO, MARKUS: 3D-pelien kenttäsuunnittelu

Teknisen visualisoinnin opinnäytetyö, 59 sivua, 1 liitesivu

Kevät 2008

TIIVISTELMÄ

Opinnäytetyössä käsitellään videopelien kenttäsuunnittelua sekä teorian, että käytännön soveltamisen kautta. Työssä selvitetään, mitä kenttäsuunnittelu on käytännössä, millainen on modernissa peliprojektissa työskentelevän kenttäsuunnittelijan työnkuva ja mitä osaamisalueita häneltä vaaditaan. Lisäksi opinnäytetyössä käsitellään kenttäsuunnittelun historiaa, teknologiaa videopelien takana sekä pelattavuutta ja pelikenttien rakentamista käytännössä.

Kenttäsuunnittelu on erittäin laaja käsite, koko käsitteen selittäminen opinnäytetyön puitteissa on mahdotonta. Työssä onkin pyritty vain selvittämään kenttäsuunnittelun pääpiirteet. Tämän lisäksi keskitytään syvemmin tiettyihin aihealueisiin, kuten grafiikan luomiseen, pelattavuuteen ja pelaajan ohjaukseen.

Opinnäytetyön case-osiossa rakennettaan esimerkkikenttä Source-pelimoottorille. Pelikentässä esitellään opinnäytetyössä käsiteltyjä grafiikkatekniikoita ja pelaajan ohjausta.

Avainsanat: Kenttäsuunnittelu, level design, reaaliaikainen grafiikka, pelattavuus, videopelit

Lahti University of Applied Sciences

Faculty of Technology

KALLIO, MARKUS: Level design for 3D video games

Bachelor's Thesis in Visualization Engineering, 59 pages, 1 appendix

Spring 2008

ABSTRACT

This Bachelor thesis is about leveldesing for 3D games. The subject is analysed through theory and practical examples. The objective of the study was to explain what level design is all about, what the stages are and what kind of skills are required from the leveldesigner working on a modern video game project.

In addition, the thesis also deals with the history on level design and the techonology involved and explains terms such as gameplay and gameflow.

The thesis draws only a broad outline of level design. As level design is a very large concept, it would have been impossible to explain everything about the subject within the limits of this thesis.

The case section is about an example level that was built on the Source-engine. The level presents elements such as graphic techniques and player directing.

Key words: Level design, real-time graphics, gameplay, video games

TERMIT JA LYHENTEET

FPS,	first person shooter, Ensimmäisestä persoonasta eli pelihahmon silmistä kuvattu räiskintäpeli.
Entity,	objekti, jolle voidaan määrittää erilaisia ominaisuuksia
Mod,	pelaajien toimesta tehty, yleensä ilmainen pelilaajennus
Verteksi,	monikulmion piste, jossa kaksi sivua kohtaa toisensa, 3D-grafiikan peruselementti
Polygoni,	muodostuu vähintään kolmesta verteksipisteestä
Plane,	taso, joka muodostuu kahdesta polygonista
Materiaali,	2D-kuvaan voidaan lisätä mm. kiiltoja, displacement-, bumb- ja normalmap -tasoja.
Node,	verkoston solmukohta, jolle voidaan asettaa jonkin arvo
NPC,	Non player character eli ei-pelaajahahmo, tietokoneen kontrolloima pelihahmo
Trigger,	laukaisin, käytetään laukaisemaan tapahtuma
Alpha-kanava,	määrää kuvan tai tason läpinäkyvyyden
Skybox,	kaikki pelikentän ulkopuolinen tila muodostuu skyboxista, pelikentän tausta
Immersio,	käyttäjän tunne siitä, että hän sulautuu mediaan ja "uppoaa" virtuaaliseen maailmaan
Pelimoottori,	runko-ohjelma, joka yhdistää pelimoottorin eri ominaisuudet toimivaksi kokonaisuudeksi eli pyörittää peliä
Fysiikkamoottori,	ohjelma, joka simuloi esineiden fyysistä käyttäytymistä
Primitiivi,	yksinkertainen muoto, esimerkiksi ympyrä tai neliö

1 JOHDANTO

Kenttäsuunnittelu on varsin uusi ala peliteollisuudessa. Vasta noin vuodesta 1995 lähtien kenttäsuunnittelu on ollut varsinainen ammattinimike. Peliala kehittyy todella nopeasti samoin kuin kenttäsuunnittelu ja lisääntyvien vaatimusten myötä myös kenttäsuunnittelu on pirstaloitumassa eri segmentteihin.

Kenttäsuunnittelu on videopelikenttien sekä niissä esiintyvien skenaarioiden suunnittelua ja toteuttamista. Kenttäsuunnittelijalta vaaditaan sekä taiteellista näkemystä että teknistä osaamista. Kenttäsuunnittelu on se rajapinta, joka yhdistää pelin eri elementit, kuten äänen, kuvan ja pelattavuuden toimivaksi paketiksi. Nykyään kenttäsuunnittelija voi erikoistua muun muassa grafiikkaan tai pelattavuuteen.

Tässä opinnäytetyössä käsitellään kenttäsuunnittelua sekä teorian että käytännön soveltamisen kautta. Työssä pohditaan kenttäsuunnittelijan työnkuvaa ja pelien sekä kenttäsuunnittelun historiaa. Syvällisemmin työssä käsitellään pelattavuutta, pelaajankuljetusta ja pelimaailmojen mallinnusta. Pelimoottoreita ja kenttäeditoreja on lähes yhtä monta kuin ilmestyviä pelejä, joten aiheita on käsitelty yleisesti, siten että tiedot voidaan helposti soveltaa mille tahansa pelimoottorille tai kenttäeditorille.

Case osiossa käsitellään pienen esimerkkikentän rakentamista Source-pelimoottorille.

2 KENTTÄSUUNNITTELU

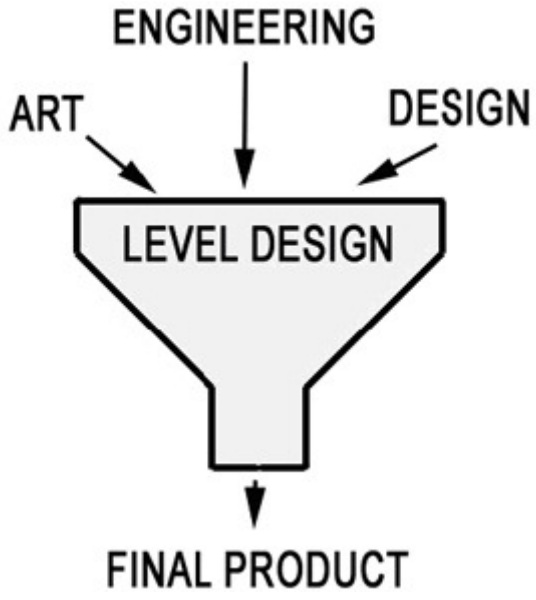
2.1 Yleistä

Tietokonepelit ovat olleet olemassa noin 30 vuotta. Alkuaikoina pelin tekemiseen tarvittiin vain yksi henkilö ja hän vastasi pelin grafiikasta, ohjelmoinnista, suunnittelusta, äänistä ja pelattavuudesta. Nykyään, kun pelit ovat entistä monimutkaisempia ja laajempia, työtehtävät ovat jakaantuneet pienempiin pirstaleisiin. Yksi näistä pirstaleista muodostuu kenttäsuunnittelusta.

Pelisuunnittelijat luovat säännöt ja järjestelmät, jotka muodostavat jokaisen pelin selkärangan. Kenttäsuunnittelija toteuttaa ja laittaa suunnitelmat toimimaan käytännössä. Sellaisenaan kenttäsuunnittelulla on erittäin tärkeä rooli nykypäivän pelituotannossa, koska kärjistetysti, pelaaja kokee pelin juuri kenttien kautta. (Byrne 2004, 11.)

Kenttäsuunnittelija ei itse luo 3D-malleja, tekstuureja eikä hahmoja, eikä edes ohjelmoi peliä, mutta hän järjestelee nämä elementit siten, että siitä muodostuu jotain hauskaa ja viihdyttävää. (Feil, Scattergood 2005, 4.)

Pähkinänkuoressa kenttäsuunnittelu on työtä, jossa käännetään pelitiimin ideat pelattavaan muotoon. Kenttäsuunnittelija on se rajapinta, jossa ohjelmointi, elokuva, ääni, taide ja suunnittelu kohtaavat (kuva 1.) (Byrne 2004, 10.)



KUVA 1. Kenttäsuunnittelun rajapinta (Byrne 2004)

2.2 Visuaalinen ja tekninen osaaminen

Kenttäsuunnittelu on yhtä paljon taidetta kuin tiedettä. Se vaatii taiteellisia kykyjä ja hyvää teknistä tietämystä. Suunnittelija, joka on taiteellinen ja pätevä arkkitehti, ei menesty, jollei hän pysty käsittämään aiheita, kuten ruudunpäivitys, pelattavuus ja ajoitus. Suunnittelija, joka ymmärtää nämä elementit, mutta jolla ei ole tippaakaan taiteellista tai arkkitehtuurista kykyä, on tuomittu yhtälailla epäonnistumaan. Taide ja tiede ovat kenttäsuunnittelun kaksi puolta, toinen ei voi olla olemassa ilman toista. (Bleszinski 2000.)

Nykypäivänä kenttäsuunnittelijan rooli voi olla hyvin teknistä tai koostua pelkästään tekijäryhmän johtamisesta, varsinkin jos kyseessä on isompi projekti. Huolimatta siitä kuka luo kentän sisällön, kenttäsuunnittelija on silti se, jonka näkemys kentän lopullinen ulkoasu on. Kenttäsuunnittelijalla täytyy olla kyvyt toteuttaa tuo visio itse, tai pystyä selittämään visio tarpeeksi perusteellisesti muille työntekijöille.

Kenttäsuunnittelussa ei ole kuitenkaan kyse pelkästä visuaalisesta puolesta, sillä yhtä tärkeää on hyvä pelattavuus. Tasapainoilu taiteellisten ja pelillisten elementtien välillä on jokapäiväinen osa modernia kenttäsuunnittelua.(Byrne 2004 4-5.)

2.3 Kenttäsuunnittelun historia

Videopelien 30-vuotisen historian aikana, pelit ovat kehittyneet harrastelijoiden puuhista miljardiluokan bisnekseksi. Mikään muu viihteenmuoto ei ole saavuttanut näin suurta suosiota näin lyhyessä ajassa. Suosion lisäksi myös peleissä oleva sisältö ja teknologia ovat kehittyneet räjähdysmäisesti.

On tärkeää huomata, että kenttäsuunnittelu ei tarkoita ainoastaan kolmiulotteista suunnittelua, vaan se on jotain mikä kuuluu osaksi jokaista pelityyppiä. Ylimääräinen ulottuvuus kolmiulotteisissa peleissä aiheuttaa huomattavan määrän lisätyötä kenttäsuunnittelijoille. Nyt suunnittelijat joutuvat miettimään liikettä kaikilla kolmella akselilla – x, y ja z, pelkästään x:n ja y:n sijasta. 3D-pelien tekniikan nostaminen nykyiselle tasolleen ei ole ollut helppo tehtävä. Ala on kokenut ennen Unreal Tournament, Doom3, Half-Life 2 tai F.E.A.R pelejä lukuisia pieniä ja suurempia kehitysaskelmia sekä myös epäonnistumisia. (Shahrani 2004.)

Eräs kehitysaskelma oli Doom (1993). Se oli Id softwaren jatko-osa jo alun perin melkoisena vedenjakajana toimineelle Wolfenstein 3D:lle. Doomin myötä myös kenttäsuunnitteluun alettiin kiinnittää yhä enemmän huomiota.

Doomin pelimoottori tuki useita uusia toimintoja, jotka viimein mahdollistivat realistiset ja interaktiiviset ympäristöt. Uutta Doomissa oli myös triggerit, näillä pystyttiin muuttaman peliympäristöä ja pelitapahtumia pelaajan etenemisen mukaisesti. Esimerkiksi siltatasoja pystyttiin nostamaan ylipääsemättömien esteiden päälle. Triggerit antoivat kenttäsuunnittelijoille työkalut, joilla voitiin suunnitella ansoja ja pelitilanteita. Monia Doomin aikoihin kehitettyjä ansoja sovelletaan nykyäänkin. (Shahrani 2004.)

Doomin kenttäsuunnitteluun käytettiin kehittyneempää kenttätyökalua kuin missään aikaisemmassa pelissä. John Romero ohjelmoi pelimoottorille suunnatun kenttäeditoriohjelman DoomEd. 3Drealms vastaavasti kehitteli vuonna 1996 peliä Duke Nukem 3D, jonka kenttäeditori Build oli vielä paljon DoomEdiä kehittyneempi. Build esimerkiksi sisälsi reaaliaikaisen ”mitä näet on mitä saat” (wysiwyg) käyttöliittymän, jonka avulla suunnittelija pystyi ensin suunnittelemaan kentän kaksiulotteisesti ja vaihtamaan sitten kolmiulotteiseen esikatseluun, josta näki suoraan millainen kenttä on pelimoottorin sisällä. Aiemmin kenttä piti ensin kääntää pelimoottorin ymmärtämään muotoon ennen kuin sitä voitiin tarkastella. (Shahrani 2004.)

Seuraava hyppy pelien ja myös kenttäsuunnittelun historiassa tapahtui, kun Quake saapui pelimarkkinoille hieman Duke Nukem 3D:n jälkeen vuonna 1996. Se ei ollut pelillisesti samanlainen innovaattori kuin Duke Nukem 3D, mutta Quaken suuri innovaatio oli täysin kolmiulotteinen kenttäarkkitehtuuri. Quaken pelimoottoria lisensoitiin ennätysmäärin. Suurin osa tuon ajan hittipeleistä käytti jotain versiota Quaken pelimoottorista.

Valve Software julkaisi Half-Life pelinsä jouluna 1998. Peli sijoitti pelaajan Gordon Freeman nimisen tiedemiehen saappaisiin. Half-Life uudisti ensimmäisen persoonan räiskintäpeli-genreä. Ensinnäkin Half-Lifen tarinankuljetus oli uudenlaista, niin sanottua elokuvamaista tarinankuljetusta. Valve panosti myös pelaajan immersioon entistä enemmän. Sen sijaan, että pelaajalle kerrottiin pelintapahtumista tekstin tai kertojan välityksellä, Half-Lifessä pelaajaa ohjastavat muiden pelihahmojen kom-

mentit tai esimerkiksi tiedekeskuksen keskusradiosta kuuluvat kuulutukset. Pelihahmoille lisättiin huulisynkronisaatio immersion lisäämiseksi. Pelialueiden suunnittelussa pyrittiin realistisuuteen, esimerkiksi pelin aloittava junamatka oli todenmukainen ja hyvin erilainen aloitus kuin muissa tuon ajan peleissä. Peli myös kehitti kenttien välistä siirtymistä, sen sijaan että pelikenttä loppui kuin seinään, Half-Lifessa seuraavaan kenttään siirryttiin saumattomasti. Jokainen kenttä oli osa isompaa kokonaisuutta. Tällainen kenttäarkkitehtuuri vaikeutti kenttäsuunnittelijoiden elämää, mutta toisaalta edesauttoi pelaajan immersiota huomattavasti. (Shahrani 2004.)

Myös Half-Life perustui muokattuun Quake-pelimoottoriin. Valven muutokset moottoriin olivat huomattavia. Pelimoottoriin lisättiin esimerkiksi värilliset valot, tuki skriptatuille tapahtumille, jotka toimivat tarinankerronnan elementteinä ja paremmat työkalut ulko- ja sisätilojen mallintamiseen. Kun hieman pelin ilmestymisen jälkeen Valve julkaisi pelille kehitystyökalut, ilmestyi peliin lukuisia modeja eli käyttäjien tekemiä pelejä, muun muassa Counter-Strike ja Day of Defeat. (Shahrani 2004.)

2.4 Kenttäsuunnittelun tulevaisuus

Tulevaisuudessa elokuvateollisuudesta tutut työskentelytavat yleistyvät myös peliprojekteissa. Osa kenttäsuunnittelijoista erikoistuu esimerkiksi pelkästään valaistukseen, osa teksturointiin. Työnä kenttäsuunnittelu tulee jatkossa vain monimutkaistumaan. Esimerkiksi, kun aikoinaan Wolfestein 3D:n kentän tekemiseen kului vain muutamia tunteja, pelien kuten Half-Life 2-kenttien tekemiseen vaaditaan kokonaisen ihmisjoukon viikkojen tai kuukausien työpanos. (Shahrani 2004.)

Tulevaisuudessa tuskin enää nähdään yhden ihmisen suunnittelemia kenttiä. Työmäärän kasvattamisen lisäksi myös kenttäsuunnitteluun käytettävät työkalut tulevat kehittymään, työkalut ovat kehittyneet hieman hitaammin kuin pelimoottorit, joten tilaa käyttöliittymien ja toimintojen parantamiseen on. (Shahrani 2004.)

Peligrafiikan kehittyessä yhä lähemmäs esirenderoitua grafiikkaa, kenttäeditorit tulevat muistuttumaan yhä enemmän kehittyneimpiä 3d-mallinnusohjelmia, kuten Lightwave 3D:tä tai Maya:aa. Monet kehittäjät ovatkin siirtyneet käyttämään tällaisia ohjelmia itsenäisten kenttäeditorien sijasta. Kenttäsuunnittelijan kannattaakin opetella Mayan tai Lightwaven perusteet. Tulevaisuudessa kenttäeditorit tulevat muistuttumaan

käytettävyydellään ja ominaisuuksiltaan näitä ohjelmia. (Bleszinski 2000.)

Pelimoottorien kehittyessä myös grafiikalta vaaditaan enemmän. Kenttäsuunnittelijoilta vaaditaankin nykyään enemmän monipuolisuutta: esimerkiksi Epic Megagamesin kenttäsuunnittelijat ovat taitavia kenttäsuunnittelijoita ja osa on päteviä tekstuurisuunnittelijoita ja osa hallitsee mallintamisen tai sisustamisen. Moniosaaminen on tärkeä ominaisuus, taitava kenttäsuunnittelija voi korvata hetkellisesti työryhmästä puuttuvan jäsenen. Toisaalta töiden kasautumisessa yhdelle ihmiselle on omat riskinsä. (Bleszinski 2000.)

2.5 Ryhmätyöskentely

Peliteollisuus kehittyy jatkuvasti, vielä vähän aikaa sitten ei ollut edes olemassa käsitettä ”kenttäsuunnittelija”, mutta nykyään hän on yksi peliprojektin tärkeimpiä jäseniä. Kenttäsuunnittelijalle on tärkeää tietää mihin lokeroon hän tuotantoryhmässään kuuluu. (Bleszinski 2000.)

Pelituotantoryhmät ovat kasvaneet koossa. Sellaisen asian hoitamiseen, jonka ennen pystyi hoitamaan yksi henkilö, tarvitaan nykyään tekemään kaksi tai kolme henkilöä. 90-luvun alussa pelialalle ruvettiin palkkaamaan ammattilaisgraafikoita, sillä ohjelmoijat eivät enää pystyneet tekemään grafiikkaa ja ohjelmointia. Uudet työtehtävät, kuten graafikko, pelisuunnittelija, kenttäsuunnittelija ja hahmoanimaattori, luotiin helpottamaan pelikehitystä. Räjähtävästi monimutkaistuneiden pelien ja jatkuvasti nousevien odotusten takia pelituotannoissa saattaa työskennellä nykyään 10, 30 tai jopa 100 henkilöä vuosien ajan. Tällaisessa ympäristössä työ on jaettu hyvin pieniin segmentteihin. Nykyään myös kenttäsuunnittelu on usein jaettu eri segmentteihin. (Byrne 2004, 11.)

Kun peliä työstetään ryhmässä, oli kyseessä sitten virallinen tai modiprojekti, täytyy kenttäsuunnittelijan käydä aina tiivistä keskustelua pelisuunnittelijoiden kanssa. Samoin on elintärkeää tuntea pelin ominaisuudet ja pelin elementit. (Duarte 2005.)

Nykyään kenttäsuunnittelijan työnkuva on hyvin vaikeasti määriteltävä: kenttäsuunnittelija tasapainoilee suunnittelun, taiteen ja teknisen osaamisensa välissä luodessaan kenttiä. Kenttäsuunnittelijan työnkuva työryhmän sisällä on hyvin laaja, jotkut suunnittelijoista voivat esimerkiksi

erikoistua pelkästään pelattavuuden suunnitteluun ja antaa muiden luoda kentän visuaalisen ilmeen ja muun sisällön. Kenttäsuunnittelija voi työskennellä joko pääsuunnittelijan tai (jos työryhmä on tarpeeksi iso) pääkenttäsuunnittelijan alaisena. (Byrne 2004, 33.)

Isoissa työryhmissä luovantyön johtajat, eli esimerkiksi pääkenttäsuunnittelija, ovat omistautuneet pelin visuaalisen ilmeen yhtenäistämiseen. Usein he luovat dokumentteja ja rakenteita varmistaakseen, että kaikki pelin visuaalisen ilmeen luomiseen osallistuvat työtekijät luovat yhtenevää grafiikkaa. Ilman tällaista ohjausta usean voimakastahtoisen taitelijan tiimeissä, jokainen kenttä ja jokainen hahmo olisi tyyliltään erilainen. Tällöin pelistä tulee epäharmonisen näköinen ja tuntuinen. (Byrne 2004, 32.)

3 KENTTÄSUUNNITTELUN TEORIAA

3.1 Hauskuus

Peli tehdään aina pelaajan näkökulmasta, hyvän suunnittelijan tulisikin pystyä astumaan pelaajan saappaisiin. Kentät tehdään aina pelaajalle ja on tärkeää tietää mistä pelaaja nauttii ja mikä yksinkertaisesti on hauskaa. Suunnittelijan tulisi osata ennustaa pelaajan käyttäytymistä ja suunnitella kenttä sen mukaan. Hyvässä pelissä pelaaja ei koskaan tunne turhautumista, tylsistymistä tai epäreiluutta. Sen sijaan hän kokee hauskuutta, aikaansaamisen tunteita ja juuri sopivan määrän haastetta. (Duarte 2005.)

Pelien tärkein tehtävä on viihdyttää. Tämä on tärkein asia mitä pelinkehittäjän tulee pitää mielessä, oli kyseessä sitten kännykkäpeli tai massiivinen roolipeli. Suunnittelija on viihdyttäjä eikä sadistinen jumalhahmo, jonka päämääränä on tehdä pelaajan elämästä mahdollisimman tuskallista. Tärkeintä on tehdä pelistä hauska. Peliteollisuus työllistää tuhansia testaajia ja käyttää vuosittain miljoonia markkinatutkimuksiin yrittäessään määritellä, mitä on hauskuus. Tähän mennessä kukaan ei ole kuitenkaan onnistunut luomaan varmaa kaavaa hauskuuteen. (Feil, Scattergood 2005, 4.)

3.2 Pelattavuus

Pohjimmiltaan on olemassa vain kahdentyyppistä pelattavuutta: runkopelattavuutta ja kenttäpelattavuutta.

Runkopelattavuus on pelattavuutta, jonka peli itse tarjoaa. Se määrittelee pelaajan ominaisuudet, kuten esimerkiksi kuinka nopea pelihahmo on, tai millainen painovoima pelissä vaikuttaa. Jos pelaaja asetetaan tyhjään kuutioon, koostuu kaikki mitä hän näkee kuutiota lukuun ottamatta runkopelattavuudesta. Kenttäsuunnittelijalla on vain vähän tekemistä runkopelattavuuden kanssa. Se on jo olemassa, ja kenttäsuunnittelija ei voi helposti vaikuttaa siihen. Runkopelattavuus on eräänlainen perusta, johon kenttäsuunnittelija mukautuu ja jonka päälle hän alkaa rakentaa mahdollisimman hyvin perustaan sopivaa kenttää.

Kenttäpelattavuus määrää, missä pelaaja syntyy ja millaisia reittejä pelaaja kulkee. Aseiden ja esineiden asettelu on osa kenttäpelattavuutta. Runkopelattavuus määrään pelin keskeiset ominaisuudet, kenttäpelatta-

vuus määrää peliympäristön ja tapahtumat. Kenttäpelattavuuden on tarkoitus vahvistaa runkopelattavuutta. Peli, jota on hauska pelata, on eri asia, kuin kenttä, jota on hauska pelata. Hyvä kenttäpelattavuus on pelattavuutta, joka tukee ja motivoi pelaajaa muodostamaan erilaisia pelistrategioita, lisää pelin syvyyttä ja hauskuutta. Kenttäpelattavuus elementtejä ovat esimerkiksi erilaiset ansat, pulmat, korkeuserot ja haarautuvat ratkaisumallit. Ilman kenttäpelattavuutta pelin kaikki kentät olisivat samanlaisia. Jos kenttä ei tuo tarpeeksi uutta runkopelattavuuteen, se on pelattavuudeltaan epäonnistunut kenttä. (De Jong 2004.)

3.3 Pelinkuljetus

Pelinkuljetus (Gameflow) on tärkeä käsite. Pelinkuljetus on tapahtumaketju, joka tapahtuu pelin sisällä. Yksinpeleissä pelikuljetus on täysin kenttäsuunnittelijan käsissä, moninpelissä pelaajat määräävät pelinkuljetusta. Pelinkuljetus on pelin elinvoima, joka tekee pelistä hauskan ja haastaa pelaajaa etenemään pelissä. (Bleszinski 2000.)

Pelinkuljetus on kentän tärkeä elementti: se määrää, miten kenttä käyttäytyy ja miten paljon pelaaja nauttii kokemuksesta. Hyvän pelinkuljetuksen omaavassa kentässä tulisi näiden asioiden olla kunnossa:

- Korkeuserot: korkeuserot tekevät kentistä paljon mielenkiintoisempia ja taisteluista vaihtuvampia. Yhdellä tasolla käytävät taistelut tuottavat vain yhdenlaista taistelua ja käyvät nopeasti tylsiksi.
- Tasapaino: pelaaja ei saisi koskaan nousta ylivoimaiseen asemaan, toisaalta pelaaja ei saa tuntea oloaan liian heikoksi.
- Vaihtelevuus: monotoninen kenttäarkkitehtuuri ja vaihtelemattomuus käy nopeasti tylsäksi. Pelaajan voi myös eksyä helpommin jos pelitilat muistuttavat liikaa toisiaan. Vaihteleva rakenne tarkoittaa vaihtelevaa pelattavuutta.
- Huippukohdat ja huomiopisteet: kentästä löytyviä elementtejä joista kentän tunnistaa ja joiden perusteella pelaaja voi helposti suunnistaa kentän sisällä. Esim. mielenkiintoinen rakennus joka hallitsee tilaa ja houkuttaa pelaajaa luokseen.
- Kuljetus: kentän tulisi rohkaista pelaajaa liikkumaan, taaksepäin palaamista tulisi käyttää mahdollisimman harvoin.
- Valinta: pelaajat pitävät valintojen tekemisestä, kentän tulisi sisältää useita vaihtoehtoisia reittejä ja ratkaisumalleja ongelmiin. Kentät, jossa on vain yksi reitti tuntuvat helposti putkijuoksulta ja käyvät nopeasti tylsiksi. (Mapping 101: Gameflow 2006.)

3.4 Rytmitys

Pelisuunnittelijat näkevät pelin usein kokonaisuutena eivätkä pohdi paljonkaan pelin yksittäisiä kohtia. Kenttäsuunnittelija sitä vastoin kiinnittää huomionsa jokaiseen hetkeen, jonka pelaaja viettää hänen suunnittelemassaan kentässä. Pelirytmitys koostuu tapahtumista ja suvannoista. Hyvin suunnitellussa kentässä on jouheva, jännitteistä ja rennommista hetkistä koostuva jatkumo, joka vetoaa pelaajaa etenemään pelissä. Tekniikka on tuttua elokuvista ja kirjallisuudesta; toiminta johtaa jännitykseen, joka ei heti häviä ja pelaajalle täytyy antaa mahdollisuus levähtää toiminnan välillä. (Feil, Scattergood 2005, 14.)

Lineaarisissa peleissä rytmin kontrolloiminen on helppoa: avoimissa tai muuttuvissa peliympäristöissä täydellisen kontrollin aikaansaaminen on lähes mahdotonta. Tällaisissa peleissä kentät ovat sarja tapahtumia ja kokemuksia huolimatta siitä, määrääkö kenttäsuunnittelija niiden järjestyksen vai päättääkö pelaaja sen. Rytmiä ja pelaajan kiinnostusta kontrolloidaan rajoittamalla tai vaihtelemalla tarjottavia tapahtumia. (Byrne, 2004 67-68.)

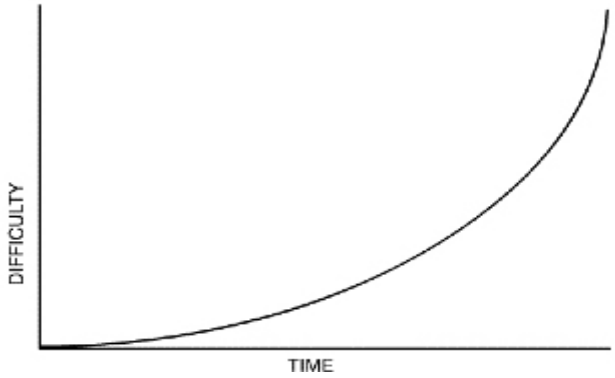
Jännityksen luomiseksi täytyy kentän tarjota erilaisia kokemuksia: taukoja, yllättäviä elementtejä ja kauhumaisia pitkiä käytäviä. Jos koettelemukset ovat liian irrallisia tai niitä on liian vähän, menettää pelaaja mielenkiintonsa. Kenttäsuunnittelijan tulisikin pystyä ennustamaan tunneskaala, jota pelaaja käy läpi edetessään pelissä ja tämän mukaan päättää miten rytmittää kenttänsä jotta pelaajan mielenkiinto säilyisi. (Byrne, 2004 67-70.)

Kenttäsuunnittelijalla on lukuisia erilaisia tapoja, joilla vaikuttaa pelirytmiin. Tärkeää on hahmotella rytmitys etukäteen ennen kentän työstövaihetta. Toisaalta suunnitelma kannattaa pitää melko avonaisena, sillä käytäntö osoittaa usein pelirytmin toimivuuden. Jos pelirytmi on lyöty lukkoon jo suunnitteluvaiheessa, voi sen muuttaminen olla vaikeaa enää myöhemmissä työstövaiheissa. Kannattaakin kiinnittää erityistä huomiota lähinnä esim. kentän avainkohtauksiin ja käännekohtiin. Lopullinen kentän rytmitys luo kentän pelattavuuden. Ilman kunnollista rytmiä pelaaja tylsistyy, liian jyrkillä rytmin muutoksilla pelaaja uupuu kesken kentän. (Byrne, 2004 67-69.)

3.5 Vaikeustaso

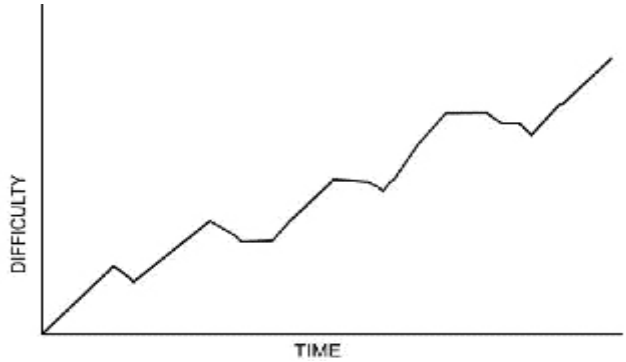
Pelin vaikeustason manipulointi on eräs kenttäsuunnittelijan tärkeimmistä työkaluista. Jokaisella pelillä ja pelikentällä on muuttuva vaikeustasonsa. Vaikeustason ei tulisi olla liian vaativa eikä liian helppo. Kuten pelirytmin ja kuljetuksen tulisi myös vaikeustason olla vaihteleva mielenkiintoisen pelikokemuksen takaamiseksi. (Byrne, 2004 73-74.)

Vaikeustasosta voidaan piirtää yksinkertainen kuvaaja(kuva 2.), äärimmäisessä tapauksessa se on hyvin yksiselitteinen viiva asteikon alavasemmalta yläoikeaan. Tämänkaltainen vaikeustaso ei käytännössä olisi kovinkaan mielenkiintoinen, eikä se usein toteudu pelien interaktiivisen luonteen vuoksi. (Byrne, 2004 73.)



KUVA 2. Yksinkertainen vaikeustasonkasvun kuvaaja (Byrne, 2004)

Kuvassa 3. on paljon parempi kuvaaja pelien vaikeustasosta. Siinä nähdään pienempien alueiden vaikeustason kasvu, kun pelaajaa haastetaan enemmän ja enemmän. Kuvaajan kuopissa pelaajan kyvyt ylittävät pelin tarjoamat haasteet. Nämä kohdat ovatkin hyviä paikkoja uusien pelihahmokykyjen tai pelielementtien esittelyyn: uusi elementti haastaa pelaajan jälleen opettelemaan uuden pelityylin, mikä vastaavasti jälleen nostaa vaikeustasoa. Käytännössä useiden pelien vaikeustasot näyttävät kuvan 3. kaltaisilta. (Byrne, 2004 73-74.)



KUVA 3. Normaali vaikeustasonkasvun kuvaaja (Byrne, 2004)

Joissain peleissä on käytössä dynaaminen vaikeustasonsäätely: peli määrittelee itse vaikeustasonsa riippuen pelaajan taidoista. Jos peli havaitsee pelaaja kuolevan liian usein, helpotetaan vaikeustasoa. Vastaavasti jos pelaaja on taitava, peli nostaa vaikeustasoa esimerkiksi nostamalla vihollisten elämäpisteitä tai lukumäärää. Systeemin tarkoituksena on sekä ehkäistä turhautumisen tunne, joka pelaajalle tulee vaikeustason epäsopivuuden takia, että antaa jokaiselle pelaajalle samankaltainen pelikokemus riippumatta pelaajien tasoeroista. Tietokoneen ohjaama vaikeustaso ei ole täysin autonominen, vaan kenttäsuunnittelijan tulee silti määritellä kenttään perusvaikeustaso ja rajat, joiden sisään perusvaikeustasoa voidaan venyttää. (Byrne, 2004 74.)

3.6 Pelaajan ohjaus

Pelaajan ohjaus on tärkeä kenttäsuunnittelun osa-alue. Pelaaja etenee kentissä A pisteestä, B pisteen kautta C pisteeseen. Joskus pelaaja kuitenkin hämmentyy eikä löydäkään reittiä A pisteestä B pisteelle. Tällöin pelaaja turhautuu ja ohjaus on epäonnistut. Parhaimmillaan ohjaus on täysin näkymätön osa pelikenttää ja pelaaja etenee kentässä luonnollisesti. Valve on kehittänyt Source-pelimoottoriin ominaisuuden, joka tarkkailee pelaajien edistymistä ja lähettää tästä sitten tiedon kehittäjälle. Tämän tiedon pohjalta pelikenttiä voidaan muokata julkaisun jälkeen päivitysten yhteydessä. Esimerkiksi liian vaikeiksi osoittautuneita kohtauksia voidaan helpottaa ja pelikuljetusta sekä pelaajan ohjausta voidaan parantaa.

Oviaukkojen, kulkureitien, rakojen ja käytävien joita pelaajan on tarkoitus käyttää, tulisi olla mahdollisimman helposti huomattavissa. Pääsääntönä voidaan pitää sitä, että pelaajan tulisi nähdä uloskäynti sillä hetkellä, kun hän astuu uuteen tilaan. Pelaajalle täytyy jatkuvasti olla selvää, miten kentässä edetään tai ainakin nähdä reitti eteenpäin. Tämä on yksi pääsäännöistä hyvän pelinkuljetuksen ja pelaajan ohjauksen takaamiseksi. (Johnston 2005.)

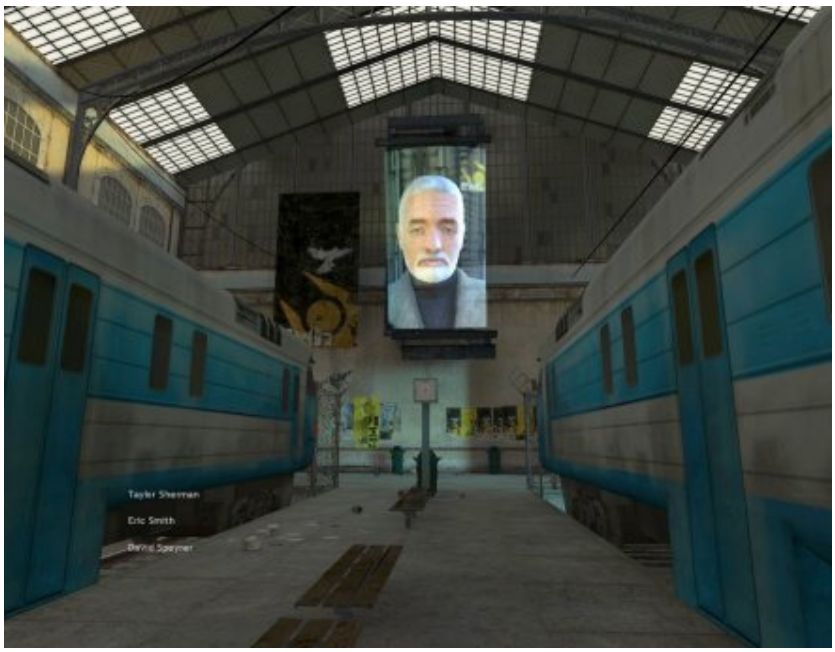
Pelaajan eksymistä ja siten turhauttamista tulee välttää kaikin keinoin. Yleisin syy, miksi pelaaja luovuttaa pelaamisen, johtuu huonosta ohjauksesta. Pelaaja pääsee yleensä yli muutamasta eksymisestä, mutta jos harhailu ja eksyminen toistuvat usein, pelaaja yleensä luovuttaa nopeasti. On elintärkeää hyvän pelattavuuden ja pelikuljetuksen kannalta, että pelaaja pystyy navigoimaan kentässä ongelmitta. (Johnston 2005.)

3.61 Tavoite

Kentän tavoite eli objektiivi on ensimmäinen asia, joka kenttäsuunnittelijan tulee päättää alkaessaan suunnitella kenttää. Paraskaan kenttä ei toimi ilman pelaajalle annettua objektiivia. Ilman objektiivia pelaaja vain vaeltaa päättömästi kentässä ja jättää nopeasti pelin kesken. Kentän ensimmäisestä hetkestä lähtien pelaajan tulee tietää, mitä hänen tulee tehdä. Ei ole väliä, onko kenttä monin- vai yksinpelattava, jokaisen kentän tulee tarjota pelaajalle jotain tehtävää. Objektiivin päättäminen ei ole vaikea osa suunnittelua, mutta se, miten se tuodaan pelaajan tietoon, on usein monimutkaisempaa. Usein yksinpeleissä pelaajan objektiivit kerrotaan pelaajalle kentän alussa, joko NPC-hahmon kautta tai yksinkertaisesti tekstinä tai puheena pelin taustalla. Tärkeää on pitää huoli siitä, että pelaaja tajuaa viestin. Esimerkiksi Half-Life 2 pelissä pelaajaa usein ohjeistettiin kesken tulitaistelun, jolloin objektiivi jäi usein arvailun varaan. Moninpeleissä kentän objektiivi on paljon yksinkertaisempi ja helpompi ymmärtää, sillä usein pelattavuus on pelin objektiivi. Pelaajaa usein ohjeistetaan lyhyellä tekstillä, jossa kerrotaan kentän tyyppi ja voittoehdot. (Duarte 2005.)

3.62 Huomiopiste

Huomiopiste on yksinkertaisesti piste, johon kenttäsuunnittelija haluaa pelaajan katseen kohdistuvan. Huomiopiste voidaan luoda väreillä, äänillä, muodoilla tai valoilla. Esimerkiksi värein luotu huomiopiste muodostetaan kontrastin tai korostuksen keinoin: kirkas valo pimeän käytävän päässä tai vihreä laatikko punaisessa huoneessa. Nyrkkisääntönä voidaankin pitää sitä, että kaikki mikä erottuu ympäristöstään, houkuttaa pelaajaa, eli muodostaa huomiopisteen. Huomiopisteiden kanssa täytyy kuitenkin olla varovainen, väärin tai epähuomiossa asetettu huomiopiste voi johtaa pelaajaa harhaan. (Valve Developer community 2008.)



KUVA 4, Half-Life 2, huomiopiste-esimerkki. (Valve 2002)

3.63 Visuaaliset vihjeet

Visuaalisiin vihjeisiin perustuvien huomiopisteiden luominen realistiseen ympäristöön voi olla ongelmallista. Oikeassa elämässä ei aukinaista ovea tai oikeaa reittiä ole valaistu tai värjätty. Mutta oikeassa elämässä ihmisellä on myös kärsivällisyyttä ja aikaa etsiä oikea reitti. Peleissä tätä ylellisyyttä ei kuitenkaan ole, pelaajan tulisikin löytää oikea ovi tai reitti vaivatta, mutta visuaalinen vihje ei saisi kuitenkaan erota liikaa ympäristöstään. Yleensä reitin erilainen valaistus ja sijoittelu riittävät. Jos oikea oviaukko on valaistu hieman eri tavalla ja umpikujat jätetään varjoon, pelaaja tajuaa, mitä reittiä hänen tulisi käyttää. Jotkut visuaaliset vihjeet ovat yksinkertaisia mutta toimivia. Eräs jo Doomin aikoihin kehitetty keino on vieläkin toimiva: Pelaajat yleensä yrittävät seurata reittejä, jotka jakavat samankaltaisen maatekstuurin. Suuri muutos maatekstuurissa viestii yleensä esteestä tai siitä, että pelaaja on eksynyt reitiltä. Alitajuisesti pelaaja ajattelee, että seuraamalla tietyn väristä reittiä hän etenee vääjäämättä kohti kentän objektiivia. (Johnston, 2005.)

Näkyvä maamerkki yleinen tapa ohjastaa pelaajaa. Maamerkki voi näkyä kentän taustalla tai sijaita kartassa pelattavana elementtinä, jolloin pelaaja tietää koko ajan mihin suuntaan hänen tulisi edetä. Half-Life 2 peli on tästä hyvä esimerkki: lähes koko pelin keston ajan pelaaja näkee kartan reunalla majesteettisen Citadel-rakennuksen siluetin. Toinen paljon käytetty tapa on sijoittaa ohjeita itse kenttään, esimerkiksi Counter-Strike pelissä pelaajaa ohjastavat kentän seiniin maalatut ohjeet ja suun-

tanuolet. Jo yksinkertainen nuoli ja kirjain riittävät ohjeistamaan pelaajaa siitä miten hänen tulisi kartassa edetä. (Duarte 2005.)

KUVA 5, Counter-Strike Source, yksinkertaista pelaajan ohjastamista. (Valve 2002)

3.64 Ääni

Ääni on pitkään ollut peleissä toisarvoisessa asemassa. Tilanne on paranemassa, sillä nykyajan äänikortit ja konsolit ovat mahdollistaneet muun muassa 5.1-surround äänen peleihin. Äänen pelaajan immersioon tuoma lisä on suuri.

Äänillä on tärkeä tehtävä tunnelmanluojina, mutta niitä voidaan myös käyttää pelaajanohjauksessa. Yksinkertaisimpana esimerkkinä ovat NPC tai vihollishahmojen keskustelut, joita pelaaja voi salakuunnella. Pelaajalle voidaan näin antaa tietoa objektiivista tai määränpäästä. Hahmoäänet ovat myös tärkeä apuväline pelaajalle eri vihollisten tunnistamiseen ja paikantamiseen. Pelaajan tunnetiloihin voidaan myös vaikuttaa esimerkiksi kauhupelien hyytävillä kirkaisuilla tai ukkosen jyrinällä. (Duarte 2005.)

3.7 Pulmat

Pulmien eli puzzlejen rakentaminen on haastavaa, pelitestaus on oleellinen osa pulmien rakentamista. Kenttäsuunnittelijan ei tulisi koskaan itse testata rakentamaansa pulmaa, vaan hänen täytyisi antaa jonkun ulkopuolisen testata kohta. Kenttäsuunnittelija tuntee yleensä työstämänsä kentän läpikotaisesti, jolloin erittäinkin vaikea tai epälooginen pulma voi tuntua hänestä aivan toimivalta. Pulmatyyppejä on useita, ne voidaan luokitella karkeasti kolmeen tyyppiin: loogisiin, taito ja konfliktipulmiin. Yleensä ajatellaan että pulmat koskevat vain yksinpelejä, mutta niitä esiintyy myös moninpeleissä.

Looginen pulma on yleensä hauskin pelaajalle, sillä se saa hänet tuntemaan itsensä viisaaksi. Toisaalta loogiset pulmat ovat myös vaikeimpia rakentaa. Maalaisjärjellä ja ahkeralla ulkopuolisella testauksella loogisten pulmien rakentaminen helpottuu.

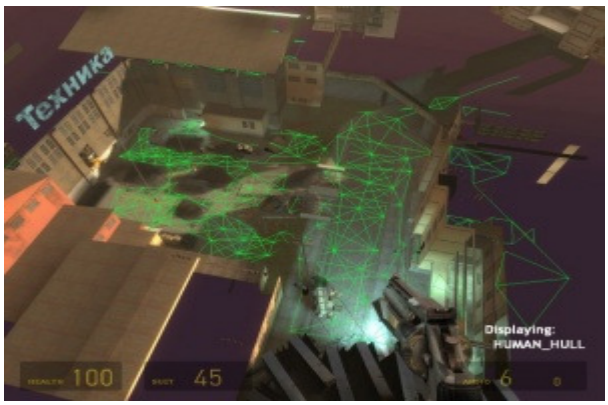
Taitoon perustuva pulma voi olla esimerkiksi vaativa tasohyppelykohtaus tai uuden esineen käytön opetteleminen. Tärkeintä tämäntyylisissä pulmissa on se, että pelaaja, tietää mitä hänen tulee tehdä ja että hän pystyy suoriutumaan pulmasta. Uusia pelielementtejä esitellessä pelaajalle täytyy antaa mahdollisuus uuden pelitavan sisäistämiseen. Tämän jälkeen pelaajan voi haastaa käyttämään juuri oppimaansa taitoa taitopulman ratkaisuun.

Kolmas pulma tyyppi on konflikti, eli ”hoitele vihollinen” -tyylinen pulma. Tämän tyylinen pulma on yleisin pelipulma, jos peli on räiskintäpeli ei peliä olisi ilman konflikteja. Konfliktipulmaa rakentaessa tulee kenttäsuunnittelijan miettiä seuraavaa kolmea asiaa: mitä aseita käytetään, millaisessa tilassa taistelu käydään ja millainen vihollinen pelaajaa vastaan asetetaan. (Duarte 2005.)

3.8 Tekoäly

Tekoäly eli AI kontrolloi lähes kaikkea pelihahmojen käyttäytymistä. Source-pelimoottorin tekoäly pystyy toimimaan täysin ilman mitään apukeinoja, mutta tekoälyä voidaan auttaa erilaisilla työkaluilla. Tärkein tekoälyn navigoinnin kannalta on node-kartta (kuva 6.), koska pelikenttä on yleensä lähes staattinen, voidaan pelialueeseen laskea ennalta niin sanottu ”ajokartta”. Node-kartta kertoo tekoälylle suorimat reitit pisteestä A pisteeseen B. Source-pelimoottorissa voidaan myös käyttää vihjemuuttujia, ne antavat tekoälylle vihjeitä siitä missä on hyvä suojautua ja esimerkiksi ladata asetta. Näiden lisäksi on joukko muita tekoälytyökaluja, joiden kaikkien käyttö on tärkeää hyvän tekoälyn ja haastavan vihollisen luomisessa. (Valve Developer community 2008.)

Kun vihollisia tai muita hahmoja asetetaan kenttään, on tärkeää muistaa varsinkin realistista peliä tehdessä, että oikeassa elämässä kaikella on jokin syy olla siinä, missä se on. Hyvään kenttäsuunnitteluun kuuluu, että vihollisten asemaan johtaa jokin reitti, mistä vihollinen on päässyt asemaansa ja että vihollisen asema on järkevä. Esimerkiksi vahvasti aseistetun tukikohdan porteilla ja vartiotorneissa on luultavasti vartioita ja tämän lisäksi osa tukikohdan miehityksestä voi olla esimerkiksi parakeissa. (Feil, Scattergood 2005, 105.)



KUVA 6, Node-kartta esimerkki. (Valve 2005)

3.9 Skriptaus

Skripteiksi kutsutaan pieniä ohjelmia, jotka antavat kenttäsuunnittelijalle hallinnan pelikentän tapahtumiin. Skripteillä kenttäsuunnittelija voi vaikuttaa esimerkiksi tekoälyn käyttäytymiseen. Kenttäsuunnittelijan on tärkeää osata ainakin jonkin verran ohjelmointia. Toisaalta skriptikieli on yleensä varsin yksinkertaista: lukutaidolla ja perusmatematiikalla pärjää pitkälle.

Skriptauksella kenttä nivotaan tavallaan yhteen, pelikentän eri elementit, kuten triggerit, viholliset, erikoisefektit ja ansat saadaan toimimaan yhdessä skriptauksen avulla. Pelien skriptaustasoissa on eroja. Farcry pelissä scriptaus on rajoittunut lähinnä pääobjektiiveihin pelin avoimen luonteen vuoksi, mutta esimerkiksi Neverwinter Nights peli on lähes kokonaan skriptattu. (Feil, Scattergood 2005, 120.)

4 TEKNOLOGIAA

4.1 Pelimoottori

Pelimoottori on runko-ohjelma, joka yhdistää pelimoottorin eri ominaisuudet toimivaksi kokonaisuudeksi, eli pyörittää peliä. Ilman pelimoottoria peli on vain kokoelma erilaisia ohjelmia, jotka vastaavat vain omasta osa-alueestaan. Pelimoottori koostuu ohjelmapalasista, kuten fysiikkamoottorista, törmäystarkistuksesta, tekoälystä, verkko-, ääni-, skripti- ja animaatiomoottoreista. (Korva 2002.) Ohjelmoijat ja teknikot kokoavat yhteen moottorin, joka toimii kohdealustalla ja sisältää työkalut artisteille, suunnittelijoille ja ääniteknikoille, jotka saavat tuotua peliin tuotetun sisällön. Pelimoottorin piirtomoottori on yleensä pääkoodi, ja se tuottaa pelin visuaalisen puolen. (Byrne, 2004 47.)

Usein pelimoottori tukee valmiiksi useita eri pelialustoja, jotta peli voitaisiin julkaista mahdollisimman monella eri alustalla. Eri pelialustoja ovat esimerkiksi pelikonsolit ja erilaiset käyttöjärjestelmät. (Wikipedia 2008.)

Pelimoottoreita on useita, osa pelimoottoreista on rakennettu lisensointia varten, kuten Source ja Unreal Engine pelimoottorit (Kuvat 7 & 8). Pelifirmat, joilla ei ole yrityksen sisäisiä resursseja rakentaa omaa pelimoottoria, lisensoivat valmiin pelimoottorin ja rakentavat pelinsä jo valmiille pohjalle tarjotuilla työkaluilla. Usein pelifirmat joutuvat muokkaamaan pelimoottoria omiin tarpeisiinsa, toisaalta tällöin lisensoidessa saavutettu resurssisäästö käy kyseenalaiseksi. Osalla pelifirmoista on omat yrityksen sisäiset pelimoottorit, jotka on rakennettu alusta lähtien yrityksen tarpeisiin. Hyvänä esimerkkinä esimerkiksi Remedy Entertainmentin MaxFX pelimoottori, jonka Remedy rakensi Max Payne -pelisarjaa varten.

KUVA 7, Unreal 3.0 pelimoottori (Epic 2005)

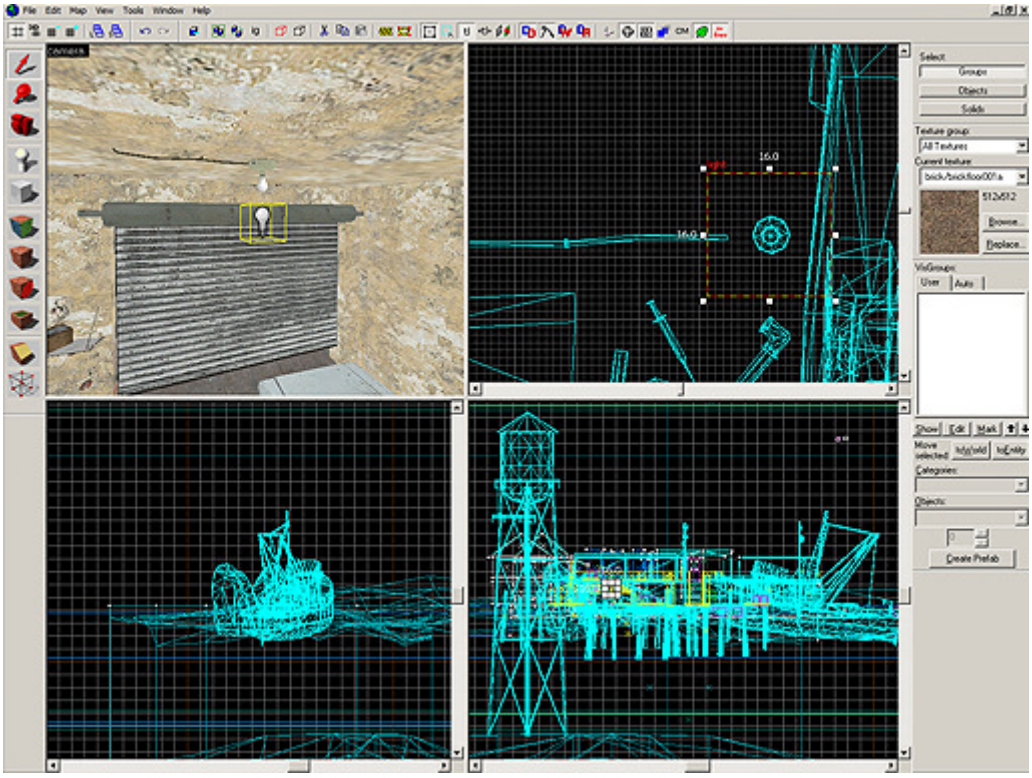
KUVA 8, Source-pelimoottori. (Valve 2007)

4.2 Kenttäeditori

Kenttäeditori on työkalu, jolla pelin tekniikasta tietämätönkin henkilö voi luoda sisältöä tai tuoda peliin sisältöä jostain muusta ohjelmasta. Jotkut editorit ovat hyvin yksinkertaisia, eivätkä ne esimerkiksi sisällä mahdollisuuksia esikatseluun, jolla kentän todellista ulkonäköä voisi nopeasti tarkastella. Tällöin kentät täytyy joka kerta erikseen kääntää pelin tukemaan muotoon, jota pelimoottori ymmärtää.

Jotkut editorit ovat pitkälle kehittyneitä ohjelmia, jotka vastaavat toiminnoiltaan lähes kaupallisia mallinnus- ja ohjelmointiohjelmia. Tällaisista editoreja ovat esimerkiksi UnreadEd ja Hammer.(Kuva 9.) Jotkut editorit antavat suunnittelijoilleen paljon työkaluja kentäntekoon, aina tekoälyn skriptaamisesta, reaaliaikaiseen kentän visuaalisen ilmeen muokkauk-

seen asti. Peliprojektissa käytettävä editori ja pelimoottori määräävät kehittäjien työskentelytavat ja tahdin.
(Byrne, 2004 47.)



KUVA 9, Hammer-kenttäeditori

4.3 Source-pelimoottori

Valven Source-pelimoottori on laajalti lisensoitu, ja Valve käyttää pelimoottoria myös omissa peliprojekteissaan. Sourcen katsotaan olevan yksi monipuolisimmista, joustavimmista, helpoiten muokattavista ja tehokkaimmista pelimoottoreista. Source-pelimoottori on myös tunnettu hyvästä skaalautuvuudesta, eli moottori osaa skaalata pelin hyvin laajalle kirjolle eri tehoisia kokoonpanoja. (Valve developer community 2008.)

Vaikka Source-pelimoottori ei grafiikassa edustakkaan aivan alan terävintä kärkeä, ovat helppokäyttöisyys, muokattavuus ja työkalujen, kuten animaatio ja verkko-ominaisuudet aivan omaa luokkaansa. Moottori taipuu myös mitä erilaisimpiin grafiikkatyyleihin helpommin kuin esimerkiksi Unreal Engine tai CryEngine2.

4.31 Työkalut

- Valven Hammer-kenttäeditori
- Plugin ohjelmat suosittuihin 3D-mallinnusohjelmiin, kuten Maya ja Lightwave 3D
- Faceposer, kasvoanimaatio työkalu puheen ja ilmeiden animointiin.
- Source model viewer, 3D-mallien esikatseluohjelma, malli nähdään suoraan sellaisena kuin se näkyy pelissä
- Partikkelieditori
- Materiaalieditori
- Kommentointieditori, työkalu jolla voidaan lisätä pelikenttään tekijöiden kommentteja
- Suorituskyvyn mittaustyökalut
- Half-Life 2, Team Fortress 2 ja Portal pelien täydellinen lähdekoodi

(Valve developer community 2008.)

4.32 Tekniset ominaisuudet

- LOD tuki objekteille ja ympäristölle.
- Valmis shader 3.0 kirjasto, voidaan käyttää Valven kehittämiä shadereitä suoraan. Esim. Team Fortress 2 phong shading.
- Dynaamiset ja staattiset valot. Mm. HDR, dynaamiset varjot ja materiaalien valaisuominaisuudet.
- Erikoisefektit mm. motion blur, volumetrinen savu, realistinen vesi.
- Peliympäristöt mm. maasto, BSP-brush ja staattiset objektit
- Materiaalit mm. bumb-map, yksityiskohtatekstuuri ja multitekstuurit.
- Animaatio mm. huulisynkronointi, luurankojärjestelmä, kasvoanimaatio ja lihassimulointi.
- Fysiikkamoottori, mm. ajoneuvot, köydet, inverse kinematic ja ragdoll.
- Tekoäly mm. Reitinhaku, älykäs interaktio, aistijärjestelmä.
- Ääni mm. Surround tuki, 3D-ääni ja materiaaliasetukset.
- Verkko-ominaisuudet, mm. ennustava verkkokoodi, serveriselain ja viestit
- Multi-core tuki uusille pc- ja konsoliprosessoreille.

(Valve developer community 2008.)

4.4 Hammer-kenttäeditori

Valven Hammer on Source-pelimoottorin kenttäeditori. Alun perin Hammer-kenttäeditori oli nimeltään Worldcraft. Worldcraft oli Quake ja Quake II peleille suunnattu editori. Hammer-kenttäeditorin voi ladata ilmaiseksi Valven Steam palvelun kautta.

Hammeria on kritisoitu sen vanhanaikaisen käyttöliittymän vuoksi, käyttöliittymä on yhä pitkälle sama kuin se oli vuonna 1998, kun Half-Life ilmestyi ja kun ensimmäinen versio julkaistiin. Silti Hammeria pidetään yhtenä käyttäjäystävällisimmistä ja tehokkaimmista kenttäeditoreista. Viimeisin versio Hammerista (Hammer v4.1) julkaistiin marraskuussa 2007, noin kuukausi Valven Orange Box- pelipaketin ilmestymisen jälkeen. Uusin versio lisäsi tuen Portal-pelille ja valojen esikatselumahdollisuuden. Nyt kenttää ei tarvitse käydä tarkastelemassa pelissä joka kerta, kun valaistuksen muutoksia halutaan tarkastella. (Wikipedia 2008.)

5 GRAFIIKKA

5.1 Pelattavuuden ja grafiikan korrelaatio

Hyvä kenttä ei voi koostua pelkästään hyvästä grafiikasta tai pelattavuudesta. Vain näiden yhdistelmä ja eri aspektien hyvä harmonia voi luoda hyvän pelielämyksen. Grafiikan on tarkoitus tukea pelattavuutta ja pelattavuuden on tarkoitus tukea grafiikkaa. Pelottava pelielämys tarvitsee pelottavan peliympäristön, samoin visuaalisesti pelottava pelimaailma tarvitsee toimiakseen pelottavan pelisysteemin. Nykyään, kun kentän pelillisestä ja graafisesta kenttäsuunnittelusta vastaavat yleensä eri suunnittelijat, tämä vuorovaikutus usein unohtuu. Pelattavuutta suunnittelevalla kenttäsuunnittelijalla on vähän tietoa grafiikasta ja grafiikasta vastaavalla kenttäsuunnittelijalla vastaavasti vähän tietoa pelattavuudesta. Tästä seuraa se, että kummatkin suunnittelijat toteuttavat oman visionsa ja työskentelevät erillään, sen sijaan, että he työskentelisivät yhdessä. Suunnittelijoiden tulisikin kiinnittää erityistä huomiota yhteistyöhön. (De Jong 2004, 10.)

Yleensä pelaaja näkee kentän ennen, kuin hän kuulee tai pelaa sitä. Monet pelit alkavat esinäytöksellä, joka esittelee pelin tekniikkaa ja grafiikkaa, antaen pelaajalle kuvan siitä mitä peli tulee tarjoamaan. Pelin visuaalinen tarjonta on usein se elementti, joka koukuttaa pelaajan aivan pelin alussa. Sen jälkeen täytyy pelattavuuden kuitenkin nousta esiin ja pitää pelaaja otteessaan, muuten hienoinkaan visuaalinen esitys ei auta. Pelien historiassa oli aika, jolloin maailman rumin peli pystyi menestymään pelkän hyvän pelattavuuden avulla. Tätä ei nykyään enää juurikaan tapahdu, pelaajat vaativat yhä parempaa grafiikkaa ja odottavat jokaisen pelin käyttävän uuden sukupolven graafista suorituskykyä täysin hyväkseen. Tämä ei kuitenkaan estä hieman huonommilla grafiikoilla varustetun pelin suosiota, hauska peli myy huonoista grafiikoista huolimatta, mutta paremmilla grafiikoilla peli olisi luultavasti vielä suositumpi. Hienot kuvankaappaukset pelipaketin takakannessa voivat vaikuttaa hyvinkin paljon satunnaispelaajan ostopäätökseen. (Byrne 2004, 240-241.)

5.2 Geometria

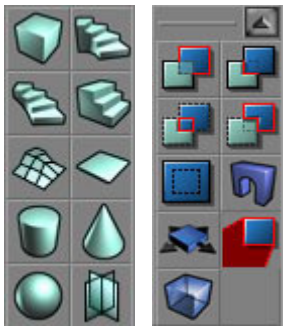
Moderneissa 3D-peleissä kenttägeometria muodostuu suurimmaksi osaksi staattisista muodoista. Aikaisemmin kentät tehtiin lähes kokonaan BSP-primitiiveillä, (BSP brush) joko kaivertamalla tai lisäämällä peliavaruuteen BSP-primitiivien muotoisia tiloja. Nykyään primitiivejä käytetään

vain kentän perusmuodon luomiseksi ja tilat täytetään erikseen mallinnusohjelmissa tehdyillä, paljon monimutkaisimmilla, staattisilla objekteilla. (Unreal Developer Network 2007.)

KUVA 10. Kenttä muodostuu kokonaan staattisista muodoista, koko tilassa on vain 8 normaalia polygonia. (Epic Games 2007)

5.21 Primitiivit

Nykyään kentät voidaan rakentaan lähes kokonaan ilman BSP-primitiivejä. Tarvitaan kuitenkin ainakin yksi BSP-primitiivi, jolla ”leikataan” peliavaruudesta tila, jossa kenttä sijaitsee. Voidaan ajatella että peliavaruus on kiinteätä ainetta, johon BSP-primitiivillä kaiverretaan tila, jonka sisälle kenttä rakennetaan. Kuvan 11. työkalukuvakkeita tarkastelemalla nähdään, että BSP-primitiivien käyttötarkoitus on ollut erilainen. Aikaisemmin BSP-primitiivit olivat ainut tapa luoda geometriaa. Nykyään BSP-primitiivejä käytetään lähinnä kenttien perusmuotojen, taustataivaslaatikoiden ja erikoisalueiden muodostamiseen. Suurin osa kenttien geometriasta rakennetaan staattisilla objekteilla. (Unreal Developer Network 2007.)



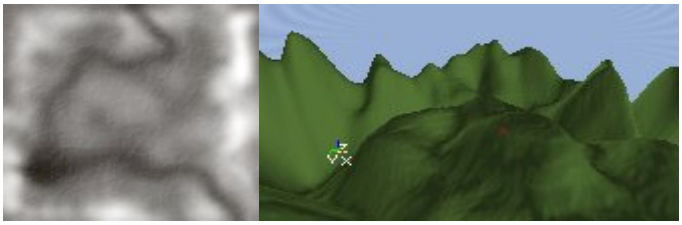
KUVA 11. UnrealEd-kenttäeditorin BSP-työkalut (Unreal Developer Network 2007)

5.22 Staattinen objekti

Staattinen objekti piirretään kokonaan näytönohjaimen resursseilla. Niissä ei ole BSP-järjestelmän ongelmia, kuten vuotava geometria tai huono optimoitavuus. Staattiset objektit vaativat paljon vähemmän laskentaresursseja sekä voivat olla huomattavasti yksityiskohtaisempia. Staattiset muodot eivät ole nimensä mukaisesti aivan staattisia, niitä voidaan liikuttaa, kääntää, uudelleen teksturoida ja skaalata reaaliajassa. Niitä ei voi kuitenkaan animoida, koska objektien verteksien relatiiviset asemat on varastoitu näytönohjaimen muistiin, tämä estää objektin animoinnin. Koska staattiset objektit varastoidaan näytönohjaimen muistiin kerralla – sen sijaan, että ne laskettaisiin ”kerran joka kuvassa” – usean staattisen objektin piirtäminen on suhteellisen yksinkertaista. Kun staattinen objekti piirretään ruudulle, pelimoottorin tarvitsee vain kertoa näytönohjaimelle objektin sijainti, koko, kulma ja mitä tekstuuria objekti käyttää. (Unrealwiki, 2008.)

5.23 Maasto

Pelien maasto voidaan luoda joko mallinnusohjelmassa tai kenttäeditorissa. Nykyään yhä useammassa kenttäeditorissa on omat maastotyökalut. Kenttäeditorilla luodulla maastolla on etuna suora yhteensopivuus pelimoottorin kanssa, mallinnusohjelmassa luotu maasto voi esimerkiksi olla liian raskas kyseiselle pelimoottorille. Yleisin tekniikka maaston luomiseen on korkeuskartta, joka on yksinkertainen harmaasävykuva maaston korkeudesta. Korkeusasteikkoa voidaan myöhemmin helposti skaalata. (Development, Premier. *Beginning Game Level Design.* 39.)



KUVA 12. Korkeuskartta ja sillä luotu maasto (Unreal Developer Network 2007)

Maasto muodostuu polygoneista, mitä vähemmän polygoneja sitä vähemmän maastossa on geometrisia yksityiskohtia. Teksturoinnilla maastoon luodaan vaihtelua, Unreal pelimoottorissa maasto teksturoidaan editorin maalaustyökaluilla, tekstuuri tavallaan maalataan maastogeometrian päälle. Useita tekstuureita voidaan pinota päällekkäin tasoiksi, näiden tasojen läpinäkyvyyttä säätelemällä voidaan tasoista sekoittaa realistisen näköinen maasto. (Unreal Developer Network 2007.)

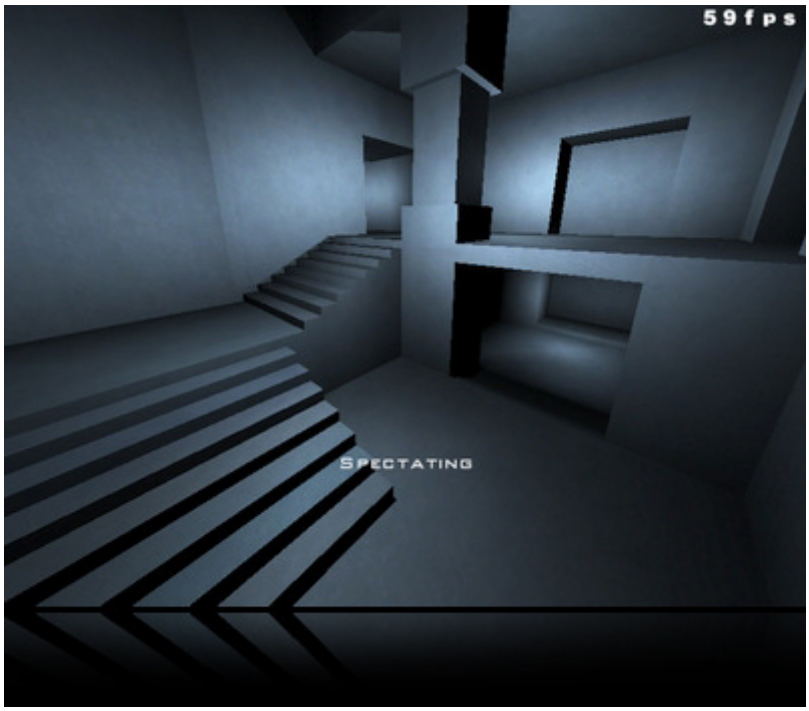
5.24 Optimointi

Jokainen objekti, tapahtuma ja kaikki kentän elementit hidastavat osaltaan peliä. Kentän optimointi on usein vaikeaa ja monimutkaista. Toisaalta jo yksinkertaisinkin optimointi voi tuoda huomattavasti lisää frameratea ja laskea kartan esilaskenta-aikaa. Ideaalitapauksessa kentän optimointia aletaan suunnitella jo esisuunnitteluvaiheessa. (Valve developer community 2008.)

Optimointi on pähkinänkuoressa pelimoottorin ohjeistamista hyötysuhteen parantamiseksi. Pelimoottorille kerrotaan, mitä sen pitää laskea ja mitä ei. Source-pelimoottorissa optimointiin tarjotaan kahta hyödyllistä työkalua: Showbudget ja Mat_wireframe. Showbudget näyttää, mitkä elementit kentässä vievät minkin verran konetehoja. Mat_wireframe näyttää kaiken, minkä pelimoottori milläkin hetkellä piirtää. Yleensä pelimoottori yrittää piirtää liikaa asioita, tällöin pelimoottoria ohjastetaan rajaamalla käsin piirtovyöhykkeitä. (Valve developer community 2008.)

5.3 Tekstuurit

Tekstuuri on kuvankäsittelyohjelmassa luotu 2D-kuva, jonka pohjana usein käytetään valokuvattua tai skannattua lähdemateriaalia. Tekstuuri lisää objektin tai pinnan yksityiskohtia sekä vahvistaa muotoja. Tekstuuriin piirretyt yksityiskohdat, kuten vaikka naulat puutekstuurissa, lisäävät pintaan vaikutelman lisägeometriasta, vaikka informaatio on pelkästään kaksiulotteista. Tekstuureilla luodut yksityiskohdat ovatkin hyvä tapa säästää konetehoa ja lisätä kentän yksityiskohtia. Vaikka tekstuureilla luodut yksityiskohdat eivät vastaa todellista geometriaa, tuntuvat ne pelin tiimellyksessä oikeilta, sillä pelaajalla ei useinkaan ole aikaa tutkiskella kentän yksityiskohtia lähemmin. Suurin osa visuaalisesta ympäristöstä luodaan valaistuksella ja teksturoinnilla. Vaikka perusgeometria olisi kuinka monimutkainen tahansa, vasta teksturointi ja valaistus tekevät kentästä valmiin. (Byrne 2004, 243.)



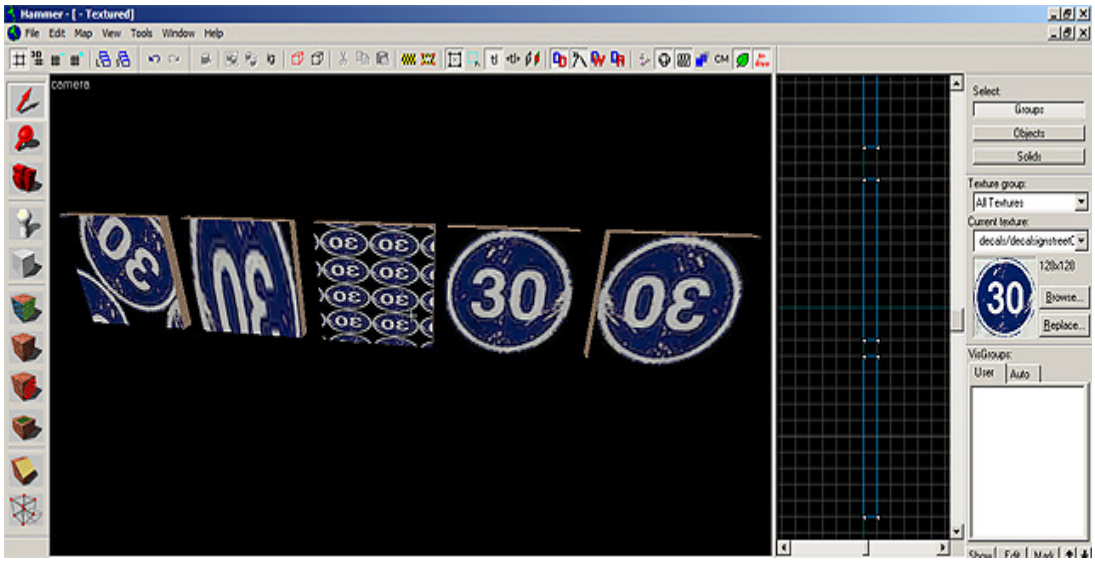
KUVA 14. UT2K3 perusgeometria. (Ostretsov 2004)

KUVA 15. UT2k3 viimeistelty kenttä (Ostretsov 2004)

Teksturointi kannattaa aloittaa heti, kun kentän perusgeometria on valmistunut. Tässä vaiheessa kenttään on vielä kohtalaisen helppo tehdä muutoksia. Teksturoinnilla voidaan myös piilottaa tai korjata geometrian virheitä. Teksturointi on – kuten kaikki kenttäsuunnittelussa – monivaiheinen prosessi, eikä kentän teksturointia ole tarkoitus saada lopulliseksi ensimmäisellä kerralla. Usein kentän teksturointi saadaan valmiiksi vasta valojen ja esivalmistettujen objektien sijoittelun viimeistelyn jälkeen. Teksturoidessa on tärkeää muistaa tekstuurien ja kentän harmonia, sekä valojen ja tilojen vaikutus ympäristöön. Jos työstettävä pelikenttä on rea-

listinen, on tärkeää tutkia ja kerätä taustatietoa tiloista ja tunnelmasta jota tavoitellaan. (Määttä 2002.)

Kun 2D-kuva on tuotu kenttäeditoriin, se lisätään materiaaleihin. Materiaalieditorissa tekstuuriin voidaan lisätä mm. shader-tasoja ja säätää kuvan sävyjä. 2D-kuva käsitellään aina ennen editoriin tuomista Photoshopissa tai muussa vastaavassa kuvankäsittelyohjelmassa. Kun tekstuuri on saatu valmiiksi materiaalieditorissa, voidaan se tuoda kenttään. Lähes kaikissa kenttäeditoreissa tekstuurien käyttö on melko samankaltaista. Valitaan materiaali ja pinta, jolle tekstuuri tahdotaan näkymään. Tämän jälkeen käytössä on erilaisia työkaluja, joilla tekstuuriin voidaan vaikuttaa. Vaikka 3D-avaruudessa onkin kolme akselia: X, Y ja Z, tekstuurit eivät kaksiuloitteisuudestaan johtuen toimi kovin hyvin XYZ -koordinaatiston kanssa. Yleinen 3D-pintojen teksturoinnissa käytettävä järjestelmä on U-, V- ja W- järjestelmä. Yksinkertaisesti selitettynä: W-akselilla kontrolloidaan tekstuurin kulmaa, U-akseli vastaa tekstuurin leveyttä asetetulla pinnalla ja V-akselilla pituutta. Yleisimmät työkalut joilla tekstuurin asetteluun voidaan vaikuttaa ovat rotaatio, skaalaus, liikuttaminen ja peilaaminen.(Kuva 16) Yleensä tekstuurin asettelu pinnoille vaatii hienosäätöä. Esimerkiksi kulmauksia teksturoidessa, tekstuurit täytyy asettaa kohdakkain käsin. Tämä tehdään joko UVW:tä muokkaamalla tai liikuttamalla tekstuuri kohdalleen. (Byrne 2004, 245-247.)



KUVA 16. Erilaisia tekstuurityökaluja

5.31 Resoluutio

Tekstuurien resoluutio mitataan pikseleissä. Tekstuurien resoluutiosta puhuttaessa käytetään muotoa (leveys) × (korkeus). Esim. 128 x 64 tekstuuri on siis 128 pikseliä leveä ja 64 korkea. Tekstuurien resoluutiot nousevat yleensä kahden potensseissa, koska tietokoneet suosivat tie-

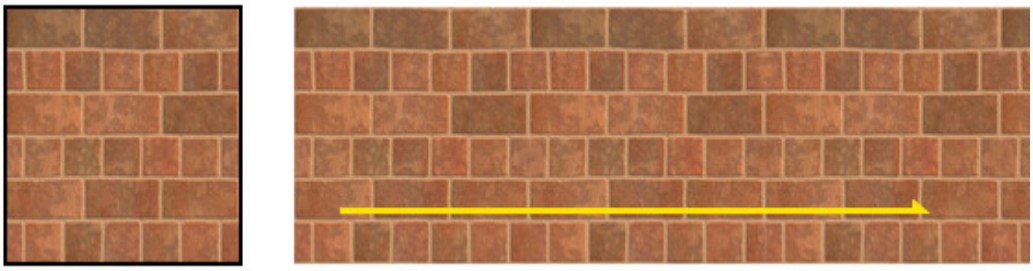
toa joka on jäsennelty näin. Eli 2, 4, 8, 32, 64, 128 ja niin edelleen. (Byrne 2004, 243.)

Tekstuuri ei saa olla liian iso, koska isoa tekstuuria on vaikea käsitellä ja se syö paljon muistia. Isoja tekstuureja käytetään vain tarvittaessa ja käytettäessä 1024x1024 resoluutioisia tai isompia tekstuureja, täytyy käyttökohde valita huolella. Toisaalta liian pieniresoluutioinen tekstuuri näyttää sotkuiselta tarkempiin tekstuureihin verrattuna. (Brinck 2007.)

Tekstuurinkoko on riippuvainen käyttökohteestaan, joissain tapauksissa hyvin toistuva 128x128 tekstuuri toimii hyvin, joskus tarvitaan 2048x2048 tekstuuri. Nykypeleissä suurin osa tekstuureista on 512x512 tai 1024x1024 resoluutiossa. Hahmojen, ajoneuvojen ja muiden uniikkien objektien tekstuuriresoluutio on yleensä 2048x2048. Tekstuurien koon ja määrän määrää käytettävissä oleva grafiikkamuisti.

5.32 Toistuvat ja toistumattomat tekstuurit

Peleissä käytetään yleensä kahdenlaisia tekstuureja, pinta- ja objektitekstuureja. Pintatekstuurit ovat yleistekstuureja, jotka sopivat monenlaisiin tilanteisiin, kun taas objektitekstuurit tehdään yksilöllisesti jotain tiettyä tarkoitusta varten, eikä niitä voida useinkaan käyttää kuin tarkoitetussa yhteydessä. Koska on mahdoton tehdä jokaiselle pinnalle omia uniikkeja tekstuurejaan, täytyy tekstuuriartistien tehdä tekstuureista toistuvia. Toistuva tekstuuri voi toistua vierekkäin tai päällekkäin ilman, että tekstuurissa näkyy mitään rajoja tai silmiinpistävästi toistuvia elementtejä. (Kuva 17) Toistuvia tekstuureja käytetään esimerkiksi seinien tai maaston teksturointiin. Toistumattomia tekstuureja käytetään yleensä luomaan yksityiskohtia ilman polygonimäärän kasvua. Toistumaton tekstuuri voi olla uniikkiseinä-, objekti- tai yksityiskohtatekstuuri. Yksityiskohtatekstuurilla saadaan luotua ikkunoita, oviaukkoja, likatahroja tai luodinreikiä. (Byrne 2004, 247-248.)

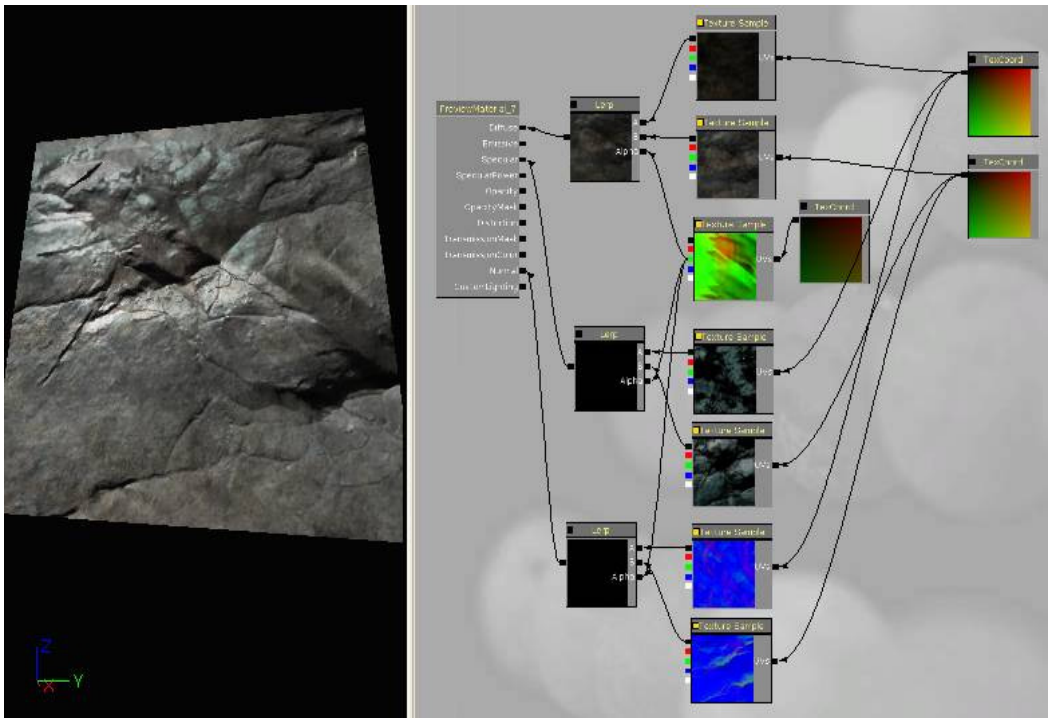


KUVA 17. Toistuva tekstuuri

KUVA 18. Toistumaton yksityiskohtatekstuuri

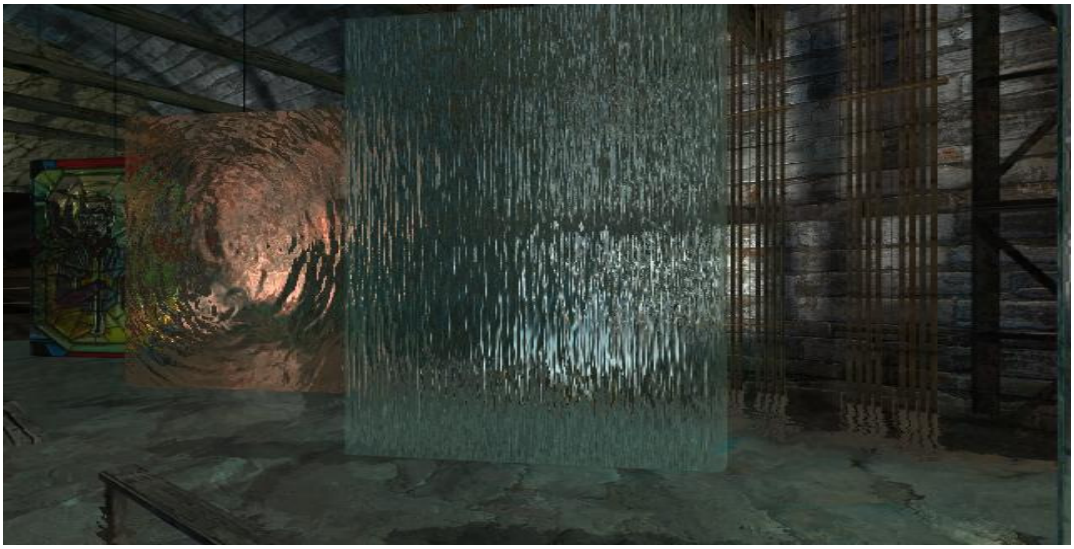
5.33 Materiaalit ja shaderit

Uudet pelit ja pelimoottorit eivät enää käytä tekstuureina pelkkiä 2D-kuvia, vaan käytössä ovat materiaalieditorit, joilla luodaan materiaaleja. 2D-kuvaan voidaan lisätä mm. kiiltoja, displacement-, bumb- ja normal map -tasoja. Nykyajan kenttäeditorien materiaalityökalut muistuttavat 3D-mallinnusohjelmien vastaavia.
(Byrne 2004, 244.)



KUVA 19. Unreal 3- pelimoottorin tyypillinen materiaalipuu. (Brinck 2007)

Nykyiset näytönohjaimet mahdollistavat ns. ohjelmoitavan työstövaiheen, mikä tarkoittaa käytännössä sitä, että näytönohjaimen ulostuottoa voidaan muokata shadereilla. Shadereitä on kahta luokkaa: pikseli- ja verteksi-shadereita. Pikselishaderillä voidaan vaikuttaa jokaiseen pikseliin, joka piirretään ruudulle, Verteksi-shaderillä voidaan taas vaikuttaa vain vertekseihin. Shaderit ovat eräänlaisia ohjelmia, shaderimalleja on lukuisia, 1.1, 1.4, 2.0, 2.0a, 3.0 ja uusimpana 4.0.(Matt Stafford 2007.)



KUVA 20. Erilaisia shadermateriaaleja, Source (Valve 2004)

5.4 Valaistus

Valaistus on elintärkeä osa hyvännäköistä kenttää, ilman valoja kenttä on joko täysin musta tai valaistu. Tämä toimi ehkä pelien alkuhistoriassa, muttei nykyään. Paras lähtökohta valaistuksen suunnittelussa on todellisuus. Kentän valaistus kannattaa suorittaa vaiheittain muun kentänteon ohella: usein lopullinen valaistus löytyy kokeilemalla ja vaihtelemalla eri valoyhdistelmiä. (Määttä 2002.)

Kentän tekovaiheessa käytetään väliaikaisia, karkeasti aseteltuja valoja tukemaan kentän muotoa ja helpottamaan kentän hahmottamista mallinnusvaiheessa. Vaihe vaiheelta valaistukseen lisätään yksityiskohtia, kuten esimerkiksi punainen valo liikennevalosta tai monitorin sininen hehku. Kannattaa kuitenkin välttää liiallista yksityiskohtien lisäämistä, järjestelmää on tällöin vaikeampi hallita ja liian usean valon yhdistelmä voi vaikuttaa kaoottiselta. Yleensä hyvä lähtökohta on asettaa yksi isompi valonlähde ja lisätä yksityiskohtia pienemmillä värillisillä valonlähteillä. (Määttä 2002.)

5.41 Dynaamiset ja staattiset valot

Valaistustyökalut, jotka nykyaikaiset kenttäeditorit ja pelimoottorit antavat kenttäsuunnittelijalle, eroavat vain vähän 3D-mallinnusohjelmistojen vastaavista. Pelien reaaliaikaisuus rajoittaa kuitenkin kenttien valaistusta: yhden valon piirtäminen ja laskeminen ei ole vaativaa, mutta nykyaikaiset kentät sisältävät usein tuhansia valonlähteitä. Tällaisen määrän laskeminen reaaliajassa ei ole mahdollista, joten kenttien valaistus muodostuukin usean erilaisen valaistustekniikan sekoituksesta, kuten esi-

merkiksi valokartoista ja verteksivalaistuksesta(Kuva 21). (Byrne 2004, 251-253.)

Koska suurin osa kenttien geometriasta on paikoillaan, eikä niiden muoto muutu, ei yleensä ole tarvetta laskea valojen käyttäytymistä reaaliajassa. Tällöin puhutaan staattisesta valaistuksesta. Kentän jokaisen valon vaikutus lasketaan, ja tämän perusteella luodaan uniikkitekstuuri eli valokartta. Tämä tekstuuri asettuu automaattisesti laskettujen pintojen päälle eräänlaiseksi varjostuskerrokseksi ja kenttään saadaan esilasketu valaistus. Esilasketun valaistuksen tarkkuus riippuu valokartan resoluutiosta. Valokarttatekniikka toimii erittäin hyvin isoilla pinnoilla, mutta pienempien pintojen tai monimutkaisen geometrian kanssa valokarttatekniikkaa ei useinkaan käytetä, sillä laskuajat ja virheiden määrät kasvavat liian suuriksi. Näissä tapauksissa valaistukseen käytetään verteksivalaistusta. Kaikki liikkuva ja hyvin monimutkainen valaistaan yleensä verteksivalaistuksella. 3D-mallit muodostuvat polygoneista, ja polygonit muodostuvat vertekseistä. Verteksit voivat sijaintinsa ohella tallentaa myös väri-arvon. Pelimoottori laskee jokaisen verteksin vastaanottaman valonmäärän ja valaisee mallin tämän tiedon perusteella. Verteksivalaistuksen lopputulos on huomattavasti yksinkertaisempi kuin valokartoilla aikaansaatu (Kuva 21.), mutta tekniikka on nopea. Verteksivalaistus on myös riippuvainen polygonien määrästä, esimerkiksi pitkä putki, jonka sivut muodostuvat pitkistä polygoneista, valaistuu väärin, koska putkessa ei ole tarpeeksi verteksejä tarkemman valaistuksen laskemiseksi. Mitä enemmän verteksejä, sitä tarkempi verteksivalaistus. (Byrne 2004, 253-252.)

KUVA 21. Vasemmalla käytössä verteksivalaistus, oikealla valokartat (Spoondog 2004)

Uusimmat pelimoottorit ja pelit käyttävät reaaliaikavalaistusta, jonka lopputulos ei välttämättä ole aivan samaa luokkaa kuin valokartoilla ja verteksivalaistuksella aikaansaatu, mutta se on täysin dynaaminen. Jos pelaaja esimerkiksi sammuttaa huoneesta lampun, muuttuu valaistus sen mukaisesti. Muunmuassa Doom 3-peli käyttää reaaliaikavalaistusta. Valokartat ja verteksivalaistus tulevat silti säilymään vielä pitkään pelien perusvalaistustekniikkoina. (Byrne 2004, 253-256.)

5.42 Sävy ja värikylläisyys

Valoilla ei useinkaan ole varsinaista väriä, lähes kaikki valot ovat valkeita joihin on sekoittunut hieman tiettyä värisävyä. Esimerkiksi kattolamppu on valkea valo, jossa on hienoinen keltaisen sävy. Usein värikylläisyyttä käytetään liikaa, mikä luo amatöörimäisen vaikutelman. (Valvewiki 2008.)

Todellisuudessa värien välillä vallitsee harmonia eli värit ovat tasapainossa. Tämä johtuu valon tavasta heijastaa heijastamansa pinnan värejä: erilaiset värit värittävät toisiaan heijastuksen kautta. Tämä on hyvä ottaa huomioon tekstuurien ja valojen valinnassa. (Määttä 2002.)

5.5 Väri

Värin tehtävä on muutakin kuin määritellä miltä mikäkin esine näyttää. Jokaiseen väriin liittyy tietty tunnetila ja tarkoitus. Tosielämässä esimerkiksi vihreä väri assosioidaan liikkeeseen ja välkkyvä valo kuvastaa toi-

mintaa. Jos värejä käytetään pelissä ristiriidassa tosielämän kanssa, pelaaja usein hämmentyy ja tulkitsee väärin pelin tunnelman tai viestin. (Valvewiki 2008.)

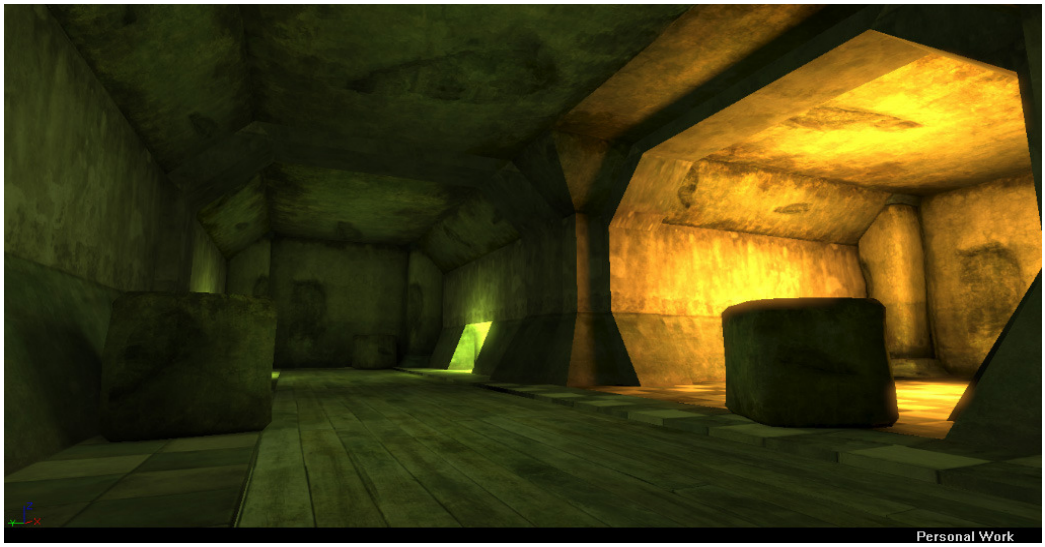
Hyvä apuväline niin kentän kuin tosielämänkin väriratkaisuihin on väriympyrä. (Kuva 17.) Väriympyrä voidaan jakaa lämpimiin ja kylmiin väreihin. Tämän lisäksi kenttäsuunnittelijan olisi hyvä tietää ainakin seuraavat värijärjestelmät kentän ulkoasua suunnitellessaan: Analogiset värit ovat värejä, jotka ovat lähekkäin toisiaan väriympyrässä. Vastavärit ovat värejä, jotka ovat vastakkain väriympyrässä, ja niillä on esimerkiksi helppo luoda kenttään kontrasteja. (Loeffler 2004.)

KUVA 22. Väriympyrä

5.51 Tunnelma ja tyyli

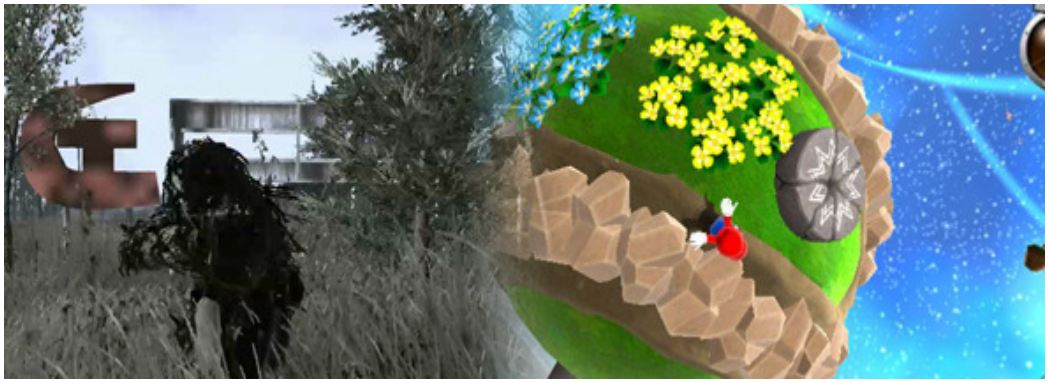
Kenttäsuunnittelijan tulisi tietää värien sisäänrakennetut viestit ja käyttää niitä hyödykseen kenttää rakentaessaan. Esimerkiksi Half-Life 2 pelissä käytettiin haaleaa sinisiä valoja luomaan kylmää tunnelmaa ja lämpimän keltaisia valoja turvallisen tunnelman luomiseksi. (Valve developer community 2008.)

Kuvassa 23 nähdään käytännössä, miten väreillä saadaan aikaan tunnetiloja, vihreä päävalaistus viestii epähygieniasta, turvattomasta ja hylätystä tilasta. Sivuhaaran keltainen valo viestii turvasta, lämmöstä. Värit ja valaistus ovat myös hyvä keino ohjata pelaajaa, keltainen valo kutsuu pelaajaa tutkimaan sivuhaaraa.



KUVA 23. Esimerkki tunnelmallisesta värienkäytöstä (Loeffler 2004)

Tunnelman lisäksi värien määrällä ja värikylläisyydellä on myös vaikutus pelintyyliin. Jos tavoitellaan piirrosmaista tai epätodellista tyyliä, värejä voidaan käyttää paljon vapaammin kuin realistisuutta tavoitellessa. Yleensä mitä fotorealistisempi peli sitä kesympi värimaailma.(Kuva 24) (Loeffler 2004.)



KUVA 24. Call of duty 4 ja Super Mario Galaxy

6 CASE: Kenttä Source pelimoottorille

6.1 Esittely

Opinnäytetyön case-osiossa käsittelen pienimuotoisen esimerkkikentän rakentamista Source-pelimoottorille. Valitsin Sourcen alustakseni koska pelimoottori on monipuolinen ja sillä pystytään toteuttamaan suoraan lähes kaikki käsittelemäni aihealueet. Myös Sourcen kattava dokumentointi ja käyttäjäystävällisyys vaikuttivat valintaan.

Case-osiossa tarkoituksenani on rakentaa kenttä, jossa esitellään opinnäytetyön teoriaosuudessa käsiteltyjä asioita.

6.2 Esityö

Aloitin kentän suunnittelun paperilla. Laitoin ylös asioita, joita kentässä tulisi olla ja asetin itselleni aikataulun kentän valmistumiseen. Tavoitteena on luoda kohtalaisen yksinkertainen kenttä, joka kuitenkin esittelee opinnäytetyössä käsiteltyjä asioita kattavasti.

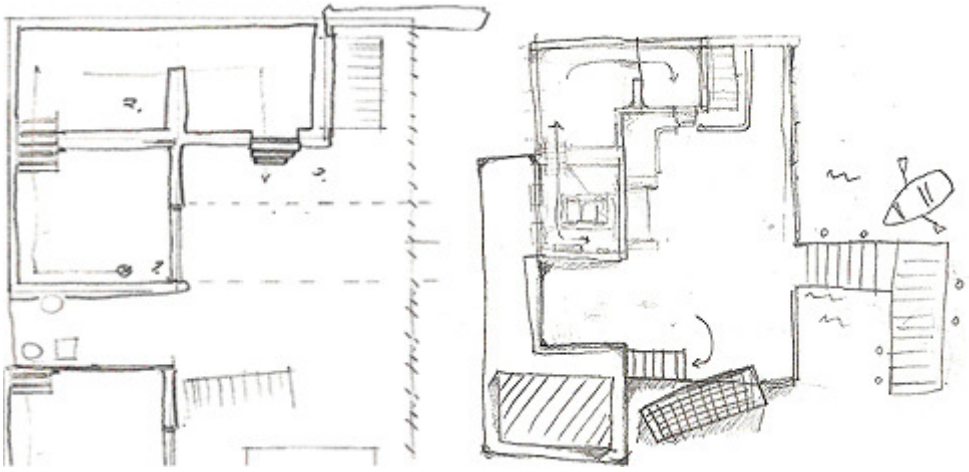
Kentässä käsitellään mm. seuraavia opinnäytetyöhön liittyä asioita:

- väri ja valo.
- staattiset objektit. Useita esimerkkejä staattisista objekteista ja niiden käytöstä.
- ulkotila, huomiopiste ja maasto. Esimerkki maastosta ja huomiopisteen käytöstä.
- skybox, taustageometria ja aurinko.
- vesimateriaali
- fysiikka. Kentässä objekteja joilla fysiikkaominaisuuksia.

Kentän graafinen tyyli on realismi ja pohjautuu nykyaikaan. Kentässä käytetään Half-Life 2 pelin tekstuureita ja objekteja.

6. 21 Konseptointi

Laadin kentästä ensimmäiseksi lyijykynäkonseptit. Tarkoituksena on laatia peruslayout kentälle, jonka perusteella lähdetään sitten jatkamaan perusgeometrian rakennusvaiheeseen. Konseptointivaiheessa on hyvä suunnitella kentän perusrakennetta, kuten esimerkiksi pelaajan kulkureitejä.



KUVA 25. Konseptipiirustuksia kentän layoutista

Piirsin kenttää varten vain kaksiulotteisen suunnitelman kentän layoutista, mutta usein konsepteista tehdään kolmiulotteisia ja värillisiä. Usein konseptien päälle tai niiden perusteella tehdään myös nopeita väritutkielmia, joista on suurta apua kentän teksturointi ja valaistusvaiheessa.

6.22 Referenssikuvat

Referenssikuvia kerätään yleensä prosessin alkuvaiheessa, sen jälkeen kun on päätetty millaisia paikkoja ja mitä elementtejä kenttään mahdollisesti halutaan. Referenssikuvia ei ole tarkoitus kopioda yksi yhteen pelimaailmaan, vaan kuvista yleensä käytetään vain pieniä osia. Referenssikuvia käytetään myös kentän yleisen ulkoasun ohjenuorana.

KUVA 26. Kollaasi referenssikuvista

Case-kenttään hain kuvia kellareista, autotalleista, satamasta sekä ränsistyneistä taloista. Kuvassa 26. olen koonnut muutamia käyttämiäni referenssikuvia yhteen. Yleensä kuvia haetaan todella paljon, kappalemäärissä puhutaan sadoista aina tuhansiin kuviin. Kenttäsuunnittelija saattaa käyttää pelkkien referenssikuvien etsimiseen päiviä, riippuen projektista.

6.3 Kentän rakentaminen

6.31 Perusgeometria

Rakensin kentästä ensimmäiseksi raakaversion. Käytin BSP-primitiivejä joilla veistin kentän karkean geometrian. Tämän vaiheen jälkeen kenttää voidaan jo testata, esimerkiksi huoneiden skaalat ja reitit voidaan todeta toimiviksi. Käytin hyödyksi Valven Development- tekstuureja, joista on apua esimerkiksi huoneiden skaalan kohdalleen määrittämisessä. Tässä vaiheessa lisäsin kenttään myös muutamia valolähteitä, jotka sijoitin kenttään kuitenkin lähinnä vain helpottamaan geometrian hahmotusta.

KUVA 27. Perusgeometria ja development tekstuurit

Perusgeometriaan kuuluu vain välttämättömin geometria eli sellainen geometria, joka on pelattavaa. Esimerkiksi talojen kattoja tai pienimpiä ikkunoita ei huomioida tässä vaiheessa ollenkaan. Perusgeometria on vain suuntaa antavaa lähes konseptimaista veistelyä. Usein rakennukset rakennetaan uudelleen myöhemmässä vaiheessa ja alkugeometriaa käytetään vain pohjana.

6.32 Teksturointi ja yksityiskohtatekstuurit

Perusgeometrian veiston jälkeen aloitin teksturoinnin. Kokeilin erilaisia väriyhdistelmiä ja kontrasteja. Lopulta päädyin ränsistyneeseen tehdasmaiseen ulkoasuun, jota lähdin työstämään eteenpäin. Valven tarjoama

tekstuuripankki on varsin kattava, eikä tietyn tyylisen tekstuurin löytäminen tuottanut paljonkaan ongelmia. Lopulliseen kenttään tuli 29 uniikkia tekstuuria. Suurin osa tekstuureista on 512x512 resoluutiossa.

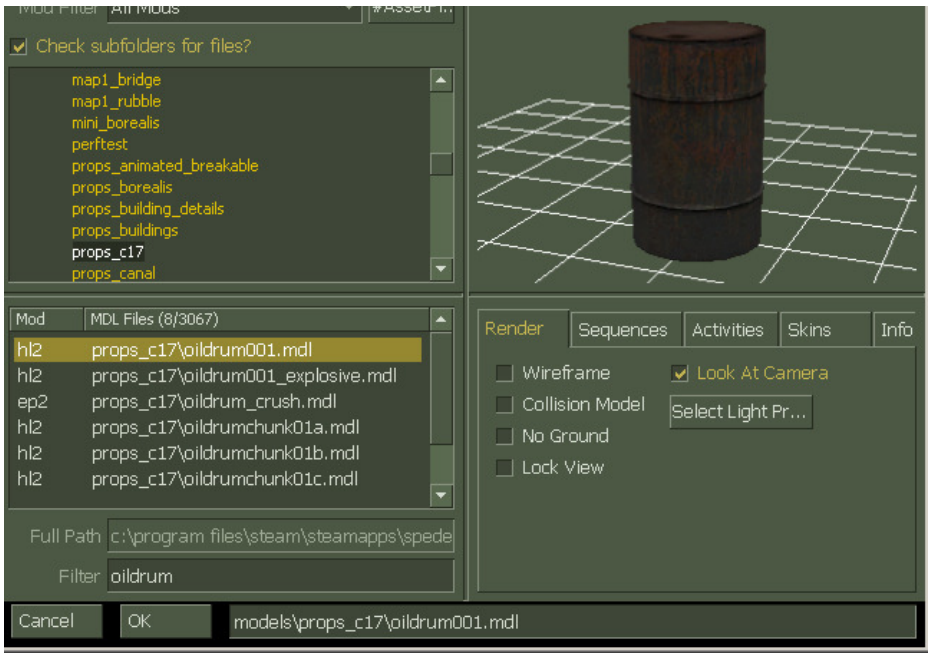
KUVA 28. Alustavaa teksturointia

Käytin normaalien tekstuurien lisäksi yksityiskohta- eli decal- tekstuureja rikkomaan toistuvia tekstuureja ja luomaan lisäyksityiskohtia. Yksityiskohtatekstuuri on plane objekti, joka asetetaan aivan seinäpinnan viereen. Planelle projisoidun tekstuurin Alpha-kanavalla määritetään tekstuurin näkyvä osa. Kuvassa 29 näkyvät seinän eri rapaumat ovat kaikki luotu yksityiskohtatekstuurein.

KUVA 29. Yksityiskohtatekstuurin käyttö

6.33 Staattiset objektit

Teksturoinnin ja BSP-geometrian jälkeen lisäsin kenttään staattisia objekteja. Staattisilla objekteilla saadaan kenttään luotua yksityiskohtaisempaa grafiikkaa kuin pelkällä BSP-geometrialla. Staattinen objekti lisätään kenttään luomalla ensin Prop_Static entity, jonka jälkeen entityn asetuksista valitaan mikä objekti ladataan. Objektien valinta suoritetaan Model Browser- nimisessä ohjelmassa. Half-Life 2 pelin mallikirjastossa on noin 3100 eri objektia.

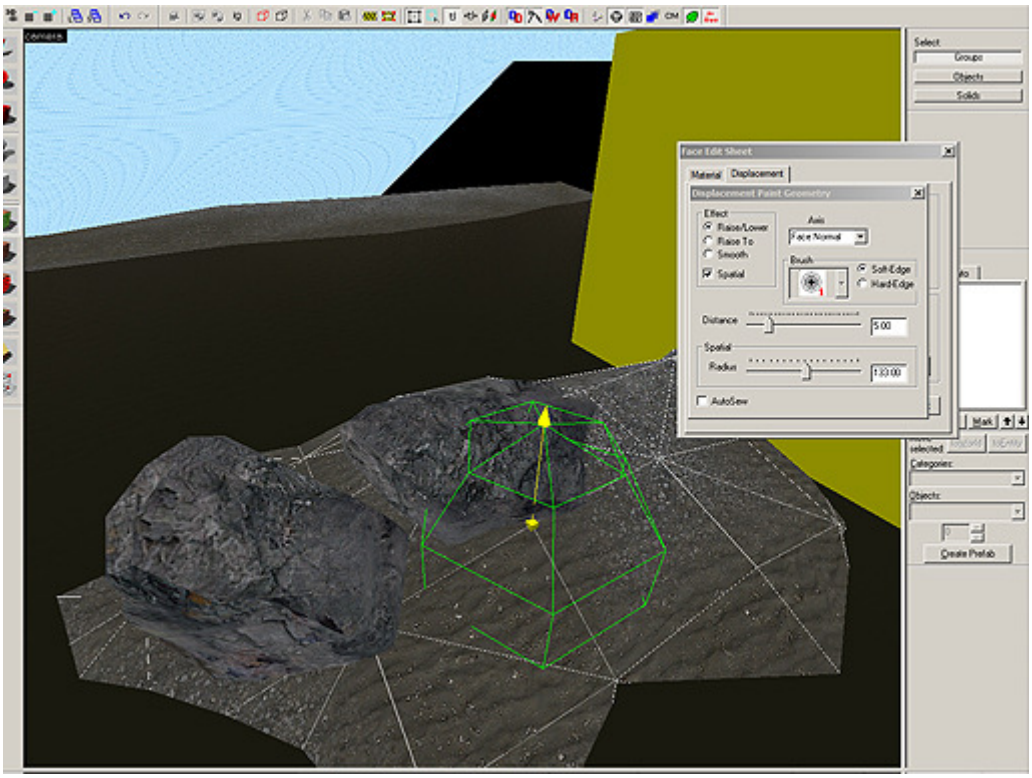


KUVA 30. Tynnyrimalli esillä Model Browser -ohjelmassa

Pros_Static entity on nimensä mukaisesti staattinen, fysiikkaobjekteille on erillinen Prop_Dynamic entity, jolla pelimaailmaan saadaan tuotua fysiikkamallilla varustettuja objekteja, kuten tynnyreitä tai laatikoita.

6.34 Maasto

Käytin Hammerin maastotyökaluja luomaan pienet saaret, jotka näkyvät pelikentän rajoilla. Hammerin maastotyökalut eivät ole muiden kilpailevien kenttäeditorien tasoisia, mutta niillä saadaan silti tehtyä hyvännäköisiä maastoja. Maasto tehdään luomalla ensin BSP-laatikko, jonka muokattavaksi haluttu taso sitten valitaan tekstuurityökalulla. Tämän jälkeen valitaan Displacement välilehti. Valittu BSP-taso jaetaan pienenpiin osiin, tämän jälkeen tasoa voidaan muokata nostelemalla ja laskemalla verteksejä.(Kuva 31.) Hammerissa on myös tuki eräänlaisille ”maalattaville” maastotekstuureille, ominaisuus jota käytin saarien teksturointiin. Luotu maasto teksturoidaan Blend- materiaalilla, joka muodostuu kahdesta eri tekstuuritasosta. Displacement- välilehdeltä valitaan Alpha paint työkalu, jolla voidaan vertekseille määrittää kuinka paljon halutaan näyttää Blend- materiaalin tiettyä tekstuuria.



KUVA 31. Maastonluonti Hammer-editor

6.35 Taivas ja taustageometria

Koska kentässä pääsee myös ulkotiloihin, täytyi kenttään tehdä taivaslaatikko eli skybox ja jonkin verran taustageometriaa. Source-

pelimoottori vaatii tilan, johon pelikenttä on suljettu, muuten moottori ilmoittaa geometriavuodosta ja kenttä ei käynnisty. Skyboxilla on helppo luoda tämä tila ja samalla saadaan kenttään myös taustataivas. Pelikenttä täytyy vain rajata tarpeeksi isolla ”laatikolla”, joka sitten teksturoidaan Skybox- nimisellä materiaalilla. Tämän jälkeen voidaan kartan asetuksista valita haluttu taustataivas. Kokeilin kenttään useita eri taivas vaihtoehtoja. Lopulta valitsin kenttään melko selkeän taivaan, jossa auringon sijainti oli suunnilleen haluamallani kohdalla.

Lisäksi kenttässä on sumua, joka häivyttää pelaajasta kauempia objekteja ja geometriaa ja luo kentästä realistisemman. Samalla sumu myös optimoi kenttää, sillä sumuun häviäviä objekteja ei piirretä. Sumulle voidaan määrittää väri, tiheys sekä alkamis- ja loppumisetäisyydet.

6.36 Vesi

Veden luominen on aika yksinkertaista. BSP-primitiivillä luodaan halutun vesialueen kokoinen muoto, BSP:lle asetetaan Nodraw- materiaali ja päällimmäinen plane valitaan erikseen, sille asetetaan Water- materiaali. Pelimoottori osaa suoraan heijastella pelimaailmaa vedenpinnasta. Vedellä on myös fysiikka ominaisuudet ja asetin veteen muutamia veneitä jotka reagoivat veteen realistisesti.

Kuva 32. Vesimateriaali

6.37 Valot

Kentässä on käytössä kolme erilaista valotyyppiä: spotti- ja pistevalo, sekä enviroment light eli auringonvalo. Ulkotilassa käytössä on pelkästään auringonvaloa, valo luo kenttään realistisia varjoja. Ulkokentissä aurinko on sekä tärkeä valonlähde että visuaalinen kenttäelementti. Aurinko tehdään luomalla Light_Enviroment entity. Entityn sijainti määrää auringon paikan ja suunta voidaan määrätä erikseen ”Point at” työkalulla. ”Point at” työkalulla valitaan kohde, johon auringonvalo halutaan suunnata. Tämän lisäksi auringolle on hyvä luoda heijastusobjekti (Env_sun), joka saa aikaan häikäisyä simuloivan efektin kun pelaaja katsoo auringon suuntaan. (Kuva 33.)

Sisätiloissa käytin spottivaloja autotalliin johtavan portaikon valaisuun, spottivalojen tarkoitus on ohjata pelaajaa.(kuva 34) Lisäksi autotallissa on hehkulamppuun liitetty pistevalo.

Valot lasketaan ennalta kenttään kentän käännösvaiheessa. Kenttään lasketaan varjot, jotka tallennetaan valokarttatietona kenttään. Valokarttojen resoluution kokoa voidaan muutella ja saada aikaan terävämpiä varjoja. Muutin joidenkin seinien valokartta-arvoja isommiksi, mutta esilaskenta-aika karkaa hyvin äkkiä käsistä, jos valokarttojen kokoa nostetaan liikaa. Korkeampi resoluutioiset valokartat ovat käytössä vain isoimman rakennuksen sisällä. (Kuva 34)

KUVA 33. Auringon häikäisyefekti.

KUVA 34. Spottivalo ohjastaa pelaajaa.

6.4 Viimeistelty kenttä

Kentän työstäminen on vaiheittain etenevä prosessi, välillä työstetään geometriaa, välillä valoja. Lopullinen kenttä saattaa erota huomattavasti esisuunnitteluvaiheen visiosta. Esisuunnittelun tulisi olla perusteellinen, mutta kuitenkin tarpeeksi avoin jotta pienet muutokset ovat mahdollisia.

Case-kenttä toteutui hyvin pitkälle sellaisena kuin se oli suunniteltu. Kenttä on lähes julkaisuvalmis, lukuun ottamatta sitä että kentässä ei ole varsinaista objektia eikä vihollisia. Kenttä on näyttävä ja täyttää tehtävänsä, eli esittelee opinnäytetyössä käsiteltyjä asioita.

Lyhyt video [Liite a], sekä kuvankaappauskokoelma[Liite b] kentästä löytyvät liitteenä olevalta cd-levyltä. Liikkuvassa kuvassa asiat, kuten esimerkiksi vesimateriaali ja valaistus, tulevat paljon konkreettisemmin esille kuin still-kuvista.

KUVA 33. Viimeistelty kenttä

7 YHTEENVETO

Kenttäsuunnittelussa käytetyt työskentelytavat riippuvat hyvin pitkälle käytetystä pelimoottorista. Pelien perusrakenne on kuitenkin sama riippumatta käytetystä pelimoottorista tai työkaluista. Pelattavuus, hauskuus ja haaste ovat asioita, jotka pysyvät muuttumattomina. Ne ovat samalla asioita, joita on vaikein saavuttaa riippumatta teknisistä edistysaskelista.

Tekniikan kehittyksen myötä muuttuu myös pelialan rakenne: uusia työtehtäviä syntyy ja vanhat jakautuvat pienempiin osiin. Kenttäsuunnittelu tulee tulevaisuudessa luultavasti pirstaloitumaan vielä pienempiin erikoisaloihin kuin mitä nykyään voidaan erotella. Ryhmäkokojen kasvaessa myös yhteistyön merkitys kasvaa.

Pelikenttien suunnittelussa on tärkeää pitkä ja perusteellinen esisuunnitteluvaihe sekä vahva visio lopputuloksesta. Hyvän suunnitelman pohjalta on helppo lähteä rakentamaan kenttää. Toisaalta toteutusvaiheessa saadaan usein ideoita, joita ei vielä paperilla osattu nähdä. Suunnitelmiin onkin syytä jättää pieni vara yllättäville muutoksille.

Source-pelimoottori oli hyvä valinta esimerkkikentän toteutusympäristöksi, ja pelimoottori on joustava ja näyttävä. Työkalut ovat helppokäyttöisiä ja tehokkaita. Pelikentästä tuli näyttävä, ja kaikki suunnitellut asiat pystyttiin toteuttamaan ilman suurempia ongelmia.

Kenttäsuunnittelu on yksi peliteollisuuden kiinnostavimpia työaloja, ja samalla se on myös yksi tärkeimmistä. Kenttäsuunnittelijalta vaaditaan ryhmätyöskentelykyvyn lisäksi taiteellista ja teknistä osaamista. Kenttäsuunnittelu onkin teknistä visualisointia parhaimmillaan.

LÄHTEET

1. Painetut lähteet

Byrne, E. 2004. Game level design. Charles River Media.

Feil, J. & Scattergood, M. 2004. Beginning Game Level Design. Course Technology, Inc.

2. Sähköiset lähteet

Shahrani, S. 2004. [Verkkolähde] A History and Analysis of Level Design in 3D Computer [viitattu 06.01.2008]
Saatavissa:
http://www.gamasutra.com/view/feature/2674/educational_feature_a_history_and_.php
http://www.gamasutra.com/view/feature/2682/educational_feature_a_history_and_.php

Duarte, S. 2005. [Verkkolähde] What is Level Design? [viitattu 11.12.2007]
Saatavissa:
http://www.interlopers.net/index.php?page=what-is-level-design

De Jong, S. 2006 [Verkkolähde] Hows and whys of Level design. [viitattu 10.12.2007]
Saatavissa:
http://book.hourences.com/bookgameplay.htm

Mapping 101: Gameflow,. 2007 [verkkolähde] A Mallo's World design: Mapping 101: Gameflow [viitattu 10.01.2008]
Saatavissa:
http://www.mallosworld.co.uk/creating_worlds/2006/07/15/mapping-101-gameflow/

Johnston, D. 2008. [Verkkolähde] Level design [viitattu 24.03.2008]
Saatavissa: http://www.johnsto.co.uk/design

Korva, T. 2002. [Verkkolähde] Open Source & Low Cost Game Engines [viitattu 25.03.2008]

Saatavissa:
http://ludocraft.oulu.fi/elias/dokumentit/open_source_game_engines.pdf

Valve developer community, 2008 [verkkolähde] [viitattu 24.3.2008]
Saatavissa: http://developer.valvesoftware.com/wiki/Main_Page

Määttä, A. 2002. [verkkolähde] GDC 2002: Realistic Level Design for Max Payne [viitattu 20.01.2008]
Saatavissa:
http://www.gamasutra.com/features/20020508/maatta_01.htm

Loeffler, K. 2004. [verkkolähde] The Theory of Lighting [viitattu 20.01.2008]
Saatavissa:
http://www.keenleveldesign.com/html/misc/lightingtutorial1.htm

Stafford, M. 2007. [verkkolähde] Introduction to Source Shaders[viitattu 22.01.2008]
Saatavissa:
http://www.wraiyth.com/data/shadertute/Introduction_To_Shaders.doc

Smalley, T. 2005. [verkkolähde] Cinematic Effects in Source [viitattu 23.01.2008]
Saatavissa:
http://www.bit-tech.net/gaming/2005/12/09/source_film_effects/

Unreal Developer Network, Epic Games, 2007 [verkkojulkaisu] [viitattu 24.1.2008]
Saatavissa: http://udn.epicgames.com/Main/WebHome.html

Unrealwiki, 2008. [verkkojulkaisu] [viitattu 24.1.2008]
Saatavissa: http://wiki.beyondunreal.com/wiki/

Waylon Brinck, 2007. [verkkojulkaisu] [viitattu 26.1.2008]
Learning Unreal Editor 3.
Saatavissa: http://waylon-art.com/LearningUnreal/

KUVALÄHTEET

KUVA 1. Kenttäsuunnittelun rajapinta, Ed Byrne. 2004. Game level design. Charles River Media.

KUVA 2. Yksinkertainen vaikeustasonkasvun kuvaaja Ed Byrne. 2004. Game level design. Charles River Media. 73.

KUVA 3. Normaali vaikeustasonkasvun kuvaaja Ed Byrne. 2004. Game level design. Charles River Media. 74.

KUVA 4, Half-Life 2, city 17, huomiopiste. Valve 2002 http://pcmedia.gamespy.com/pc/image/article/566/566585/01440_1100638784.jpg

KUVA 5, Counter-Strike source, yksinkertaista pelaajan ohjastamista. Valve 2002 http://www.interlopers.net/images/articles/level-design/choice_t.jpg

KUVA 6, Node-kartta Half-Life:episode 1. Valve 2004 http://developer.valvesoftware.com/w/images/thumb/0/09/300px-Ep1_c17_05_nodegraph.jpg

KUVA 7, Unreal 3.0-pelimoottori. Epic 2005 http://pcmedia.ign.com/pc/image/article/515/515431/unreal-30-engine-screens-20040513083736063_640w.jpg

KUVA 8, Source-pelimoottori. Valve 2007 http://www.yonixgames.com/Images/Post/Half-Life-2-Episode-2-The-coming-combine-forces-on-the-bridge.jpg

KUVA 9, Hammer- kenttäeditori ruudunkaappaus Hammer ohjelmasta ,Markus Kallio 2008

KUVA 10. Kenttä muodostuu kokonaan staattisista muodoista, koko ti-lassa on vain 8 normaalia polygonia. Unreal Developer Network, Epic Games 2007 http://udn.epicgames.com/Two/rsrc/Two/StaticMeshesTutorial/mesas.jpg

KUVA 11. UnrealEd- kenttäeditorin BSP-työkalut. Unreal Developer Network, Epic Games 2007 http://udn.epicgames.com/Two/rsrc/Two/BspBrushesTutorial/Primitives.jpg

KUVA 12. Korkeuskartta ja sillä luotu maasto. Unreal Developer Network, Epic Games 2007 http://udn.epicgames.com/Two/rsrc/Two/CreatingTerrain/river.jpg

LIITTEET

CD-levyllä lyhyt video kentästä ja ruudunkaappauskokoelma.