SISÄLLYS

1 Johdanto

1.1 Opinnäytetyön tarkoitus

Tämän opinnäytetyön tarkoituksena on esitellä CoolBasic-peliohjelmointikieltä sellaisen henkilön näkökulmasta, joka ei ole välttämättä koskaan aikaisemmin ohjelmoinut. Toki tämä sopii myös aiemmin ohjelmoineiden henkilöiden luettavaksi, jotka haluavat löytää uuden ja helposti opittavan peliohjelmointikielen.

Yhtenä tarkoituksena on myöskin koota yhteen tietopaketti CoolBasicista, siihen liittyvistä työkaluista sekä sen käyttäjien muodostamasta yhteisöstä. Tarkoituksena ei ole verrata CoolBasicia muihin ohjelmointikieliin, eikä myöskään pelinteko-ohjelmiin.

1.2 Aihealueen valinta ja rajaus

Aihealueena on itse CoolBasic-ohjelmisto – sen asentaminen ja käyttäminen - esimerkkipelin tekeminen kielellä, CoolBasic:n käyttäjäyhteisö sekä kielen historia ja tulevaisuuden suunnitelmat. Yhtenä aihealuuena on myös omat pohdintani ja käyttökokemukseni kielestä. Tämä aihealue sisältää siis omia mielipiteitäni, jotka eivät pohjaudu lähdemateriaaleihin. Tähän aihealueeseen kuuluvat luvut olen merkinnyt selkeästi lukujen alkuun, jotta lukija tietää lukujen sisällön olevan omaa mielipidettäni.

Työssä tehdään yksinkertainen esimerkkipeli ja kerrotaan, kuinka se toimii. Tämä opinnäytetyö ei kuitenkaan ole ohjelmointiopas, joten itse kielen opiskelu tapahtuu tämän opinnäytetyön ulkopuolella. Esimerkiksi CoolBasicin komentoja ja funktioita ei tässä työssä opeteta järjestelmällisesti. Rajaan ohjelmointioppaan pois työstä, sillä CoolBasic sisältää laadukkaan manuaalin, josta kielen oppii helposti.

Käyttäjäyhteisöstä käydään läpi muutamia CoolBasiciin liittyviä internetsivustoja sekä CoolBasicin irc-kanava. Yksittäisiin käyttäjiin ei keskitytä.

Kielen historiassa keskitytään lähinnä versiohistoriaan, joten historiaosuus ei sisällä niinkään käyttäjiltä tulleita mielipiteitä kielen eri kehitysvaiheista.

Tulevaisuuden suunnitelmissa tarkastellaan CoolBasicin pääkehittäjän, Jukka Lavosen, keskustelupalstoille kirjoittamia suunnitelmia CoolBasiciin tulevista muutoksista ja uusista ominaisuuksista. Opinnäytetyön sisältämissä omissa pohdinnoissani ja käyttökokemuksissani kerron hieman omaa ohjelmointitaustaani – ja miten se on vaikuttanut CoolBasicin oppimiseen ja käyttämiseen - mitä olen CoolBasicilla tehnyt ja mitä odotan sen tulevaisuudelta.

2 Pelien tekemisen historiaa

Kun ensimmäiset keskustietokoneet syntyivät 1960-luvulla, sai samalla alkunsa myös peliohjelmointi. 1970-luvulle mennessä oli jo valmiina suuri määrä tekstipohjaisia pelejä, jotka saattoivat sisältää myös karkeatekoista grafiikkaa. Silloin useimmat pelit olivat verkkopohjaisia MUDeja. Lyhenne *MUD* tulee englannin kielen sanoista *Multi-User Dungeons*. (LaMothe, 2000, 9 - 10)

Suuri yleisö tosin tutustui tietokonepeleihin vasta *Nolan Busnell*'in suunnitteleman, suureen suosioon yltäneen *Pong*-pelin julkaisun jälkeen. Käytännössä kyseinen peli aikaansai videopeliteollisuuden. (LaMothe, 2000, 10)

Ensimmäiset yksityishenkilön kukkarolle sopivat tietokoneet – *TRS-80*, *Apple* ja *Atari 800* – ilmestyivät markkinoille vuosien 1976 – 1978 aikoihin. Näistä *Atari 800* oli selkeästi tehokkain ja sopivin pelikoneeksi. Monet teini-ikäiset nuoret alkoivat pikkuhiljaa tehdä näille järjestelmille pelejä. Kuitenkin loppujen lopuksi vain kourallinen ihmisiä osasi tehdä tietokonepelejä, eikä aiheesta löytynyt paljoakaan kirjallisuutta, joten ohjelmoijat olivat suurelta osin omillaan. (LaMothe, 2000, 10)

1980-luvulla tulivat saataville ensimmäiset 16-bittiset tietokoneet ja peliohjelmointi alkoi nousta suurempiin mittakaavoihin, ja pelit alkoivat jo näyttää graafisesti hyvältä. Muiden muassa *Wing Commander* ja *Flight Simulator* olivat jopa 3D-pelejä. *Amiga 500* ja *Atari ST* hallitsivat pelikonemarkkinoita vuoden 1985 aikoihin – *IBM PC*:n pysytellessä selkeästi pienemmässä suosiossa. (LaMothe, 2000, 10)

PC oli kuitenkin edullinen ja myös hyödyllinen tietokone työelämässä, joten sen suosio nousi pikkuhiljaa, ja 1990-luvun alussa *IBM PC* – yhteensopiva tietokone oli jo yleisempi kuin muut tietokoneet. Silti se oli edelleen grafiikaltaan ja ääniltään jäljessä muista tietokoneista. Se

vaikutti liian tehottomalta, jotta sille olisi voitu tuottaa yhtä hyvältä näyttäviä pelejä, kuin mitä oli totuttu näkemään *Amigalla* tai pelikonsoleilla. (LaMothe, 2000, 10)

Vuonna 1993 tapahtui kuitenkin muutos, kun *Id Software* julkaisi PC:lle pelin nimeltä *DOOM*, minkä johdosta PC:stä tuli suosituin peli- ja ohjelmointijärjestelmä kotitietokonemarkkinoilla. Kun ihminen on tarpeeksi kekseliäs, saa hän PC:n tottelemaan itseään erinomaisesti – ja *DOOM* on tästä todiste. (LaMothe, 2000, 11)

Samoihin aikoihin alkoi myös *Microsoft* pohtia uudelleen omaa asemaansa peliohjelmointialalla. Se halusi kuulua mukaan viihdeteollisuuteen, joten suunnittelutyöt alalle pääsemiseksi käynnistettiin. Käytännön ongelmana oli se, että *Windows 95* oli liikkuvan kuvan ja äänen käsittelyominaisuuksiltaan kehno. Tätä ongelmaa Microsoft yritti ratkaista kehittämällä *Win-G*-nimisen ohjelman, jonka uskoteltiin olevan korkeatasoinen peliohjelmointi- ja grafiikka-alijärjestelmä. Tosiasiassa kyseessä oli vain pari grafiikkakomentoa sisältävä ohjelma bittikarttojen piirtämiseksi, jonka olemassa olon Microsoft myöhemmin jopa kielsi. (LaMothe, 2000, 11)

Microsoft ei kuitenkaan jättänyt asiaa siihen, vaan oli jo aloittanut uusien grafiikka-, ääni-, syöttö-, verkko- ja 3D-järjestelmien työstämistä ostamalla *Rendermorphicsin*. Syntyi *DirectX*, jonka avulla Windows-peleistä olisi pitänyt Microsoftin mukaan tulla nopeampia kuin *DOS32*-pelit. Todellisuudessa näin ei kuitenkaan käynyt. DirectX:n ensimmäiset kaksi versiota nimittäin olivat tuotteina epäonnistuneita. Microsoft oli arvioinut peliohjelmoinnin monimutkaisuuden väärin, jolloin DirectX:stä ei tullut heti tarpeeksi tehokasta työkalua pelialalle. Tekniikaltaan se oli kuitenkin hyvä aloitusaskel ja suunta oli oikea. DirectX 3.0 oli jo siinä mielessä onnistunut, että se päihitti toimivuudessaan DOS:n. (LaMothe, 2000, 11)

Markkinat tosin eivät olleet DirectX:lle vielä tarpeeksi kypsiä, sillä valtaosa peliyhtiöistä valitsi peliensä alustaksi edelleen DOS32:n Windowsin sijaan, joten DirectX:lle ei alussa ollut käyttöä. Enemmän DirectX:ää ryhdyttiin käyttämään, kun siitä oli julkaistu versio 5.0. DirectX teki mahdolliseksi kaikkien grafiikka- ja äänitekniikoiden käyttämisen suoraan, mikä on DirectX:n laadukkaan rakenteen ansiota. (LaMothe, 2000, 11 - 12)

DOOM käytti pelkästään ohjelmistopohjaista rasterointia, mutta seuraavan sukupolven 3D-pelit – muun muassa Quake ja Unreal – käyttävät ohjelmoistorasteroijien lisäksi myös laitteistopohjaista kiihdytystä, mikä oli todella suuri kehitysaskel. (LaMothe, 2000, 12)

3 CoolBasicin esittely

3.1 Mikä on CoolBasic?

CoolBasic on erityisesti pelien tekemiseen tarkoitettu ilmainen ohjelmointiympäristö. Sillä on mahdollista luoda Windows-pohjaisia, DirectX 7.0:aa käyttäviä pelejä ja ohjelmia sekä kääntää niistä *EXE*-tiedostoja. CoolBasic sopii täysin vasta-alkaville ohjelmoijille sekä myös kokeneemmille pelinikkareille ja se on nopea oppia helpon *BASIC*-tyyppisen kielen ja laadukkaan, suomenkielisen manuaalin ansiosta. CoolBasic on suomalaisten kehittämä. Lisäksi CoolBasicin lisenssi antaa käyttää ja levittää ohjelmistoa todella vapaasti. Esimerkiksi tietojenkäsittelyn opettajat voivat käyttää sitä ohjelmoinnin opetuksessa! (URL 3.1, FILE 1)

CoolBasicin mukana tulee kattava manuaali, joka sisältää oman ohjesivun jokaiselle kielen sisäänrakennetulle komennolle ja funktiolle. Useimmille komennoille ja funktioille on niiden ohjesivulla lisäksi myös pieni esimerkkiohjelma, joka valaisee niiden toimintaa käytännössä. Manuaali on myös suurimmaksi osaksi suomenkielinen. Ainoastaan *Mitä*

uutta?-osio (*Version History*) sekä esimerkkikoodien kommenttimerkinnät ovat kirjoitettu englanniksi.

Kielen mukana tulee myös ohjelmointiympäristö, joka on sopivan yksinkertainen, jolloin aloittelijankin on helppoa käyttää sitä. Itseasiassa sana *koodieditori* kuvannee paremmin tätä *ohjelmointiympäristöä*, juuri yksinkertaisen käyttöliittymän vuoksi, joka ei sisällä suurta määrää erilaisia valikoita, ikkunoita tai nappuloita ynnä muuta. Kaikki toiminnot, mitä aloitteleva tai kokeneempikin ohjelmoija editorissa tarvitsee, löytyy nopeasti. (URL 3.1, URL 3.2)

3.2 CoolBasicin tekniset ominaisuudet

CoolBasic on vielä beta-vaiheessa, joten sen suorituskyky ei ole vielä huippuluokkaa, mutta riittää hyvin pienten pelien tekemiseen. 3D-grafiikkapuolta ei niin ikään vielä ole, mutta sellaista on lupailtu tulevaisuudessa. (URL 3.1, URL 3.3)

Coolbasic on tulkattava ohjelmointikieli, mikä tarkoittaa sitä, että ohjelmaa ei käännetä suoraan konekielelle, vaan kääntäjä kääntää ohjelman ensin välikielelle, jota tulkataan samalla kun itse luotua ohjelmaa ajetaan. Tämä hidastaa ohjelmia selkeästi, mutta kääntäjä on ollut huomattavasti yksinkertaisempi toteuttaa, kuin jos se kääntäisi lähdekoodin suoraan konekielelle.
Kielen tulkattavuuden vuoksi sen ydin koostuu kahdesta osasta: kääntäjästä ja virtuaalikoneesta. Kääntäjä kääntää ohjelman lähdekoodin virtuaalikoneen ymmärtämään muotoon (ja samalla tarkistaa koodin syntaksivirheiden varalta). Käännetty koodi liitetään virtuaalikoneeseen, ja tuloksena on ajettava *EXE*-ohjelma. Virtuaalikone taas tulkitsee siihen liitettyä koodia samalla ajaen sitä. (URL 3.4)

CoolBasic piirtää grafiikkaa *DirectX 7.0* –rajapintaa käyttäen. 3D-kiihdytys ei CoolBasicilla ole käytettävissä, joten pelien tulee käyttää 2D-grafiikkaa.

Äänikirjastona puolestaan toimii FMOD, joka on kaupallinen kirjasto useiden eri musiikki- ja ääniformaattien toistamiseen. Sitä saa käyttää ilmaiseksi, mikäli ei aio myydä tuotetta, jossa sitä käyttää. Näin ollen, mikäli kehittää CoolBasicilla ääniä tai musiikkia käyttävän pelin tai ohjelman, jota aikoo myydä, tulee ostaa lisenssi FMOD-kirjastoon. CoolBasicista itsestään ei tule mitään lisäkustannuksia, vaikka myisi tuotoksisiaan. Lisäksi ei tarvitse maksaa FMOD:n lisenssimaksua, mikäli myy sellaista peliä tai ohjelmaa, missä ei käytetä ääniä tai musiikkia.

3.2.1 Laitteistovaatimukset

Koodieditori

Internet Explorer 5. Suorittimena tulee olla vähintään Intel Pentium 166 mHz. CoolBasic toimii käyttöjärjestelmillä Windows 98, Me, 2000, XP tai uudempi. Näytön resoluutiona tulee olla vähintään 800x600 pikseliä. (URL 3.5)

CoolBasic-pelit

Suorittimena tulee olla vähintään Intel Pentium 350 mHz, mutta pelistä riippuen voi olla tarvetta myös tehokkaammalle suorittimelle. Keskusmuistia tarvitaan 64 Mb (Windows 98) tai 256 Mb (Windows XP). CoolBasic toimii käyttöjärjestelmillä Windows 98, Me, 2000, XP (ja Vista, ohjelma ajettava järjestelmävalvojan oikeuksilla). Näytönohjaimessa tulee olla vähintään 8 Mb muistia ja koneelle tulee olla asennettuna DirecX 7.0 tai uudempi. (URL 3.5)

Suositeltava kokoonpano

Peleissä on hyvä olla 500 Mhz –suoritin, mutta pelistä riippuen tarve voi olla suurempikin. Keskusmuistia tarvitaan 128 Mb (Windows 98) tai 512 Mb (Windows XP). Näytönohjaimessa on hyvä olla ainakin32 Mb muistia. Lisäksi suositusvarustukseen kuuluvat rullahiiri ja äänikortti. (URL 3.5)

3.2.2 Koodieditori

Koodieditori osaa värittää koodista avainsanoja eri väreillä. Värit on vieläpä mahdollista itse muuttaa haluamikseen. Editori ottaa koodista automaattisesti varmuuskopioita, ja sopivaa varmuuskopiota voi hätätapauksessa hakea juuri tätä varten olevalla *Recover Utility* – työkalulla. Sisäänrakennettu ohje samassa ikkunassa editorissa tarjoaa apua erittäin nopeasti. Kursorin ollessa tietyn komennon kohdalla, riittää että painat F1 niin manuaali aukeaa tämän komennon kohdalta. Funktioiden, tyyppien, taulukoiden, muuttujien ja vakioiden listaus. Kaikki eri ohjelman lohkot ovat näin ollen silmäiltävissä ja löydettävissä mahdollisimman nopeasti. (URL 3.2)

3.2.3 Kääntäjä

EXE-tiedostoon sisällytetään automaattisesti kaikki dll-tiedostot, joita ohjelmat tarvitsevat toimiakseen. Poikkeuksena kuitenkin sellaiset dll-tiedostot, jotka käyttäjä erikseen määrittelee koodissa. Tämä on kuitenkin harvinaista. Privaattimuuttujia ei tarvitse erikseen esitellä. Kääntäjää voi myös käskeä ottamaan tiukempi linja ja pakottamaan ohjelmoijaa esittelemään kaikki muuttujat. Kääntäjää voi käyttää suoraan komentoriviltä – editorin avaaminen tähän tarkoitukseen ei siis ole välttämätöntä. (URL 3.2)

3.2.4 Ydin

CoolBasic-kielessä on viisi eri tietotyyppiä, joita ovat kokonaisluvut, desimaaliluvut, lyhyet kokonaisluvut, tavut sekä merkkijonot. Kielessä on myös DATA-listat, joihin tietoa voidaan tallettaa suoraan koodissa sekä käyttäjän määrittelemät tyyppi-listat. Esimerkiksi ammuksista voi tehdä oman tyypin nimeltään *Ammus* ja sisällyttää siihen ammusten ominaisuuksia. Tyypit vastaavat linketettyjä listoja. Käyttäjä voi määritellä omia funktioita eripuolilla ohjelmaa toistuvien tehtävien tekemiseen. Myös rekursio onnistuu, eli funktio voi kutsua itseään. Kielessä on myös monipuoliset merkkijonokomennot sekä DLL-tuki. Voit käyttää ohjelmissasi omia DLL:iä, jotka tuovat lisäkomentoja.

Esimerkiksi nettipeli on mahdollista toteuttaa eräällä olemassa olevalla DLL:llä. (URL 3.2)

3.2.5 Pelimoottori

CoolBasicissa on nopea 2D-grafiikkamoottori ja liuta valmiita matematiikkafunktioita. Eri fonttien käyttäminen on mahdollista tekstin tulostamisessa näytölle. Kielessä on tuki myös TrueType- ja rasterifonteille. Kieli tukee erilaisia ääni- ja musiikkiformaatteja: MP3, OGG WAV, moduulimusiikki sekä cd:n soittaminen. Kieli tukee erilaisia kuvaformaatteja: BMP, JPG, TGA ja PNG. (URL 3.2)

CoolBasicissa on oma objektijärjestelmänsä, joka helpottaa esimerkiksi pelihahmojen liikkumisjärjestelmien tekoa. Nopea törmäyksentunnistus objektien kesken sekä objektien ja kentän välillä helpottavat myös peliohjelmointia. Objektijärjestelmä mahdollistaa myös pääsyn jokaiseen objektiin erikseen. (URL 3.2)

Piirtokomennot on mahdollista kohdistaa pelimaailmaan tai ikkunaan. Pelimaailmassa on myös liikuteltava kamera. CoolBasic huolehtii automaattisesti siitä, ettei ruudun ulkopuolelle piirtäminen aiheuta ongelmia. Kielessä on nopea tilekarttajärjestelmä, jolla on helppoa tehdä pelikenttiä. Kenttien reaaliaikainen muokkaus onnistuu kesken pelin. Nopea ja hyvin optimoitu partikkelijärjestelmä mahdollistaa erilaisten tehosteiden ja efektien luomisen. (URL 3.2)

4 CoolBasicin asennus ja käyttö

4.1 Asennus

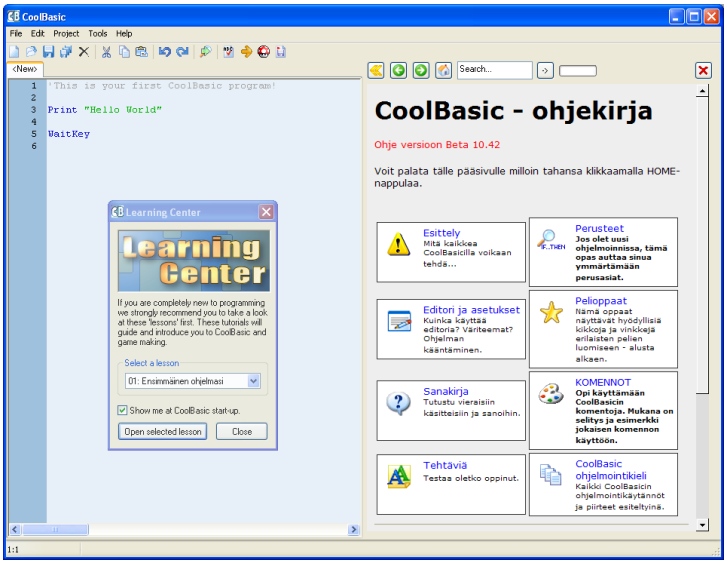
CoolBasic on ilmainen käyttää ja levittää, joten sen asentaminen ei maksa mitään. CoolBasicin käyttöehdoissa kuitenkin huomautetaan, että ohjelmiston asentaminen tapahtuu omalla vastuulla ja että CoolBasicin kehittäjät eivät ole vastuussa, jos ohjelmisto aiheuttaa haittoja. (URL 3.1, FILE 1)

Aloitetaan lataamalla CoolBasic sen kotisivuilta. Huomaa, että jos tämä dokumentti on vanha, on CoolBasicin sivusto saattanut muuttua vähän toisenlaiseksi. Mene WWW-selaimella osoitteeseen URL 1 ja klikkaa itsesi sisään suomenkielisille sivuille. Klikkaa yläpalkin Download-linkkiä ja valitse *”CoolBasic ja päivitykset”*. Valitse asennusohjelma CoolBasicin uusimmasta versiosta (kirjoitushetkellä *Beta 10.43*) ja lataa se tietokoneellesi, esimerkiksi työpöydälle.
(URL 1, URL 3.1)

Kun lataus on valmis, tuplaklikkaa ladattua asennusohjelmaa. Seuraa näytölle tulevia ohjeita ja asenna CoolBasic haluamaasi hekemistoon, esimerkiksi asennuksen ehdottamaan *Program Files* –kansioon. Asennusohjelma kysyy seuraavaksi kansiota käynnistysvalikossa, johon CoolBasicin pikakuvakkeet talletetaan. Kirjoita haluamasi kansio tai jatka asennusta asennusohjelman ehdottamalla nimellä. Kun asennus on valmis, voit alkaa käyttää CoolBasicia seuraavan kappaleen ohjeiden mukaisesti.

4.2 Koodieditorin käyttäminen

Avaa CoolBasicin mukana tullut editori, jos ei se ole vielä avattuna. Editori löytyy CoolBasicin asennushakemistosta nimellä *CBEditor.exe*. Ensimmäisellä käynnistyskerralla editori näyttää CoolBasicin manuaalin avattuna editorin oikealle puolelle sekä *"Learning Center"* –ikkunan, joka ehdottaa aloitteleville ohjelmoijille oppitunteja avattavaksi editoriin. Tässä vaiheessa *Learning Centerin* voi kuitenkin sulkea.



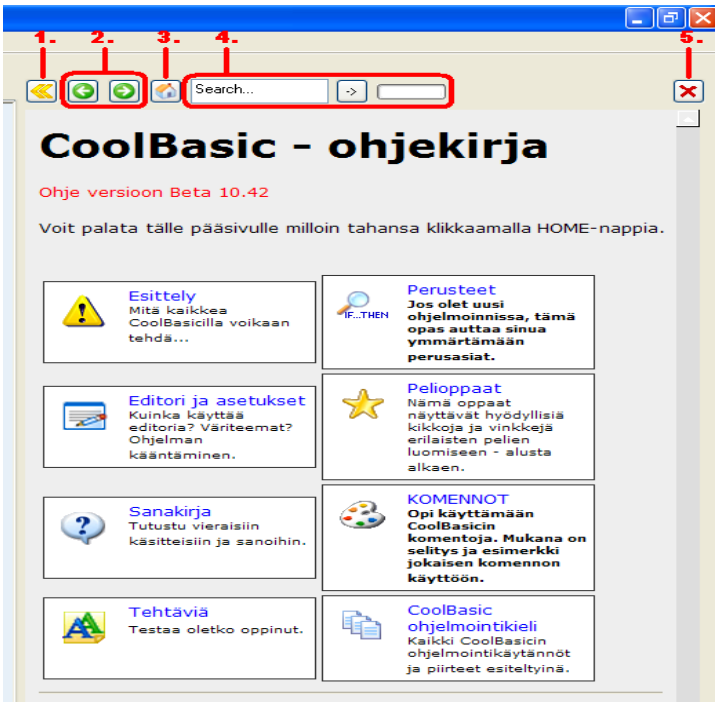
Ohjelmoiminen tapah-tuu luonnollisesti editorissa olevaan tekstikenttään, jossa on jo valmiina yksinkertainen ohjelma, jonka ajaminen tulostaa näytölle tekstin *"Hello World"*. Tässä kohtaa tutustumme kuitenkin ensin itse editorin toimintoihin ja vasta sen jälkeen koodin kirjoittamiseen ja ajamiseen.

Editorissa on sisäänrakennettu Html-pohjainen manuaali, joka on nyt esillä oikeassa laidassa. Sitä ei siis tarvitse avata erikseen toiseen ikkunaan ja se on vaivattomasti saatavilla. Olen numeroinut manuaalin lukemisessa käytettävät toiminnot oheisessa kuvassa:

1. Venyttää manuaalin koko ruudun kokoiseksi.
2. Voit siirtyä taaksepäin ja eteenpäin samoin kuin Internet-selaimessa.
3. Paluu manuaalin etusivulle.

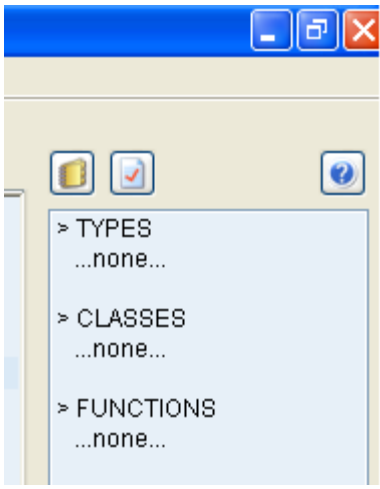
4. Haku. Voit hakea manuaalista tietoa tietyllä hakusanalla.

5. (Oikeassa ylänurkassa). Sulkee manuaalin pois tieltä kun et tarvitse sitä.



Katsomme pari ohjelmointia helpottavaa listaustoimintoa ja lopuksi säädämme editorin asetukset haluamasi kaltaisiksi.

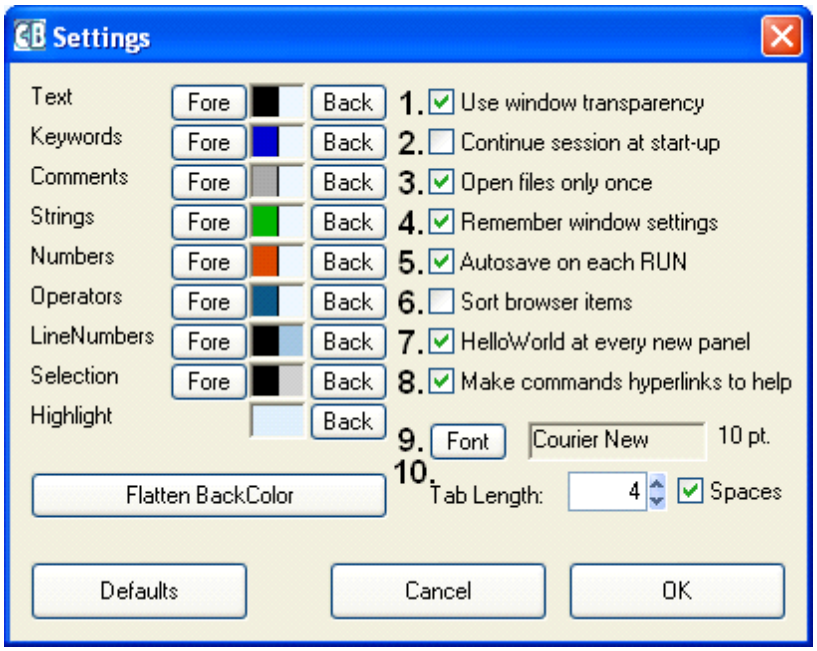
Sulje nyt CoolBasicin manuaali. Sen alta paljastuu Seuraavanlainen yksinkertainen listaustoiminto ja kolme uutta nappulaa:



Ensimmäinen nappula vasemmalta listaa kaikki lähdekoodissasi olevat *Tyyppi-*, *Luokka-* ja *Funktio-rakenteet*. Näihin tulet törmäämään ohjelmointioppaissa. Tosin CoolBasicissa (Beta 10.43) ei vielä ole Luokka-rakenteita, vaikka listaus huomioisi kyllä nekin. Toisen painikkeen klikkaaminen listaa kaikki Kirjanmerkit, Vakiot, Globaalit muuttujat, Taulukot ja Yksityiset muuttujat.
Kuitenkaan lähdekoodissa ei vielä ole mitään näistä mainituista asioista, joten sen vuoksi sellaisia ei vielä näy listauksessa. Jos listauksessa näkyisi jotakin, listuksessa olevan tietueen klikkaaminen veisi tekstikursorin koodiriville, jossa kyseinen rakenne tai tietotyyppi on esitelty.
Viimeinen painike oikealla avaa jälleen manuaalin.

Vielä viimeisenä asiana on vilkaisu editorin asetuksiin. Valitse valikosta *File -> Preferences*. Asetusikkunan vasemmasta laidasta voi säätää ns. syntaksiväritystä, eli miten eri avainsanat väritetään koodissa. Värien säätäminen ei kuitenkaan välttämättä ole tarpeellista, joten en tässä ohjeessa keskity siihen.

Sen sijaan käymme läpi ikkunan oikeanpuoleiset asetukset järjestyksessä ylhäältä alas. Oheisessa kuvassa asetukset ovat oletusarvojensa mukaisesti.



1. Osittain läpinäkyvät ikkunat editorissa.
2. Muistaa auki jääneet tiedostot ja avaa ne seuraavilla käynnistys-kerroilla.
3. Varmistus ettei samasta tiedostosta avata editoriin useaa kopiota.
4. Muistaa ikkunan koon.
5. Tallenna automaattisesti jokaisen ajon yhteydessä.
6. Lajittelee listaustoiminnon näyttämät rakenteet ja tietotyypit.
7. Avataanko jokaiseen uuteen välilehteen pieni *HelloWorld*-tekstin tulostava koodi.
8. Tekee komennoista tuplaklikattavia linkkejä, jotka johdattavat juuri haluamasi komennon ohjeeseen manuaalissa.
9. Lähdekoodin fontti.

10. Kuinka monen välilyönnin kokoinen yksi sarkain on ja muutetaanko sarkaimet välilyöntimerkeiksi, sarkainmerkin sijaan.

Kun olet säätänyt asetukset haluamiksesi on aika painaa OK:ta ja siirtyä seuraavaan kappaleeseen, jossa pääsemme jo koodaamaan. Ja asetuksien muuttaminenhan käy toki myöhemminkin, jos et vielä niihin halunnut koskea.

4.3 Ohjelmoinnin oppitunnit

CoolBasicissa pelit luodaan siis kenttiä, kuvia, ääniä, musiikkeja ja muita medioita lukuun ottamatta pelkästään kirjoittamalla koodia. Graafisia editoreita ei ainakaan vielä ole ohjelman rakenteen suunnitteluun. Jos osaat jo ohjelmoida kohtalaisen hyvin entuudestaan, ei sinun välttämättä tarvitse käydä tätä kappaletta läpi. CoolBasicin manuaali on siltikin hyvä tuttavuus sinulle, sillä sieltä löytyy tietoa kielen eri komennoista, tietotyypeistä ja muusta.

Valitse valikosta *Help -> Learning Center*, ellei se jo ole avoinna. Learning Center ehdottaa avattavaksi oppitunniksi luonnollisesti ensimmäistä oppituntia, ja se me valitsemmekin. Klikkaa *Open selected lesson* ja editoriin aukeaa pieni ohjelman koodi, jonka mukana seuraa suuri määrä koodia selostavaa kommenttia. Kommentti on siis tekstiä, jota ei huomioida mitenkään ohjelmaa käännettäessä. Oppitunnin aiheena on perinteinen *Hello World!* –tekstin tulostava koodi. Aja ohjelma ja lue oppitunnin laatijan kirjoittama analyysi, niin opit miten miten tämä koodi toimii.

Jatka oppituntien läpikäymistä omaan tahtiisi. Voit aina kokeilla muokata oppituntien koodeja ja mitä enemmän uusia komentoja opit, sitä ennemmän tulet saamaan iloa irti ohjelmoimisesta!

4.4 Tilester-karttatyökalu

Tilester on CoolBasicin mukana tuleva ohjelma pelien karttojen ja kenttien luomiseksi. Sen käyttäminen on helppoa ja yksinkertaista kun kentissä käytetään Tilesterin mukana tulevia grafiikoita. Jos halutaan tehdä kokonaan omaa grafiikkaa kenttiin, on omattava taito piirtää tietokoneella edes kohtalaisesti. (URL 4.1, URL4.2)

Tilesterillä luodaan kenttiä piirtämällä niitä hiirellä erilaisista palasista, joita kutsutaan *tileiksi*. Yksi tile voi sisältää esimerkiksi palan puulattiaa ja toinen vaikkapa tuolin tai muun esineen. Kolmannessa voi olla pala seinää tai muuta estettä. (URL 4.2, URL 4.3, URL 4.4)

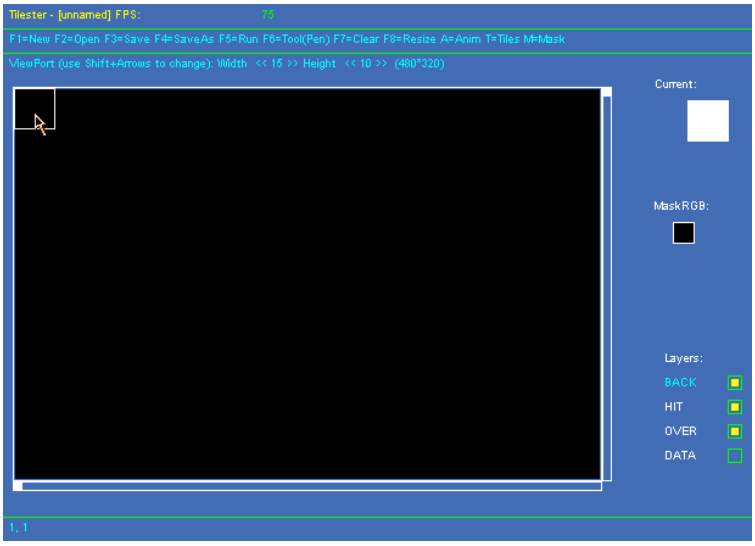
Tilejä voidaan piirtää kahteen eri kerrokseen: *back*- ja *over*-kerroksiin. *Back*-kerrokseen tulee kentän tausta. Oletetaan, että olemme tekemässä kenttää ylhäältä päin kuvattuun peliin, jolloin tausta voi olla vaikkapa lattiaa, ruohikkoa tai katua. Sivultapäin kuvatussa pelissä tausta olisi yleensä joko taivasta tai rakennuksen seinää. *Over*-kerrokseen taas tulee taustan päälle asetettavat tilet, kuten esimerkiksi huonekalut ja seinät, sekä sellaiset asiat, joiden alle tai taakse pelaaja, viholliset ja muut hahmot ja objektit voivat mennä kameralta piiloon, eli jotka piirretään objektien päälle. (URL 4.2, URL 4.3, URL 4.4)

Huonekalut ja seinät on toki mahdollista piirtää *over*-kerroksen sijaan *back*-kerrokseen, mutta tällöin taustan on oltava samassa tilessä. *Over*-kerrokseen piirrettäessä, kenttä käyttää tileissä läpinäkyvää maskiväriä (oletuksena musta, mutta voidaan vaihtaa), jolloin tausta näkyy tilen alta läpi niistä kohdista, missä maskiväriä on käytetty. (URL 4.2, URL 4.3, URL 4.4)

Kentissä on vielä *hit*-ja *data*-kerrokset. *Hit*-kerrokseen piirretyt tilet eivät varsinaisesti näy itse pelissä. Sen sijaan sitä käytetään merkitsemään kentässä kohdat, joista objektien (pelaaja, viholliset, ammukset jne.) ei tule päästä läpi, esimerkiksi seinät ja huonekalut. *Data*-kerros on edistyneemmille käyttäjille. Siinä jokaiselle tilelle voidaan antaa mikä tahansa kokonaisluku. Näin voidaan asettaa tiettyihin kohtiin kentässä vaikka ansoja. Data-kerroksesta ei tarvitse välittää ollenkaan, ennen kuin itse tunnet tarvetta sen käyttämiseen. (URL 4.2, URL 4.3, URL 4.4)

Tilester tallentaa kentän *.til*-päätteiseen tiedostoon. Se siis sisältää itse kentän kaikki yksityiskohdat, mutta ei lainkaan grafiikkaa. Tämän lisäksi on *tileset*:n sisältävä grafiikkatiedosto (esimerkiksi *.bmp*- tai *.png*-päätteinen) joka sisältää kaiken kentässä käytettävän grafiikan. Oletuksena käytettävä tileset löytyy Tilesterin kansiosta *C:\Program Files\Tilester* nimellä *Default.bmp*. Tämä tulee kopioida kentän mukaan kun kenttää käytetään pelissä. (URL 4.2, URL 4.4)

Aloitetaan käynnistämällä Tilester CoolBasicin *Tools*-valikosta. Tilesterin päänäkymä aukeaa kokonäytölle.



Musta laatikko kuvastaa kenttää, joka on tällä hetkellä tyhjä. Klikkaa oikealla olevaa neliötä *Current*-kohdan alapuolella. Tilester näyttää nyt oletustilesetin. Valitse tilesetistä hiirellä tile, jolla haluat piirtää. Voit myös valita kerralla useampia tilejä raahaamalla hiirtä. Tämän jälkeen

klikkaa hiiren oikealla painikkeella, jolloin Tilester palaa jälleen päänäkymäänsä. (URL 4.5)

Ruudun oikeassa reunassa on myös näkyvissä kentän kerrokset (*layers*). Keltainen neliö kunkin kerroksen lopussa olevassa ruudussa tarkoittaa, että kerros on näkyvillä. Voit siis klikkailla tarvittaessa eri kerroksia näkyviksi ja piilotetuiksi editorissa. Klikkaamalla kerroksen nimeä pääset muokkaamaan kyseistä kerrosta. Klikkaa siis *over*-kerrosta ja valitse jälleen tile, jolla haluat piirtää. Tilesterin päänäkymässä hiiren oikea korva toimii pyyhekumina. Tämän jälkeen voit piirtää *hit*-kerrokseen – tilesetin vasemman yläkulman tilellä - kohdat, joihin on mahdollista törmätä. (URL 4.5)

Nyt olemme käyneet Tilesterin perusteet läpi. Jos haluat tallentaa tekemäsi kentän, paina F4. Tilester kysyy, mitä muotoa käytetään – 1.0 vai 1.3. Vaikka siinä sanotaan, että 1.3 on *CoolBasic Final:lle*, valitse se silti. CoolBasicista ei vielä ole *Final*-versiota, mutta tuo 1.3 toimii paremmin CoolBasicin nykyisessä versiossa. (URL 4.1)

Kenttä ladataan peliin *LoadMap()*-funktiolla, joka on CoolBasicissa sisäänrakennettuna. Funktio ottaa parametreikseen: *kenttätiedoston polku* **,** *tileset-tiedoston polku*. Molemmat polut voidaan antaa joko kanoonisessa tai absoluuttisessa muodossa (kanooninen: *media**kenttä.til*; absoluuttinen: *c:**peli**media**kenttä.til*). Kanooninen on suositeltavampi, jotta peli toimisi myös koneissa, joissa se asennetaan eri hakemistoon kuin alkuperäisellä koneella. Seuraavassa on pieni esimerkki kentän lataamisesta sekä sillä liikkuvasta hahmosta. Koodi toimii suoraan kopioimalla se uudelle välilehdelle CoolBasicissa. Rivien alussa olevia rivinumeroita ei kopioida. Koodi näyttää selkeämmältä kun olet kopioinut sen CoolBasiciin.

```
1                   'Asetetaan ikkunan leveys ja korkeus.
2                   SCREEN 640,480
3
```

```
4                   'LADATAAN PELIIN KENTTÄ
5                   kenttä = LoadMap("media\testmap.til", "media\tileset.bmp")
6
7                   'Ladataan peliin ukkeli. Tästä sinun ei tarvitse nyt välittää.
8                   ukkeli = LoadObject("media\guy.bmp",72)
9                   PositionObject ukkeli, 100,0
10
11                  'Asetetaan ukkelin ja kentän välinen törmäyssuhde
12                  SetupCollision ukkeli, kenttä, 1,4,2
13
14                  'Pyöritetään peliä silmukassa, jottei se lopu heti.
15                  Repeat
16
17                      'Pelaajan liikkuminen (tätäkään sinun ei tarvitse vielä
täysin ymmärtää)
18                      nopeus = 2
19                      kääntyminen = 3
20                      If KeyDown(cbKeyUp) Then MoveObject ukkeli, nopeus
'Liikutetaan eteenpäin
21                      If KeyDown(cbKeyDown) Then MoveObject ukkeli, -nopeus
'Liikutetaan taaksepäin (nopeus on negatiivinen).
22                      If KeyDown(cbKeyRight) Then TurnObject ukkeli, -
kääntyminen 'Käännetään ukkelia myötäpäivään (kääntyvyys on negatiivinen).
23                      If KeyDown(cbKeyLeft) Then TurnObject ukkeli, kääntyminen
'Käännetään ukkelia vastapäivään.
24
25                      'Tämä piirtää kentän ja pelaajan ruudulle.
26                      'Komento samalla päivittää pelissä ukkelin ja kentän
väliset törmäykset.
27                      DrawScreen
28
29                  Until KeyHit(cbKeyEscape) 'Ohjelma päättyy Esc-näppäimestä.
```

Ohjelman hahmo liikkuu nuolinäppäimistä ja törmää kentän seiniin. Voit myös kokeilla avata *testmap.til*-kentän Tilesterissä.

5 Esimerkkipelin tekeminen CoolBasicilla

Teemme tässä osassa yksinkertaisen *Pong*-pelin, jossa pelaaja liikkuu vasemmalle ja oikealle ja yrittää saada liikkuvan pallon kimpoamaan ikkunan yläreunaan. Vastassaan pelaajalla on tietokoneen ohjaama vihollinen, joka yrittää saada pallon ikkunan alalaitaan.
Pelimme koostuu siis pallosta ja kahdesta liikkuvasta palkista. Ruudulla näkyy myös pistetilanne ja pallon osuessa johonkin, kuuluu ääni.

Taustalla soi musiikki. Kaikki äänet ja musiikki löytyvät CoolBasicin juuressa sijaitsevasta *Media*-hakemistosta. Kirjoitamme pelin koodin niin, että se osaa hakea kaiken tarvittavan automaattisesti tästä *Media*-kansiosta. Voit tallentaa pelin lähdekoodin aina välillä CoolBasicin juureen esimerkiksi nimellä *Pong.cb*.

5.1 Ohjattava pelaaja

Pelaajaa merkitään siis yhdellä palkilla. Ensimmäiseksi luomme kuvan palkista, jota voimme käyttää pelaajan – sekä myöhemmin myös tietokonevastustajan – piirtämiseen. Palkin leveyden ja paksuuden määritämme vakioiksi ennen palkin luomista.

```
1                   'Asetetaan ikkunan koko
2                   SCREEN 640,480
3
4                   //Määritetään pelin asetukset
5                   Const PALKIN_LEVEYS = 100
6                   Const PALKIN_PAKSUUS = 5
7
8
9                   //Luodaan mallikuva palkista
10
11                  'Tämä luo kuvan muistiin
12                  palkki = MakeImage(PALKIN_LEVEYS, PALKIN_PAKSUUS)
13
14                  'Ohjataan piirtäminen luomaamme kuvaan
15                  DrawToImage palkki
16
17                  'Piirretään palkki
18                  Color cbDarkRed
19                  Box 0,0, PALKIN_LEVEYS,PALKIN_PAKSUUS, ON
20
21                  'Ohjataan piirtäminen takaisin näytölle
22                  DrawToScreen
23
```

Koodissa *MakeImage()*-funktio luo kuvan ja palauttaa sen *palkki*-muuttujaan. Piirrämme tähän kuvaan *Box*-komennolla, joka ottaa parametrikseen palkin vasemman yläkulman koordinaatit (0,0 kuten

kuvan vasen yläkulma), palkin leveyden ja korkeuden (määrittämämme vakiot) sekä piirretäänkö täytetty laatikko (ON, eli kyllä).
Nyt kun olemme saaneet kuvan valmiiksi, teemme lenkin, joka tulee kohta pyörittämään koko peliä. Samalla teemme pelaajalle liikkumisen. Kirjoitamme seuraavat koodirivit edellisten perään:

```
24                  'Asetetaan pelaaja vaakasuunnassa keskelle ruutua ja
pystysuunnassa lähelle ikkunan alalaitaa
25                  pelaajaX = (ScreenWidth() - PALKIN_LEVEYS)/2
26                  pelaajaY = ScreenHeight()-30
27
28                  //Pelin päälenkki
29
30                  Repeat
31
32                       //Pelaajan ohjaus
33
34                       'Nuolet vasemmalle ja oikealle
35                       If KeyDown(cbKeyLeft) Then pelaajaX = pelaajaX -
PALKIN_NOPEUS
36                       If KeyDown(cbKeyRight) Then pelaajaX = pelaajaX +
PALKIN_NOPEUS
37
38                       'Tarkistetaan, ettei pelaaja mene ulos ruudusta
39                       If pelaajaX < 0 Then pelaajaX = 0
40                       If pelaajaX > ScreenWidth() - PALKIN_LEVEYS Then pelaajaX
= ScreenWidth() - PALKIN_LEVEYS
41
42
43                       //Piirretään pelaaja
44                       DrawImage palkki, pelaajaX,pelaajaY
45
46                       'Päivitetään kuva näytölle
47                       DrawScreen
48
49                  Forever
```

Lisäksi lisäämme koko koodin alkuun - heti rivin 6 alapuolelle - seuraavan rivin:

```
7                   Const PALKIN_NOPEUS = 5
```

Nyt koodimme suorittaa silmukkaa niin kauan, kunnes käyttäjä sulkee ikkunan tai painaa *Esc*-näppäintä. Silmukassa tarkistamme *KeyDown()*-funktiolla, onko vasen tai oikea nuolinäppäin painettuna ja liikutamme pelaajaa tarvittaessa *NOPEUS*-vakiossa määritetyn pikselimäärän mukaisesti muuttamalla pelaajan x-koordinaattia.
Lisäksi tarkistamme, ettei uusi x-koordinaatti vie pelaaja ikkunan ulkopuolelle. Ikkunan oikea laita on siitä erikoinen, että x-koordinaatin ja oikean laidan väliin on jäätävä palkin leveyden pituinen matka. Tämä johtuu siitä että koordinaatit osoittavat palkin vasemman yläkulman.
Lopuksi piirrämme palkin näihin koordinaatteihin ja päivitämme kuvan näytölle. Tässä vaiheessa koodimme näyttää tältä:

```
1                   'Asetetaan ikkunan koko
2                   SCREEN 640,480
3
4                   //Määritetään pelin asetukset
5
6                   Const PALKIN_LEVEYS = 100
7                   Const PALKIN_PAKSUUS = 5
8                   Const PALLON_HALKAISIJA = 10
9                   Const PALKIN_NOPEUS = 5
10
11
12                  //Luodaan mallikuva palkista
13
14                  'Tämä luo kuvan muistiin
15                  palkki = MakeImage(PALKIN_LEVEYS, PALKIN_PAKSUUS)
16
17                  'Ohjataan piirtäminen luomaamme kuvaan
18                  DrawToImage palkki
19
20                  'Piirretään palkki
21                  Color cbDarkRed
22                  Box 0,0, PALKIN_LEVEYS,PALKIN_PAKSUUS, ON
23
24                  'Ohjataan piirtäminen takaisin näytölle
25                  DrawToScreen
26
27
28                  'Asetetaan pelaaja vaakasuunnassa keskelle ruutua ja
pystysuunnassa lähelle ikkunan alalaitaa
29                  pelaajaX = (ScreenWidth() - PALKIN_LEVEYS)/2
```

```
30                  pelaajaY = ScreenHeight()-30
31
32                  //Pelin päälenkki
33
34                  Repeat
35
36                       //Pelaajan ohjaus
37
38                       'Nuolet vasemmalle ja oikealle
39                       If KeyDown(cbKeyLeft) Then pelaajaX = pelaajaX -
PALKIN_NOPEUS
40                       If KeyDown(cbKeyRight) Then pelaajaX = pelaajaX +
PALKIN_NOPEUS
41
42                       'Tarkistetaan, ettei pelaaja mene ulos ruudusta
43                       If pelaajaX < 0 Then pelaajaX = 0
44                       If pelaajaX > ScreenWidth() - PALKIN_LEVEYS Then pelaajaX
= ScreenWidth() - PALKIN_LEVEYS
45
46
47                       //Piirretään pelaaja
48                       DrawImage palkki, pelaajaX,pelaajaY
49
50                       'Päivitetään kuva näytölle
51                       DrawScreen
52
53                  Forever
```

5.2 Liikkuva pelipallo

Pelipallo liikkuu siis kokoajan vakionopeudella. Se kimpoaa pelaajan tai tietokonevastustajan (jota emme ihan vielä tee) liikuttamista palkeista, sekä ikkunan vasemmasta ja oikeasta laidasta. Pallon osuessa ikkunan ylä- tai alalaitaan, vastapäinen pelaaja saa pisteen ja pallo siirretään takaisin ruudun keskelle. Pisteidenlaskun ja pallon siirtämisen tosin toteutamme vasta kun tietkonepelaaja on luotu.

Ensimmäiseksi asetamme palloa varten pari vakiota *Pelin asetukset*-kohtaan koodimme alussa. Näillä vakioilla asetamme pallolle halkaisijan sekä nopeuden:

```
10                  Const PALLON_HALKAISIJA = 10
11                  Const PALLON_NOPEUS = 4
```

Seuraavaksi luomme kuvan pallosta. Tämä koodi lisätään rivin 25 jälkeen:

```
26 //Luodaan kuva pallosta
27 pallo = MakeImage(PALLON_HALKAISIJA,PALLON_HALKAISIJA)
28 DrawToImage pallo
29 Color cbBlue
30 Circle 0,0, PALLON_HALKAISIJA, ON
```

Pallon koordinaatit säilytetään *palloX*- ja *palloY*-muuttujissa. Pelin alussa pallo sijoitetaan keskelle kenttää, joten lisäämme seuraavan koodin rivin 41 jälkeen:

```
42 'Asetetaan pallo keskelle ruutua
43 palloX = ScreenWidth()/2
44 palloY = ScreenHeight()/2
45 palloOlemassa = True
46
47 'Asetetaan pallolle suunta. Suunta voi olla liukuluku väliltä 0.0 – 360.0.
48 '#-merkki tekee muuttujasta liukulukutyyppisen.
49 palloSuunta# = Rnd(360)
```

Rivillä 49 asetamme myös pallolle sattumanvaraisen suunnan muuttujaan *palloSuunta*. Suunta voi olla desimaaliluku väliltä 0 – 360. *Rnd()*-funktio arpoo sattumanvaraisen desimaaliluvun.

Rivillä 45 asetamme myös uuden muuttujan nimeltä *palloOlemassa* ja annamme sille arvoksi *True*. Tämä muuttuja sisältää aina tiedon siitä, onko pallo olemassa kentällä, vai ei. Palaamme tähän muuttujaan myöhemmin.

Nyt siirrymme kirjoittamaan koodia, joka liikuttaa pelipalloa. Koska pallon suunta ilmoitetaan astelukuna väliltä 0 – 360, käytämme pallon liikuttamiseen trigonometrisiä *Cos()*- ja *Sin()*-funktioita. Mainittakoon

että *Cos()*-funktio laskee pallon liikkeen vaakasuunnassa, kun pallon suunta annetaan sille parametrina ja pallon nopeus on 1. Samaan tapaan *Sin()*-funktio laskee pallon liikkeen pystysuunnassa. Jotta saamme liikkeen nopeuden muutettua haluamaksemme, kerromme kummastakin funktiosta saamamme tulokset haluamallamme nopeudella, joka on talletettuna *PALLON_NOPEUS*-vakioon. Seuraava koodi lisätään rivin 68 jälkeen:

```
69
70                  //Pelipallon liikkuminen
71
72                  'Liikutetaan palloa Sinin ja Cosinin avulla
73                  palloX = palloX + Cos(palloSuunta) * PALLON_NOPEUS
74                  palloY = palloY - Sin(palloSuunta) * PALLON_NOPEUS
```

Cos() ja *Sin()* siis ilmaisevat muutokset pallon koordinaatteihin, eivätkä tässä tapauksessa absoluuttisia koordinaatteja. Näin ollen lisäämme niiden tulokset pallon nykyisiin koordinaatteihin.

Huomautuksena kuitenkin, että koska pelimme piirtää näytölle käyttäen *ikkunan koordinaatteja*, tulee *palloY*-muuttujasta vähentää *Sin()*:n antama tulos, sillä y-koordinaatti alkaa ikkunan yläreunasta ja suurenee kohti ikkunan alareunaa. (Samaan tapaan x-koordinaatti alkaa ikkunan vasemmasta reunasta ja suurenee kohti ikkunan oikeaa reunaa).

Normaalisti trigonometriset funktiot käyttävät koordinaatteja, joissa y-koordinaatti alkaa ikkunan keskeltä ja suurenee ikkunan yläreunaa kohti, samoin kuin x-koordinaatti alkaa ikkunan keskeltä ja suurenee ikkunan oikeaa laitaa kohti. (Näitä koordinaatteja kutsutaan *pelimaailman koordinaateiksi*, mutta kuten aiemmin totesin, emme tässä pikkupelissä käytä niitä). Tästä koordinaattijärjestelmien erosta johtuen käännämme y-koordinaattiin tulevan muutoksen päinvastaiseksi, jolloin tulos vastaa *ikkunan koordinaatteja*. Mikäli et ymmärtänyt tätä, voit opiskella asiasta lisää esimerkiksi CoolBasicin manuaalista *Sin()*- ja *Cos()*-funktioiden kohdalta sekä oppitunneista.

Nyt kun pallomme liikkuu, haluamme sen myös kimpoavan kun se osuu pelaajaamme. Ensimmäinen tehtävämme tämän toteutuksessa on saada peli tunnistamaan pallon törmääminen. Tätä varten teemme päälenkin ulkopuolella funktion nimeltä *DotInArea()*, joka tarkistaa, ovatko tietyt koordinaatit suorakaiteen muotoisen alueen sisällä, vai eivät. Funktio palauttaa *True* tai *False* tuloksen mukaan. Luodaan funktio päälenkin jälkeen, eli *Forever*-lauseen jälkeen:

```
Function DotInArea(dotX,dotY, aX1,aY1, aX2,aY2)
    If dotX >= aX1 And dotX <= aX2 Then
        If dotY >= aY1 And dotY <= aY2 Then
            Return True
        End If
    End If
    Return False
End Function
```

Funktio ottaa siis parametreikseen tarkistettavan pisteen x- ja y-koordinaatit sekä tarkistusalueen koordinaatit, joissa *aX1, aY1* osoittaa alueen vasemman ylänurkan ja *aX2, aY2* osoittaa alueen oikean alanurkan. Funktio tarkistaa, onko annettu piste näiden raja-arvojen sisällä, vai ei.

Käsittelemme kohta tämän funktion kutsumisen, mutta sitä ennen luomme vielä toisen funktion. Koska toinen tehtävämme on reagoida törmäykseen laittamalla pallo kimpoamaan, voimme luoda tätä varten funktion, joka laskee pallolle uuden suunnan. Seuraava funktio luodaan edellisen funktion perään, eli *End Function*-rivin jälkeen.

```
Function BounceAngle(angle1#,angle2#)
    Return WrapAngle(angle2-(angle1-angle2))
End Function
```

Funktiomme ottaa siis parametreikseen kaksi kulmaa (=suuntaa), joista ensimmäinen (*angle1#*) on pallon suunta ja toinen (*angle2#*)

törmäyskohteen kulma, eli pelaajan palkin suunta. Funktio peilaa *angle1#*:n ja palauttaa saamansa tuloksen, joka asetetaan pallon uudeksi suunnaksi.

Nyt kun meillä on funktiot valmiina törmäyksen tarkistukseen ja pallon kimpoamiseen, voimme kirjoittaa näitä funktioita hyödyntävän koodin pelin päälenkkiin, rivin 74 jälkeen:

```
75
76                  'Tarkistetaan törmäykset
77                  If DotInArea(palloX,palloY, pelaajaX,pelaajaY, pelaajaX+
PALKIN_LEVEYS-1,pelaajaY+PALKIN_KORKEUS-1) Then
78                      'Pallo on törmännyt pelaajaan
79                      'Laitetaan pallo kimpoamaan palkista, kun palkin kulma on
180
80                      palloSuunta = BounceAngle(palloSuunta,180)
81                  End If
```

Ehtolauseemme *If DotInArea(...) Then* siis huomaa, mikäli pallo osuu pelaajaan, jolloin lauseke *palloSuunta = BounceAngle(...)* saa pallon kimpoamaan antamalla pallolle uuden suunnan.
Annamme *BounceAngle()*-funktiolle toisena kulmana 180, sillä pelaajan palkki on aina vaakasuunnassa. Toinen mahdollisuus olisi laittaa kulmaksi 0, mikä tarkoittaisi tässä tapauksessa myös vaakasuoraa palkkia. Jos siis haluat kokeilla, voit muuttaa 180:n nollaksi.

Nyt kun pallomme osuu pelaajaan, laitamme sen osumaan myös ikkunan vasempaan ja oikeaan reunaan. Reunoista pallo kimpoaa samaan tapaan kuin pelaajasta.

```
82
83                  If palloX < 0 Then
84                      'Pallo on törmännyt ruudun vasempaan reunaan
85                      'Laitetaan pallo kimpoamaan seinästä, jonka kulma on 90
86                      palloSuunta = BounceAngle(palloSuunta,90)
87                  End If
88
89                  If palloY > ScreenWidth() Then
```

```
90                      'Pallo on törmännyt ruudun oikeaan reunaan
91                      'Laitetaan pallo kimpoamaan seinästä, jonka kulma on 90
92                      palloSuunta = BounceAngle(palloSuunta,90)
93                  End If
```

Rivit 83 – 87 hoitavat siis pallon kimpoamisen ikkunan vasemmasta reunasta, ja rivit 89 – 93 taas hoitavat kimpoamisen ikkunan oikeasta reunasta.
Mikäli rivillä 83 havaitaan, että pallon x-koordinaatti on alle nollan, se tarkoittaa, että pallo on poistunut ikkunasta, sillä ikunnassahan koordinaatit alkavat nollasta. CoolBasicissa on toki mahdollista piirtää myös ruudun ulkopuolelle esimerkiksi käyttämällä negatiivisia koordinaatteja, mutta tällöin ruudulle ei tietenkään tule mitään, mikä jää ikkunan rajojen ulkopuolelle. Tämän vuoksi haluamme saada pallon takaisin ikkunaan.
Selvittääksemme, onko pallo osunut ikkunan oikeaan reunaan – eli mennyt sen ohi, on meidän ensin tiedettävä, kuinka leveä ikkunamme on. *ScreenWidth()*-funktio palauttaa ikkunan leveyden pikseleissä; ja jos pallon x-koordinaatti on enemmän kuin ikkunan leveys, tiedämme pallon menneen reunan yli ja voimme laittaa sen kimpoamaan.

Nyt olemme saaneet pallon liikkumisen suurimmaksi osaksi valmiiksi. Tämä luku on lähes lopussa, ja voimme kohta siirtyä tekemään tietokonevastustajaa. Tätä ennen kirjoitamme kuitenkin vielä koodin, jolla pallo luodaan takaisin kentälle kun se on ensin hävinnyt kentältä ja kun painetaan *Enter*-näppäintä. Kirjoitamme seuraavan koodin päälenkkiin, edellisten koodien perään:

```
94
95                  'Sijoitetaan pallo kentän keskelle, jos pallo ei ole kentällä,
ja jos painetaan Enteriä.
96                  If KeyHit(cbKeyReturn) And palloOlemassa=False Then
97                      palloOlemassa = True
98                      palloSuunta = Rnd(360)
99                      palloX = ScreenWidth()/2
100                     palloY = ScreenHeight()/2
101                 End If
```

Rivillä 97 asetamme muuttujan *palloOlemassa* arvoksi *True*, jolloin pallon sijoittaminen takaisin ruudun keskelle ei enää tämän kerran jälkeen onnistu, ennen kuin pallo taas poistuu kentältä.
Riveillä 98 – 100 arvomme pallolle uuden suunnan ja asetamme sen keskelle kenttää.

Tulevaisuudessa meille tulee siis tilanne, jossa pallo on poistunut joko tietokonevastustajan maaliin tai meidän omaan maaliimme. Tällöin muuttujan *palloOlemassa* arvoksi tulee *False* (mutta siitä huolehdimme myöhemmin). Pallo sijoitetaan uudestaan kentälle vasta, kun pelaaja painaa *Enter*-näppäintä.
Kun pallo ei ole kentällä, on turha liikuttaa sitä tai tehdä sille törmäystarkistuksia. Näin ollen teemme pari pientä lisäystä koodimme; saadaksemme pallon liikkumisen ja törmäystarkistukset toimimaan vain silloin, kun pallo on kentällä.
Lisäämme seuraavan ehtolauseen rivin 69 jälkeen:

```
70                  If palloOlemassa Then
```

Ja tälle ehtolauseelle lisäämme päätöslauseen rivin 94 jälkeen:

```
95                  End If
```

Näiden kahden rivin välissä olevan koodin sisennämme yhden pykälän verran oikealle, jolloin huomaamme selkeästi, että näiden rivien välissä oleva koodi suoritetaan vain, kun pallo on kentällä.

Tässä luvussa olemme siis saaneet pallon liikkumaan ja kimpoamaan seinistä sekä pelaajasta. Lisäksi olemme tehneet pallon luomisen takaisin kentälle, kun se on hävinnyt – tätä toimintoa pääsemme hyödyntämään seuraavassa luvussa, jossa pallo tulee myös katoamaan kentältä. Liitteessä *LIITE 1* on koko tähänastisen pelimme koodi.

5.3 Tietokonevastustaja

Tietokonevastustajamme tulee pyrkimään liikkumaan aina vaakasuunnassa palloa kohti. Aloitamme vastustajan tekemisen luomalla sille koordinaattimuuttujat pelaajan koordinaattimuuttujien jälkeen ennen pelin päälenkkiä. Seuraava koodi tulee siis rivin 41 jälkeen:

```
42
43                  'Asetetaan tietokone vaakasuunnassa keskelle ruutua ja
pystysuunnassa lähelle ikkunan ylälaitaa
44                  tietokoneX = (ScreenWidth() - PALKIN_LEVEYS)/2
45                  tietokoneY = 30
```

Tietokoneen x-koordinaatiksi tulee siis sama kuin pelaajan x-koordinaaksi, eli tietokone sijoittuu vaakasuunnassa keskelle ruutua. Pystysuunnassa tietokone sijoittuu 30:n pikselin päähän ikkunan yläreunasta.

Seuraavaksi on vuorossa tietokoneen liikkuminen. Kuten ylempänä mainitsin, teemme tietokoneesta sellaisen, että se pyrkii pysymään vaakasuunnassa pallon kohdalla. Se kuitenkin tulee liikkumaan vain, mikäli vaakasuunnassa etäisyyttä palloon on tietokoneen palkin keskipisteestä yli *n* pikseliä. *n* on sattuman varaisesti arvottava luku, joka voi olla jotakin väliltä 30 – 60. Tekemällä arvosta satunnaisen, saamme tietokoneen reagoimaan välillä nopeammin ja välillä hieman hitaammin.

Lisäksi teemme niin, että tietokone liikkuu vain, mikäli pallo on pystysuunnassa alle 200:n pikselin päässä tietokoneesta, ja mikäli pallo on tulossa tietokonetta kohti, eli sen suunta on alle 180. (Pallon suunnan ollessa alle 180, on pallo menossa ylöspäin).
Kirjoitamme seuraavan koodin pelin päälenkkiin pelaajan liikkumisen ja piirtämisen jälkeen, eli rivin 72 jälkeen:

```
73
74                  //Tietokoneen ohjaus
75
76                  'Liikkuminen vasemmalle ja oikealle
77                  etäisyys_vaaka = Abs(tietokoneX+PALKIN_LEVEYS/2 - palloX)
78                  etäisyys_pysty = Abs(tietokoneY – palloY)
79                  If etäisyys_vaaka > Rand(30,60) And etäisyys_pysty < 200 And
palloSuunta < 180 Then
80                      'Liikutamme tietokonetta kohti palloa
81                      If (tietokoneX < palloX) Then
82                          tietokoneX = tietokoneX + Min(etäisyys_vaaka,
PALKIN_NOPEUS)
83                      Else
84                          tietokoneX = tietokoneX - Min(etäisyys_vaaka,
PALKIN_NOPEUS)
85                      End If
86                  End If
```

Rivillä 77 laskemme siis tietokonepelaajan etäisyyden pallosta vaakasuunnassa, ja rivillä 78 etäisyyden pystysuunnassa. *Abs()*-funktio palauttaa aina parametrina saamansa luvun itseisarvon, eli voimme laskea etäisyydet yksinkertaisesti vähentämällä tietokoneen koordinaatit pallon koordinaateista ja varmistamalla *Abs()*-funktion käytöllä, että tulos on aina positiivinen.

Rivillä 79 testaamme, onko pallo vaakasuunnassa lähellä tietokonetta – käyttäen *Rand()*-funktiolla luotua häilyvää rajaa - , onko pallo pystysuunnassa lähellä tietokonetta – käyttäen rajana kahta sataa pikseliä – ja että onko pallo suuntansa perusteella menossa ylöspäin. Mikäli kaikki testit ovat tosia, liikutamme tietokonepelaajaa.

Tietokonepelaajan suunta valitaan rivillä 81, jossa tarkistamme, onko tietokoneen x-koordinaatti pienempi kuin pallon x-koordinaatti, vai ei. Eli toisin sanoen tarkistamme, onko pallo tietokoneen oikealla vai vasemmalla puolella. Mikäli tietokoneen x on pallon x:ää pienempi, kasvatamme tietokoneen x-koordinaattia, jolloin tietokone liikkuu oikealle. Muutoin vähennämme x-koordinaattia, jolloin tietokone liikkuu vasemmalle.

Lisäyksen tai vähennyksen suuruus (rivit 82 ja 84) riippuu pallon vaakasuuntaisesta etäisyydestä tietokoneen keskipisteeseen. Mikäli etäisyys on suurempi kuin *PALKIN_NOPEUS* (= tietokoneen suurin sallittu liikkumisnopeus), tulee liikenopeudeksi sama kuin *PALKIN_NOPEUS*. Jos taas vaakasuuntainen etäisyys on pienempi kuin *PALKIN_NOPEUS*, tulee liikenopeudeksi sama kuin etäisyys on. Näin tietokone liikkuu lähellä palloa hieman hitaammin, mutta ollessaan kaukana pallosta, se liikkuu nopeammin, mutta ei voi ylittää ihmispelaajan nopeutta.

Tietokone pyrkii vaakasuunnassa pallon kohdalle, mutta se liikkuu vain silloin, kun pallo on tulossa sitä kohti – eikä ole siis menossa kohti pelaajaa – ja kun pallo on jo pystysuunnassa melko lähellä sitä (alle 200:n pikselin etäisyydessä). Tietokone ei kuitenkaan liiku, mikäli sen keskipiste on vaakasuunnassa varsin lähellä palloa. Tällöin tietokone olettaa osuvansa palloon, mutta toisaalta myös odottaa, että pallo siirtyy vaakasuunnassa hieman etäämmäksi palkkinsa keskipisteestä, jolloin se taas reagoi hieman sattumanvaraisesti ja alkaa liikkua kohti palloa. Joskus tietokone tämän sattumanvaraisuuden vuoksi reagoi hieman liian myöhään, ja pallo karkaa siltä.

Teemme vielä pienen viilauksen tietokoneen liikkumiseen, joka estää sitä menemästä edes osittain ikkunan ulkopuolelle:

```
87
88                  'Tarkistetaan, ettei tietokone mene ulos ruudusta
89                  If tietokoneX < 0 Then tietokoneX = 0
90                  If tietokoneX > ScreenWidth() - PALKIN_LEVEYS Then tietokoneX
= ScreenWidth() - PALKIN_LEVEYS
91
```

Samanlaisen varmistuksen teimme jo aiemmin ihmispelaajalle, joten koodin toiminnan pitäisi olla selvillä. Ihmispelaajalle luomamme palkki

voidaan piirtää useaan kertaan, joten voimme hyvin käyttää sitä myös tietokonevastustajan piirtämiseen:

```
92
93                  //Piirretään tietokone
94                  DrawImage palkki, tietokoneX,tietokoneY
```

Tietokoneesta ei tosin paljoakaan ole vastusta, mikäli pallo ei voi osua siihen. Tähän seikkaan paneudumme seuraavaksi lisäämällä uuden törmäystarkistuksen heti pelaajan ja pelipallon törmäystarkistuksen perään, eli rivin 108 jälkeen:

```
109
110                 If DotInArea(palloX,palloY, tietokoneX,tietokoneY,
tietokoneX+PALKIN_LEVEYS-1,tietokoneY+PALKIN_PAKSUUS-1) Then
111                     'Pallo on törmännyt tietokoneeseen
112                     'Laitetaan pallo kimpoamaan palkista, kun palkin kulma on
180
113                     palloSuunta = BounceAngle(palloSuunta,180)
114                 End If
```

Tämäkin koodi on tuttua aiemmasta – ainoana erona on, että se on räätälöity tietokonepelaajalle ihmispelaajan sijaan. Toteutamme seuraavaksi pisteidenlaskun. Aina, kun pelipallo menee ikkunan yläreunaan, saa pelaaja yhden pisteen ja pallo pysähtyy. Kun pelipallo menee ikkunan alareunaan, saa tietokone yhden pisteen ja pallo pysähtyy. Tämän jälkeen pelaajan tulee painaa *Enter*-näppäintä, jolloin pallo siirretään uudestaan kentän keskelle ja sille arvotaan uusi suunta ja peli jatkuu. Kirjoitamme seuraavan koodin rivin 126 jälkeen:

```
2
3                   'Tarkistetaan maalit
4                   If palloY < 0 Then
5                       'Pelaaja on tehnyt maalin
6                       palloOlemassa = False
7                       pisteet_pelaaja + 1
8                   End If
9
10                  If palloY > ScreenHeight() Then
```

```
11                      'Tietokone on tehnyt maalin
12                      palloOlemassa = False
13                      pisteet_tietokone + 1
14                  End If
```

Tässä koodissa loimme pistemuuttujat pelaajalle ja tietokoneelle. Rivin 132 lauseke *pisteet_pelaaja + 1* siis kasvattaa muuttujaa yhdellä. Saman asian voisi ilmaista myös lausekkeella *pisteet_pelaaja = pisteet_pelaaja + 1*, mutta CoolBasic tarjoaa myös tällaisen oikotien kun muuttujaan lisätään yksittäinen luku. Sama pätee myös rivillä 138. Mikäli pallon y-koordinaatti menee alle nollan, tarkoittaa se, että pallo on ikkunan ylälaidassa ja pelaaja on tehnyt maalin. Mikäli pallon y-koordinaatti menee yli ikkunan korkeuden (= *ScreenHeight()*), tarkoittaa se, että pallo on ikkunan alalaidassa ja tietokone on tehnyt maalin. Kummassakin tapauksessa asetamme muuttujan *palloOlemassa* arvoksi *False*, jolloin peli jää odottamaan, että pelaaja painaa *Enter*-näppäintä.

Tämän luvun viimeisenä asiana teemme vielä pisteiden tulostuksen. Pisteet tulostetaan pelin aikana näytölle niin, että pelaajan saamat pisteet tulostuvat ikkunan vasempaan alalaitaan, ja tietokoneen pisteet ikkunan vasempaan ylälaitaan. Kirjoitamme seuraavan koodin pelin päälenkin loppuun, rivin 151 jälkeen:

```
27
28                  //Tulostetaan pisteet
29                  Color cbWhite
30                  Text 10,50, ”Tietokoneen pisteet: ” + pisteet_tietokone
31                  Text 10,ScreenHeight()-65, ”Pelaajan pisteet: ” +
pisteet_pelaaja
```

Rivillä 154 asetamme piirto- ja tekstintulostusväriksi valkoisen. *cbWhite* on CoolBasicin sisäänrakennettu vakio, joka siis tarkoittaa valkoista väriä. Kahdella alemmalla rivillä kirjoitamme pistetiedot eri koordinaatteihin. Liitteessä *LIITE* 2 on nyt koko tähänastisen ohjelmamme koodi.

5.4 Äänet ja musiikki

Tässä luvussa lisäämme peliimme äänet ja musiikin. Kaikki tarvitsemamme ääni- ja musiikkitiedostot löytyvät CoolBasicin juuren *Media*-kansiosta.

Tähänastinen peli tulisi siis olla tallennettuna CoolBasicin juurihakemistoon, jotta peliin tulevat tiedostoviittaukset toimisivat oikein. Mikäli olet tallentanut pelin lähdekoodin muualle kuin *CoolBasic*-kansioon, voit joko siirtää sen kyseiseen kansioon, tai voit kirjoittaa koodin alkuun seuraavan rivin:

```
ChDir ”X”
```

Jossa merkin *X* korvaat CoolBasicin juurihakemiston polulla, esimerkiksi *”C:\Program Files\CoolBasic\”*. Tätä riviä ei kuitenkaan tarvitse kirjoittaa, mikäli koodisi on tallennettu juurihakemistoon.
Tällä järjestelyllä takaamme sen, että peli tulee löytämään tarvitsemansa ääni- ja musiikkitiedostot.

Aloitamme laittamalla peliin taustamusiikin. Taustamusiikki alkaa aina soida uudestaan, kun kappale on päättynyt. Tämän vuoksi teemme pelin päälenkkiin tarkistuksen, jossa katsotaan, onko musiikki soimassa, vai ei. Mikäli ei ole, aloitamme musiikin soittamisen. Seuraava koodi tulee rivin 57 jälkeen:

```
58
59                  //Taustamusiikki
60                  If Not SoundPlaying(musiikki) Then musiikki = Play-
Sound(”Media\SK_Battle2.mp3”)
```

Käytämme taustamusiikkinamme *Media*-kansiosta löytyvää *SK_Battle2.mp3*-kappaletta. Koodimme luo jälleen uuden muuttujan, nimeltään *musiikki*. *Musiikki*-muuttujaan talletetaan äänikanava, jossa musiikkimme soi. Ehtolause tarkistaa, onko ääni soimassa kyseisessä

kanavassa, tai onko kanavaa edes olemassa. Mikäli vastaus on *False* (= ei), alamme soittaa musiikkia ja talletamme sen kanavan *musiikki*-muuttujaan. *Ääni* ja *muusiikki* tarkoittavat tässä siis samaa asiaa, sillä musiikkihan on vain tavallista suurempi ”ääni”.

Seuraavaksi on vuorossa ääni, joka kuuluu, kun pallo törmää pelaajan tai vihollisen palkkiin. Sama ääni kuuluu myös pallon törmätessä ikkunan vasempaan tai oikeaan laitaan. Seuraavan pienen koodirivin lisäämme kaikkiin törmäystunnistuksemme kohtiin, joissa pallo osuu joko pelaajan/vihollisen palkkiin tai ikkunan vasempaan/oikeaan reunaan. Koodi tulee siis neljään paikkaan, heti *BounceAngle()*-funktiota kutsuvien rivien jälkeen.

```
PlaySound ”Media\pop.wav”
```

Toisin kuin taustamusiikin soittamisessa, yksittäisiä ääniä soitettaessa meidän ei tarvitse tarkistaa, että edellinen ääni on loppunut. Näin on sen vuoksi, että tilanne, jossa ääni soitetaan, ei toistu lenkin jokaisella kierroksella, vaan pelkästään silloin, kun pallo törmää johonkin. Näin ollen ei ole vaaraa, että ääni tulisi soimaan kokoajan. Tämän vuoksi meidän ei tarvitse kirjoittaa näille äänille ylimääräisiä ehtolauseita tarkistamaan, voiko äänen soittaa.

Lisäämme vielä äänen tilanteeseen, jossa pelaaja tai tietokone tekee maalin. Käytämme *PlaySound*-komentoa samalla tavalla kuin ylempänäkin, mutta vaihdamme tiedostoksi *Media\blaster.wav*-äänen. Koodi lisätään maalientarkistuskohtaamme, eli kahteen lohkoon, joissa pelaajan/tietokoneen pisteitä kasvatetaan yhdellä.

Nyt pelimme on siinä määrin valmis, että emme tässä dokumentissa kehitä sitä enää enempää. Mikäli itse haluat, voit kokeilla tehdä kaikenlaisia muutoksia ja lisäyksiä peliin – kuten esimerkiksi vaihtaa

ääniä, vaihtaa pallon ja palkkien väriä sekä pelin taustaväriä, ikkunan kokoa jne.

Pelissä on myös jonkin verran virheitä ja ”bugeja”. Esimerkiksi pallo saattaa lähteä menemään keskellä kenttää tasan vaakasuoraa viivaa, jolloin se ei koskaan osu pelaajaan tai viholliseen. Samoin pallo saattaa mennä tasan pystysuoraa viivaa, mistä seuraa, että sekä pelaaja että vihollinen pysyvät paikoillaan, kunnes pelaaja kyllästyy ja päästää pallon läpi. Tietokoneen tekoälykin on paranneltavissa. Siitä on mahdollista tehdä sattumanvaraisempi ja sellainen, jonka toiminta ei ole yhtä helposti ennustettavissa kuin nykyään. Tietokone myös tärisee liikkuessaan.

6 CoolBasic-yhteisö

6.1 CoolBasic:n foorumit ja irc-kanava

Katsotaan CoolBasicin kotisivuilla sijaitsevia keskustelufoorumeita. (URL 6.1)

Foorumit ovat hyvä paikka löytää vastauksia kohtaamiinsa CoolBasiciin liittyviin ohjelmointi-ongelmiin. Siellä on useita aktiivisia jäseniä, jotka auttavat kykynsä mukaan. Tällä hetkellä kirjoitettuja viestejäkin löytyy foorumilta sen verran, että apu ongelmaan löytyy usein nopeasti. Foorumin hakutoimintoa kannattaa käyttää ennen uuden kysymyksen lähettämistä. Samoin kannattaa selata ihan CoolBasicin manuaaliakin – useiden komentojen kohdalla on kerrottu niihin liittyvistä erityishuomioista sekä yleisistä virheistä, joita niiden kanssa saatetaan tehdä. (URL 6.1, URL 6.2, URL 6.3, URL 6.4, URL 6.5)

Foorumeille on mahdollista kirjoittaa suoraan, ilman rekisteröintiä. Itse suosittelen kuitenkin rekisteröintiä jos huomaat, että tulet käyttämään foorumia myöhemminkin. Rekisteröinti on ilmaista ja siinä on muun muassa se etu, että sinut on helpompi muistaa ja tunnistaa, mikä antaa sinusta luotettavamman ja asiallisemman kuvan. Lisäksi voit tällöin

halutessasi kertoa profiilisivullasi itsestäsi. (URL 6.1, URL 6.2, URL 6.3, URL 6.4, URL 6.5)

Foorumeilla käydään keskustelua tietysti muustakin kuin ohjelmointiongelmista. Sinne voi tehdä ilmoituksen omasta peli- tai ohjelmaprojektistaan – tosin ilmoitus kannattaa tehdä vasta kun projekti on edennyt melko pitkälle ja siitä on jotakin näytettävää – esimerkiksi pari kuvaa tai kokeiluversio. Foorumilla voi myös hakeutua johonkin pelientekotiimiin ja näin yhdistää voimansa muiden kanssa. Oikestaan foorumilla voi rupatella lähes mistä tahansa – joskin muistaen asiallisuuden ja tarkistaen, että lähettää viestinsä oikealle alueelle ja sopivalla otsikolla! (URL 6.1, URL 6.2, URL 6.3, URL 6.4, URL 6.5)

Keskustelualueet on jaettu eri aiheiden lisäksi myös jopa kahdelle kielelle: suomi ja englanti. Joskin englannin alue on hiljaisempi, koska CoolBasic ei ole vielä levinnyt kunnolla ulkomaille, johtuen manuaalista, joka on toistaiseksi olemassa vain suomeksi.
Foorumien käytännöt tulevat hyvin tutuiksi lukemalla ensitöikseen foorumin säännöt. (URL 6.1, URL 6.2, URL 6.3, URL 6.4, URL 6.5)

CoolBasicilla on myös oma Irc-kanavansa: #coolbasic IRCnetissä. Kanavalla on mahdollista rupatella muiden CoolBasicin käyttäjien kanssa ja sielläkin voi tarvittaessa myös kysyä neuvoa.
(URL 6.6)

6.2 CoolBasic Koodikirjasto (CBKK)

CoolBasic Koodikirjasto on sivusto, jolle CoolBasicin käyttäjät lisäävät kirjoittamiaan, yleiseen käyttöön soveltuvia koodeja muiden avuksi ja iloksi. (URL 6.7)

Koodia on toki mahdollista lähettää myös CoolBasicin foorumeille, mutta CBKK:ssa koodit ovat selkeämmin ja helpommin löydettävissä, kun ne eivät huku muiden viestien joukkoon. Koodeja voi käyttää yleensä täysin vapaasti, vaikkapa ottamalla oppia siitä, miten joku toinen on ratkaissut tietyn ongelman tai kopioimalla suoraan jonkin sivustolta löytyvän koodin ohjelmaansa. Kopioitaessa on hyvän tavan mukaista merkitä omassa ohjelmassaan kopioidun koodin kohdalle kommenttirivillä koodin alkuperäinen tekijä, kopiointipäivämäärä sekä tarkka sivu, jolta koodi on kopioitu.

Kenen tahansa on mahdollista lähettää tekemiään koodeja sivustolle. Se tosin vaatii maksuttoman rekisteröitymisen, mutta siitäkin suoriutuu muutamassa minuutissa. Rekisteröitymisen jälkeen on myös mahdollista kommentoida sivustolla olevia koodeja ja antaa niille arvosanoja.

CoolBasic Koodikirjasto pitää sisällään myös *pastebin*-toiminnon, johon on mahdollista liittää koodia, joka säilyy tietokannassa pari viikkoa. Varsinaiselle sivustolle lisätyt koodit säilyvät tietysti niin pitkään kuin sivustokin on olemassa. (URL 6.8)

Lopuksi on vielä mainittava CBKK:ssa oleva CB-sarjakuva. Tämä muutaman jakson pituinen sarjakuva sisältää humoristista (ja fiktionaalista) tarinaa CoolBasicin foorumien tunnetuista jäsenistä – useimmiten valvojista. (URL 6.9)

6.3 CBCorner

Tämä *"CoolBasic-harrastajan ykkösmediaksi"* luonnehdittu sivusto sisältää CoolBasicilla tehtyjen pelien arvosteluja, pelintekokilpailuja, cb-yhteisössä tunnettujen henkilöiden haastatteluja, CoolBasiciin liittyviä ohjelmia sekä CB-ohjelmoinnin eri aihealueisiin liittyviä oppaita. (URL 6.10)

Haastatteluosiossa on luettavissa muun muassa CoolBasicin kehittäjän ja ohjelmoijan, Jukka Lavosen haastattelu. Haastattelussa tiedustellaan, mistä idea omaan ohjelmointikieleen on lähtenyt ja millaista sen kehittäminen on ollut. (URL 6.11, URL 6.12, URL 6.13)

Peliosiosta löytyy CoolBasicilla tehtyjä, ilmaisia pelejä aina Arcade-peleistä urheilu- ja autopelien kautta taitopeleihin. Useista peleistä on myös oma arvostelunsakin. (URL 6.11, URL 6.12, URL 6.13)

Kilpailuja sivustolla on ollut kirjoitushetkellä kolme: Kesäskaba 2006, Syyskisa 2005 ja Pelintekokisa 2005. (URL 6.11, URL 6.12, URL 6.13)

7 CoolBasicin kehittyneempi käyttäminen

Tähän lukuun kannattaa siityä vasta sitten, kun olet hetken kokeillut CoolBasicia ja sen eri komentoja. Olisi hyvä, jos sinulla olisi edes pieni, yksinkertainen peli valmiina ennen kappaleessa olevien uusien asioiden opettelua.

Kappaleessa käymme läpi eri työkaluja ja tekniikoita, joiden avulla CoolBasicin käyttäminen helpottuu ja monipuolistuu.

7.1 CoolBasic Software Development Kit

CoolBasic Software Development Kit (CBSDK) on vähän saman tapainen järjestelmä kuin CoolBasic Koodikirjasto – joskin astetta pidemmälle vietynä. Kyseessä on CoolBasicin rinnalle asennettava paketti, joka tuo monien henkilöiden tekemiä CB-funktioita suoraan CoolBasicin editoriin käytettäviksi. (URL 7.1)

CBSDK:n käyttäminen vaatii, että Windowsiin on asennettu *.NET Framework* (vähintään versio 1.0). Ohjelmisto lisää CoolBasicin manuaaliin muiden osioiden lisäksi oman osionsa, joka sisältää ohjeet CBSDK:n sisältämien funktioiden käyttämiseen. Lisäksi se lisää *Tools*-valikkoon *CBSDK Control Center* –ohjelman, jonka avulla on mahdollista

hakea päivityksiä SDK:hon – eli käytännössä lähinnä uusia funktioita, jos sellaisia on julkaistu. Päivittäminen on helppoa, eikä vaadi uusien pakettien lataamista selaimella tai uusien asennusohjelmien käynnistämistä. Samoin *Control Center* tarvittaessa avaa CoolBasiciin eri funktioihin liittyviä esimerkkikoodeja. (URL 7.1)

Tärkeä asia on se, että SDK:n tarjoamat funktiot tulee aina sisällyttää jokaiseen ohjelmaan, jossa niitä käytetään. Sisällyttäminen tapahtuu CoolBasicin *Include*-komennolla, jolle annetaan parametriksi tiedosto, joka sisältää kyseiset funktiot. Funktiot on SDK:ssa jaettu eri tiedostoihin sen mukaan, mihin ryhmiin ne on jaoteltu SDK:n manuaalissa.

Control Center tarjoaakin kätevän toiminnon, jolla tavittavan include-koodin voi helposti kopioida leikepöydälle ja sitä kautta koodiin. Kun include-koodi on liitetty oikein, on mahdollista käyttää kyseiseisiä funktioita. Erittäin helppoa kun sen kerran oppii.

CoolBasic Software Developmet Kit:iin on mahdollista saada mukaan myös itse tekemiään funktioita, mutta tämä ei ole aivan niin yksinkertaista kuin CoolBasic Koodikirjastossa. Oma koodi tulee lähettää jollekulle CBSDK:n ylläpitäjälle – mieluiten CBSDK:n omalla foorumilla. Tämän jälkeen ylläpitäjä päättää, onko koodi tarpeeksi laadukas SDK:hon. Jos on, saat todennäköisesti CBSDK:n foorumilla erityisoikeudet käyttää *Help Builder*-toimintoa, jolla pääset luomaan omat, yksityiskohtaiset ohjeet tekemillesi funktioille. Tämän jälkeen ylläpitäjä julkaisee funktiosi ohjeineen päivineen CBSDK:hon, ja muut käyttäjät voivat ladata ne päivitystoimintoa käyttäen.

CBSDK on monelle koodaajalle varsin tarpeellinen asentaa jossakin vaiheessa, sillä se tarjoaa muun muassa Windows-tyylisiä lomake-elementtejä, graafisia erikoistehosteita, kehittyneempää matematiikkaa ja jopa alkeellista 3D:n piirtämistä sekä paljon muuta. (URL 7.1)

7.2 Internet-toimintoja CoolBasiciin

CoolBasicissa on yksinkertainen DLL-tuki, jonka avulla sen toimintoja voidaan laajentaa. Osa CBSDK:n funktioista käyttävät nimen omaan DLL:ä. Tätä *cbNetwork*-kirjastoa (internet-kirjasto) ei ole ainakaan vielä lisätty CBSDK:iin, joten mainitsen sen käytöstä tässä erikseen. (URL 3.2)

cbNetwork-kirjastoa käyttämällä on mahdollista luoda nettipelejä CoolBasicilla. Kirjasto mahdollistaa pelipalvelimen luomisen ja varsinaisten pelaajien kommunikoinnin pelipalvelimen kanssa. Lisäksi kirjasto antaa ladata tiedostoja internetistä *HTTP*-protokollaa käyttäen. Tämä mahdollistaa muun muassa pelin kommunikoinnin *PHP*-skriptin kanssa esimerkiksi päivitystietoja hakiessaan. (URL 7.2)

7.3 Omien tilesettien luominen

Aloitettaessa tilesetin tekemistä, on päätettävä, minkä kokoisia tilejä peli tulee käyttämään. 32x32 pikselin kokoiset tilet ovat melko pieniä, jolloin ne tulevat helposti kyseeseen kun kentistä halutaan tehdä erittäin yksityiskohtaisia. 64x64-kokoiset tilet ovat kenties käytännöllisempiä pelissä, jossa esimerkiksi esteitä ja ahtaita paikkoja tarvitaan vähemmän. Toki pieniä tilejä yhdistelemällä voidaan saada aikaan isompi kokonaisuus – esimerkiksi pöytä. Loppujen lopuksi kyse on makuasiasta.

Kun olet päättänyt tilejen koon, voit tehdä pikaisen arvion, miten monta erilaista tileä kenttään tulee. Otetaan esimerkiksi 20 tileä. Jos yhden tilen koko on 32x32 pikseliä, voidaan tilesetin kooksi näin ollen asettaa 640x640 pikseliä. Pääasia että kaikki tilet mahtuvat tilesettiin – ja että tyhjää ei jää kovin paljoa. Voit luoda tämän kokoisen tyhjän kuvatiedoston ja alkaa piirtää siihen tilejä. Vaihtoehtoisesti voit myös piirtää tilet erikseen ja liittää varsinaiseen tilesettiin sitten kun ne ovat valmiita. Tilet tulisi siis asettaa vierekkäin.

Tilesetit toimivat niin, että jokaisella tilellä on oma järjestysnumeronsa. Kun tile luetaan tilesetistä, menee numerointi vasemmalta oikealle ja ylhäältä alas. Eli tilet on mahdollista jakaa useille riveille. Kannattaa avata CoolBasicin Media-kansion *tileset.bmp*-tiedosto ja ottaa siitä mallia.

Tärkeää on aina varmistaa, että tilesetin leveys ja korkeus ovat oikein. Tileset tulee olla jaettavissa yhden tilen koolla siten, ettei puolikkaita/keskeneräisiä tilejä jää. Tämä tarkoittaa sitä, että jos tileset on pikselinkin liian leveä, kapea, korkea tai matala, ei kenttää voida ladata oikein. Pidä siis mittasuhteita tarkasti silmällä! (URL 4.2, URL 4.4)

7.4 Vaihtoehtoiset ohjelmat Tilesterille

Tilesterin on kehittänyt sama henkilö, joka on tehnyt suurimman työn CoolBasicin kehityksen parissa. Tässä kohtaa Tilesterin kehitys on kuitenkin jäänyt vähäiseksi, eikä se ole kehittynyt aivan niin hyväksi, kuin mahdollista olisi.

Tässä kappaleessa kerron kahdesta vaihtoehtoisesta tilekarttaeditorista, jotka sisältävät Tilesteriä kehittyneemmän käyttöliittymän ja monipuolisempia ominaisuuksia. Molemmat ohjelmat ovat ilmaisia.

7.4.1 A Stigmator

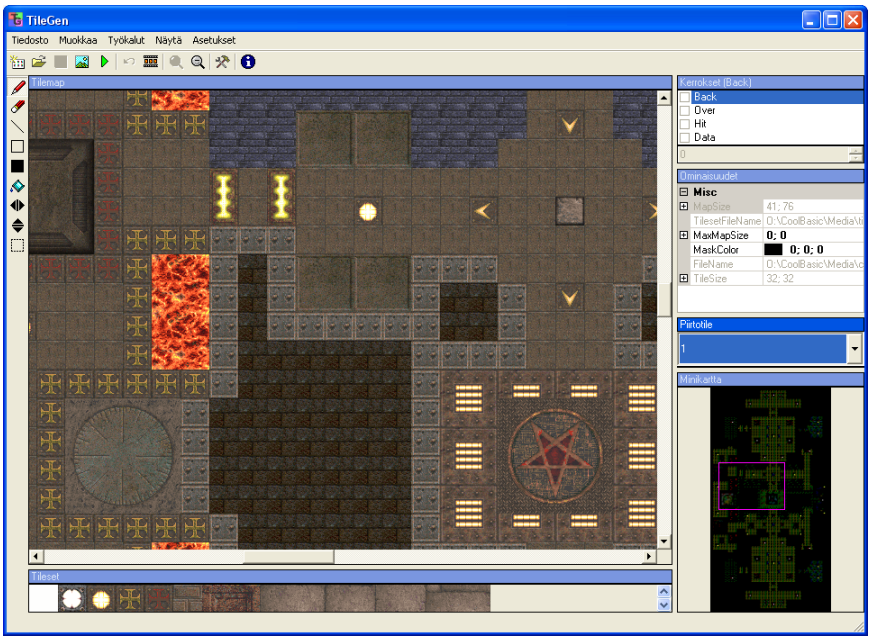
A Stigmator on CoolBasicilla tehty tilekarttaeditori. Se tarjoaa helppokäyttöisen käyttöliittymän kenttien tekemiseen. Kaikki toiminnot ovat käytettävissä käyttöliittymän nappuloista sekä pikanäppäimistä. Erikoisia toimintoja A Stigmatorissa ovat *Tee minikartta* sekä kolmitasoinen suurennus. Ensimmäinen toiminnoista luo koko kentästä kätevän bittikarttakuvan, jonka voi esimerkiksi liittää peliinsä tai lähettää internetiin näytiksi pelinsä tajonnasta. Suurennustoiminnolla taas voi tarkastella kenttäänsä normaalia lähempää ja yksityiskohtaisemmin, tai katsella kenttää pienenä, jolloin siitä

hahmottaa kerralla suuremman osan. Kuvassa A Stigmatoriin on ladattu CoolBasicin Media-kansiosta kenttä nimeltä *cdm2.til*, ja tilesettinä on käytetty samasta hakemistosta löytyvää *tileset.bmp*-tiedostoa. Kuvassa kenttä on loitonnettu.

A Stigmatorin koko on pakattuna noin 1,5 MB ja purettuna noin 3,30 MB (versio 1.33, joka on julkaistu 7.9.2006). Koska A Stigmator on tehty CoolBasicilla, ei se ole aina kovin nopea, mutta sen tehot riittävät kuitenkin varsin jouhevaan työskentelyyn. Ohjelman on todettu joillakin tietokoneilla kaatuvan välillä *"Memory Access Violation"* –virheilmoitukseen, mutta useimmilla tietokoneilla tätä ei ole havaittu. (URL 7.3, URL 7.4)

7.4.2 TileGen

TileGen on tilekarttaeditori, joka luo kentän dynaamisesti. Kentälle ei siis tarvitse määrittää leveyttä ja korkeutta, ja sitä voi aina laajentaa mihin tahansa suuntaan. A Stigmatorin tapaan myös TileGenissä on mahdollisuus loitontaa näkymää niin, että kenttä näkyy suuremmalta alalta. Loitonnus tosin tummentaa näkymää (joskaan ei vaikuta itse kenttään). TileGenissä on yhdeksän muistipaikkaa, joihin voi tallettaa vähän väliä tarvitsemiansa palasia tilesetistä. Näin usein tarvitsemiaan paloja ei tarvitse aina etsiä tilesetistä, joka saattaa pelistä ja kentästä riippuen olla suuri.



TileGen on kirjoitettu C#:lla, joten tietokoneessa tulee olla asennettuna *.NET FrameWork*. Ohjelman koko pakattuna on noin 900 KB (versio 0.3 Beta, julkaistu 1.12.2005). Purettuna ohjelman koko on noin 1,4 MB. (URL 7.5, URL 7.6)

8 CoolBasicin historia

8.1 Idea uuteen BASIC-kieleen

CBCorner.net-sivuston ylläpitäjät haastattelivat 1.5.2005 CoolBasicin kehittäjää, Jukka Lavosta (aka Zero). Jukka Lavonen on CoolBasicin pääohjelmoija – ja itseasiassa tämän dokumentin kirjoitushetkeen menessä ainoa ohjelmoija. Haastattelussa hän kertoo aloittaneensa aluksi kehittämään peliä nimeltä *GigaBot*, jossa robotit ottavat toisistaan mittaa ja pelaajan tehtävänä on ohjelmoida omalle robotilleen mahdollisimman hyvä tekoäly. Parin kuukauden kehityksen jälkeen kasassa oli jo toimiva malli, ja hän keksi että voisi saman tien tehdä kokonaisen ohjelmointikielen!

Uudesta ohjelmointikieliprojektistaan Zero teki keskusteluaiheen Pelisivut.org-sivustolla. Ketjusta tuli erittäin suosittu, mutta nykyään sitä ei enää ole olemassa, sillä Pelisivut.org:n keskustelupalsta uusittiin elokuussa 2006. Samaisessa keskustelussa uuden kielen nimeksi

päätettiin antaa *CoolBasic* – alkuperäinen työnimi kun oli *WinBASIC*. (URL 8.1, URL 8.2, URL 8.3)

8.2 Ensimmäiset BETA-versiot

CoolBasicin ensimmäinen betaversio julkaistiin vuonna 2003. Kyseisellä versiolla ei voinut ajaa ohjelmia Windows 98:lla, mutta tämä ongelma korjattiin Beta 2-versiossa. Ensimmäisen Beta-version jälkeen kielen kääntäjä on kirjoitettu useita kertoja kokonaan uudestaan. Jokaisella kerralla kääntäjä on nopeutunut ja sen luotettavuus ja sisäinen arkkitehtuuri ovat parantuneet. (URL 8.1, URL 8.3)

Ensimmäisiä betaversioita 1 – 3 on kutsuttu ainakin kahdella nimellä. *BETA 1*, *BETA 2* ja *BETA 3* ovat tarkoittaneet samaa kuin *BETA 1.0 (release 1)*, *BETA 1.0 (release 2)* jne. (URL 8.3, URL 8.4)

Beta 1 -versiossa lähdekoodin kääntäminen ei onnistunut Windows 98 -käyttöjärjestelmällä. Tämä korjattiin Beta 2 –versiossa, jolla oli kuitenkin ongelmia asennusohjelmansa kanssa. Beta 2 ei asentunut joillekin tietokoneille, sillä asennuksesta puuttui tiettyjä ajonaikaisia tiedostoja, jotka olivat olemassa joissakin tietokoneissa valmiiksi, ja joissakin eivät. Tämä korjattiin Beta 3:ssa. (URL 8.4)

Beta 3 julkaistiin 6.1.2004. Se kärsi kuitenkin muistivuoto-ongelmista. Aina kun jotakin käyttäjän kirjoittamaa funktiota kutsuttiin, luotiin funktion muuttujat uudestaan, mutta niitä ei koskaan poistettu funktion päättymisen jälkeen, mikä täytti pikkuhiljaa järjestelmän muistin. (URL 8.4, URL 8.5)

Beta 4 julkaistiin 5.4.2004. Joidenkin pikkubugien lisäksi Beta 4:llä oli ongelmia midi- ja it-musiikin kanssa, eikä kyseisiä musiikkiformaatteja pystynyt CoolBasicilla toistamaan. (URL 3.1)

Beta 5 julkaistiin 21.5.2004. Pari muutosta sisältänyt päivitys korjasi midi-musiikkiongelman, mutta it-musiikkiformaatti ei edelleenkään toiminut johtuen FMOD-äänikirjaston virheellisestä toiminnasta. Toinen muutos mahdollisti ikkunan koon määrittämisen minkä muotoiseksi tahansa. (URL 3.1)

8.3 10.x-Betaversiot

Monet CoolBasicin sisäänrakennetut komennot ja funktiot nimettiin uudestaan siirryttäessä betaversioon 10 vuonna 2004. Muutoksilla haluttiin selkeyttää kieltä ja poistaa samankaltaisuuksia *BlitzBasic*-ohjelmointikielen kanssa, jossa oli saman nimisiä komentoja ja funktioita.

Komentoja nimettiin uudelleen, joidenkin toimintaa muutettiin, ja joitakin yhdistettiin. Esimerkiksi komennot *EnableEscapeKey* (joka komentaa peliä sulkeutumeen heti, mikäli käyttäjä painaa *Esc*-näppäintä) ja *DisableEscapeKey* (joka estää pelin automaattisen sulkeutumisen käyttäjän painaessa *Esc*-näppäintä) yhdistettiin ja nimettiin uudelleen *SafeExit*-komennoksi, jota käytetään joko *SafeExit ON* (= *EnableEscapeKey*) tai *SafeExit OFF* (= *DisableEscapeKey*). Näiden muutosten johdosta CoolBasicin käyttäjien tuli opetella kielen komentoja uudelleen ja kääntää menossa olevia projektejaan uusien komentojen mukaisiksi. (URL 8.3, URL 8.6)

20.12.2004 julkaistu betaversio 10.0 toi mukanaan myös paljon muita uudistuksia. Esimerkiksi uusittu objektijärjestelmä mahdollistaa pelihahmojen ja muiden objektien luomisen, animoimisen, liikuttamisen sekä törmäämisen toisiin objekteihin ja kenttään.

Täysin uusi tyyppirakenne taas mahdollistaa omien tietoluokkien luomisen, joihin voi luoda jäseniä, jotka voivat sisältää useita eri tyyppisiä muuttujia. Samalla kieli sai kaksi lisäystä tietotyyppeihinsä:

Short ja *Byte*, jotka ovat *Integeriä* pienempiä kokonaislukutyyppejä. Versio sisälsi myös kokonaan uuden koodieditorin. (URL 3.1)

CoolBasicin koodieditori kirjoitettiin jälleen uusiksi betaversiossa 10.4, joka julkaistiin 3.11.2005. Editorin uusiin ominaisuuksiin lukeutuvat muun muassa koodissa komentojen tuplaklikkaaminen, mikä avaa kyseisen komennon ohjeistuksen manuaalissa; tiedostolistat kokonaisten projektin hallitsemista varten; sekä parennettu funktio-listaaja, joka löytää nyt myös tyypit, luokat, globaalit muuttujat, vakiot ja tavalliset muuttujat.

Kielen kääntäjä muutettiin ulkoiseksi. Aiemmin kääntäjä oli editoriohjelman sisällä, mutta nyt se on ohjelmistossa erillisenä ohjelmana (*CB.exe*), jota voi myös itse kutsua komentoriviltä. Käytännössä tämä mahdollistaa kokonaan toisen koodieditorin käyttämisen CoolBasic-ohjelmoinnissa, jos joku niin haluaa tehdä. Komentojen osalta beta 10.4 kohensi tilannetta jälleen parantelemalla muutamia syötekomentoja sekä lisäämällä *GhostObject*- ja *DrawGhostImage*-komennot, jotka mahdollistavat ohjelmistopohjaisen antialiasoinnin objekteille ja kuville. Aiemmin tämä ei CoolBasic-peleissä ja ohjelmisssa ollut ollenkaan mahdollista tehdä reaaliajassa.

Toistaiseksi viimeisin CoolBasicin päivitys on 10.43. (CoolBasicin manuaalin *Mitä uutta?*-osiossa tämän päivityksen versioksi tosin sanotaan 10.5, mikä tarkoittaa samaa kuin 10.43). Päivitys muun muassa parantelee koodieditoria lisäämällä mahdollisuuden raahata editoriin tiedostoja hiirellä, tulostaa lähdekoodin sekä muita pieniä korjauksia editorissa.

Editoriin lisättiin myös *Recover Utility* -sovellus, jolla on mahdollista palauttaa varmuuskopioita ajetuista lähdekoodeista. Myös paria komentoa on paranneltu hieman. (URL 3.1)

9 CoolBasic tulevaisuudessa

9.1 CoolBasic Advance ja CoolBasic 3D

Nykyinen CoolBasic ei ole oliopohjainen. Zero on halukas tekemään CoolBasiciin jälleen kerran suuren syntaksimuutoksen ja muuttamaan kielen oliopohjaiseksi. Näin ollen esimerkiksi käsky *MoveObject car, 10* muuttuisi muotoon c*ar.Move 10*. *Car*-objektista tulisi siis olio, joka voisi kuulua vaikka luokkaan nimeltä *vehicles*. Oliopohjaisessa kielessä on paljon muutakin erilaista kuin pelkkä syntaksi, mutta tässä dokumentissa emme käy sitä läpi. Kiinnostuneiden kannattaa lukea lisää olio-ohjelmoinnista vaikkapa kirjastossa. (URL 9.1)

Uuden, oliopohjaisen kielen nimeksi on tarkoitus tulla *CoolBasic Advance (CBA)*. Syntaksimuutoksen lisäksi kielen moottoriin on tarkoitus tehdä muitakin radikaaleja muutoksia. *BlitzBasic*, jolla CoolBasic on tällä hetkellä hyvin suurelta osin rakennettu, jää kokonaan pois uudessa *Advance*-kielessä. BlitzBasicin on havaittu hidastavan nykyistä CB:tä erittäin paljon, ja Zero aikoo tämän vuoksi vaihtaa ohjelmointialustaa johonkin paljon tehokkaampaan kieleen, kuten esimerkiksi C++:aan. Tämä tulee näkymään CBA:n suurena nopeutumisena.

CoolBasic Advancen valmistuessa vanhan kielen ylläpitäminen loppuu. Vanhasta kielestä aletaan käyttää nimitystä *CoolBasic Classic (CBC)*, ja nimitys on jo nykyään osittain käytössä, kun puhutaan CoolBasicin kehitysvisioista. (URL 9.1, URL 9.3)

Siinä missä CoolBasic Advance tulee näillä näkymin olemaan ilmainen, on sille tarkoituksena kehittää myös maksullinen lisäosa: *CoolBasic 3D (CB3D)*. Tämä lisäominaisuus tarjoaisi komennot kolmiulotteisen grafiikan toteuttamiseksi CoolBasic Advance:lla. 3D-kiihdytyksen on tarkoitus pohjautua DirectX:ään. Zero on alustavasti ilmoittanut CB3D:n hinnaksi 20€. Kuitenkaan CoolBasic Advancesta ei ole vielä ilmestynyt

mitään kokeiluversiota, joten jää nähtäväksi, valmistuuko kyseinen kieli koskaan. (URL 9.2, URL 3.1)

9.2 Uusi BASIC-ohjelmointikieli

Koska CoolBasicin kehityksessä on ollut niin suuri tauko, päätti eräs CoolBasic-käyttäjä, *Dibalo*, alkaa vuosien 2006 – 2007 vaihteessa kehittää kokonaan uutta basic-kieltä. Kyseessä on oliopohjainen ohjelmointikieli, jonka uusin versio tätä kirjoitettaessa on 0.0.2.4, eli kieli ei ole vielä lähelläkään valmista. Kyseisen kielen nimi on ChaosBasic (ChB).

Teknisiä eroja CoolBasicin ja ChaosBasicin välillä ovat muun muassa se, että ChB ei pohjaudu BlitzBasiciin, vaan C++:aan. ChB käyttää myös DirectX:n sijasta OpenGL-grafiikkarajapintaa. OpenGL on yksi seikka, joka tuo ChB:n lähemmäs tilannetta, jossa se voitaisiin kääntää myös Linuxille. (URL 9.4, URL 9.5)

ChaosBasicille ei vielä ole omaa koodieditoria, mutta sellainen on toki tulossa myöhemmin. Olen itse tehnyt *Notepad*++ -tekstieditorille asetustiedoston, joka määrittää ohjelmaan syntaksivärjäyksen ChaosBasic-kielelle. Asetustiedostoni tosin on ChB:n vanhemmalle 0.0.1.8-versiolle, joten se ei täysin vastaa nykyisen version komentolistausta. Aion myöhemmin vielä päivittää tiedostoa. Tällä hetkellä se on saatavilla osoitteessa URL 9.6
Itse Notepad++-ohjelmisto löytyy osoitteesta URL 9.7
Tätä syntaksivärjäystä voi käyttää ainakin siihen saakka, että ChaosBasicin oma koodieditori valmistuu. Lisäksi syntaksivärjäystä voi itsekin päivittää ja muokata halunsa mukaan.
(URL 9.6, URL 9.7)

9.3 CoolBasicin kehitys loppunut kokonaan?

Tässä luvussa kerron omia ajatuksiani ja pohdin CoolBasicin tulevaisuuden kehitystä. Näin ollen tämän luvun sisältö ei pohjaudu lähdemateriaalista tarkistettavissa olevaan tietoon, joten lukijan tulee tiedostaa, että tämän luvun asiat ovat minun omia mielipiteitäni ja käsityksiäni.

Tällä hetkellä näyttää siltä, että CoolBasicin kehitys on loppunut, tai ainakin se on suurten muutosten ja kenties ongelmienkin edessä. Zero ei ole kiinnostunut jatkamaan enää CoolBasic Classicia, mikä tarkoittaa sitä, ettei nykyiseen CoolBasiciin ole tiedossa esimerkiksi bugikorjauksia. Mikäli CoolBasic Advance joskus valmistuu, on meillä todennäköisesti edelleen käytettävissä myös CoolBasic Classic, joka sopii hyvin sellaisille CoolBasic-ohjelmoijille, jotka eivät ole halukkaita vielä siirtymään olio-ohjelmointiin. Silti CBC säilyttää nykyiset buginsa ja rajoituksensa, mikä todennäköisesti ajaa käyttäjiä CoolBasic Advancen puoleen.

CoolBasic Advance saattaa olla kuitenkin ongelmien edessä. Kehittäjä Zeron kaavailemat 3D-lisäominaisuudet CBA:han ovat erittäin työläitä toteuttaa. Joko CoolBasiciin on ohjelmoitava kokonaan oma 3D-moottorinsa – johon kuuluisi siis 3D-objektijärjeslmä, kenttäjärjestelmä, törmäyksentunnistusjärjestelmä ja mahdollisesti myös fysiikanmallinnusjärjestelmä – tai sitten kieleen on liitettävä jokin valmis grafiikkamoottori, joka sisältää lähes kaikki edellä mainitut ominaisuudet. Siltikin valmiin moottorin kiinnittäminen ohjelmointikieleen voi olla erittäin suuren työn takana, sillä moottorin komentoja on voitava käyttää mahdollisimman suoraan itse CoolBasic Advance:lla kirjoitettavassa ohjelmakoodissa. Kuitenkaan ei tule unohtaa helppokäyttöisyyttä, eli jotkin osat ko. moottorissa olisi CBA:n syytä hoitaa automaattisesti ohjelmoijan puolesta – kuten esimerkiksi moottorin alustus tiettyyn grafiikkatilaan. Zero ei ymmärtääkseni ole

halukas kehittämään CoolBasic Advancea, mikäli ei saa siihen tehtyä myös 3D-lisäominaisuuksia.

Toinen ongelma liittyy Microsoftin Windows Vista –käyttöjärjestelmään. Vista kun tukee DirecX 9:ää ja 10:tä. CoolBasic Classic käyttää DirecX 7.0:aa, minkä toimivuudesta Vistassa ei ole varmuutta – ainakaan tulevaisuudessa. Nykyisiä CoolBasic-pelejä on saatu toimimaan myös Windows Vistassa, mutta Zeron mukaan Vista ei varsinaisesti tue enää DirectX:n 9.0:aa vanhempia versioita. Mikäli Vistan DirectX 7.0 –tuki joskus loppuu, on CoolBasic ongelmissa. Ratkaisu olisi tietysti tehdä CoolBasic Advance käyttämään DirectX 9:ää, mutta Zeron mukaan DX9 on paljon laajempi ja monimutkaisempi grafiikkakirjasto, kuin DX7. Mikäli oikein muistan, ei DirectX9:stä löydy enää 2D-grafiikkatilaa, mikä pakottaisi siis CBA:n käyttämään 3D-kiihdytystä myös 2D-peleissä. Tämä ei sinänsä olisi haitallista itse CBA:n tuotospeleille, sillä 3D-kiihdytys kyllä antaisi nopeutta tiettyihin ominaisuuksiin myös 2D-peleissä – kuten esimerkiksi antialiasointiin. Kuitenkin valikoiden, tekstien ja kuvien piirtäminen näytölle pitäisi tehdä 3D-tilassa, mikä on työläämpää kuin 2D-tilassa. CBA-ohjelmoijalle tilanne ei olisi hankala, sillä CBA sisältäisi edelleen yksinkertaiset komennot näihin tarkoituksiin. Mutta 2D- ja 3D-välisten tilojen muutostyön hoitaisi CBA, mikä tulee Zerolle työlääksi ohjelmoida, kun CBA ei voi enää suoraan käyttää 2D-grafiikkatilaa. Lisäksi DirectX 9 on muutenkin monimutkaisempi kirjasto käyttää kuin DirectX 7.

Mielestäni ongelmat ovat kuitenkin ratkaistavissa. Jos ei nyt, niin ainakin joidenkin vuosien sisällä, kun Zero saa tarpeeksi aikaa tutustua esimerkiksi DirectX 9:ään ja Windows Vistaan. Windows Vista ei kuitenkaan vielä ole suuri ongelma CoolBasicin käytössä, sillä tällä hetkellä suurin osa Windows-käyttäjistä käyttää edelleen Windows Xp:tä, jossa CB-pelit toimivat. On kuitenkin varsin selvää, että tulevaisuudessa valtaosa käyttäjistä siirtyy joko Windows Vistaan, tai sitten pois Windowsista, esimerkiksi Linuxin tai Macintoshin pariin.

Vistaan siirtyminen edellyttää kuitenkin Vistan kypsymistä entisestään ja siinä havaittujen ongelmien korjaamista, jonka jälkeen enemmän Windows Xp-käyttäjiä uskaltaa siirtyä siihen. Itselläni ei ole Vistasta kokemusta, joten en tiedä tarkkaan sen ongelmista, mutta kuulemma siinä sellaisia on. Siksi itsekin pitäydyn vielä Windows Xp:ssä.

Jos mietin vastausta kysymykseen, *”Kehittyykö CoolBasic enää tästä eteenpäin?”*, päädyn tulokseen, että todennäköisesti kehittyy. Eri asia on, *milloin*. Zero on nähnyt todella suuren vaivan jo nykyisen CoolBasic Classicin eteen. Jos ei hänellä olisi pienintäkään kiinnostusta jatkaa projektiaan enää tästä eteenpäin, niin tuskin olisi CoolBasic edennyt edes tähän nykyiseen tilaansa. Elämäntilanne kuitenkin saattaa muuttua, ja joskus ei vain ole yksinkertaisesti aikaa tehdä. Syynä voi olla opiskelut, työtilanne, tai mikä vain; jopa useankin vuoden ajan. Töiden lomassa oleva vapaa-aikakin voi olla ymmärrettävästi helpompaa viettää jollain muulla tavalla, kuin työlästä ja haastavaa ohjelmointiprojektia tehden.

Mutta joka tapauksessa ajatus CoolBasicin jatkamisesta säilyy Zeron mielessä pitkään, ja luultavasti hänellä jossain vaiheessa taas riittää aikaa ja tarmoa alkaa toteuttamaan uutta pelientekoalan mestariteosta. Jos aikaa kuluu monia vuosia ilman kehitystä, ei Zeron kehittämä uusi ohjelmointikieli ehkä kanna enää niimeä CoolBasic, mutta varmasti se tulee säilyttämään monia nykyisen CoolBasicin perinteitä, kuten aloittelijaystävällisyys, laadukas manuaali sekä sisäänrakennetut moottorit (itse tehdyt tai ulkopuoliset kirjastot), kuten objektijärjestelmä törmäystarkistuksineen, grafiikkamoottori (mukaan lukien esimerkiksi kuvien todella vaivaton piirtämistapa) sekä äänikirjasto. Jos nimi vaihtuu, voidaan sanoa, että CoolBasicin kehitys loppuu; mutta toisaalta jos kehitys lähtee liikkeelle aivan lähivuosina, ei nykyistä nimeä kannata heittää hukkaan – se kun on kerännyt ympärilleen jo varsin laajan käyttäjäkunnan.

Zero ei ole kirjoittanut paljoakaan viime aikoina CoolBasic.com-sivuston foorumilla. Tämä on ymmärrettävää, mikäli hän ei ole saanut viimeaikoina jatkettua CoolBasic Advancen kehittämistä. Hän pyrkinee pitämään matalaa profiilia, jottei anna lupauksia esimerkiksi CBA:n ensimmäisen kokeiluversion ominaisuuksista tai julkaisuajankohdasta. Mielenkiinto hänellä kuitenkin on projektiin varmasti säilynyt, sillä ensinnä hän ei ole antanut CoolBasicia muiden kehitettäväksi, toiseksi hän uusi CoolBasic.comin foorumin viime syksynä ja sitä ennen siirsi itse sivuston uudelle kotisivutilalle lopetettuaan aiempaan sivutilaansa sidoksissa olleen lehtitilauksen. Kolmanneksi hän maksaa edelleen www.coolbasic.com-domainista vuosi- tai kuukausimaksua.

Se että Zero ei ole antanut CoolBasicin lähdekoodia julkiseen levitykseen – tai edes suljetulle kehitysryhmälle, tarkoittaa mielestäni sitä, että hänellä on tarkat suunnitelmat ja ideat siitä, miten hän haluaa kielen edistyvän. Toisaalta hänellä on myös pieni visio CB3D-pluginin maksullisuudesta, jolloin hän saisi pienen rahallisen korvauksen CoolBasicin eteen näkemästään vaivasta. Jos hän antaisi kielen muiden kehitettäväksi, hän saattaisi menettää tämän edun. En kuitenkaan usko rahan olevan kovinkaan suuri syy CoolBasicin lähdekoodin pitämisessä itsellään.

10 Omat kokemukseni

Aloin tehdä pelejä muistaakseni vuonna 2001 *Clickteamin* pelinteko-ohjelmilla. Niillä sainkin muutaman pelin aikaiseksi, jotka ovat edelleen olemassa kotisivuillani.

Noin vuotta myöhemmin aloin opiskella *QBasic*-ohjelmointikieltä *Pelintekijän Paratiisista* löytämäni oppaan avulla. Pelintekijän Paratiisi on jo aikaa sitten suljettu sivusto, joka sisälsi oppaita, peliprojekteja ja paljon muuta pelien tekemiseen liittyvää.

QBasicilla tein tekstipohjaisia pelejä, kuten *TV-Manageri* (ja sen useat jatko-osat), jossa perustetaan uusi TV-kanava ja päätetään sen ohjelmistosta, mainoksista jne.
Mitään reaaliaikaista peliä en kuitenkaan QBasicilla koskaan tehnyt – lukuunottamatta yhtä pientä *arcade*-peliä, jota en kuitenkaan koskaan julkaissut internetissä.

En muista, milloin kokeilin CoolBasicia ensimmäisen kerran. *Beta 5* oli todennäköisesti ensimmäinen versio, jota kokeilin, joten sen täytyi tapahtua vuonna 2004. Reaaliaikaisia pelejä osasin kuitenkin silloin tehdä vain pelinteko-ohjelmilla, joten niiden ohjelmoiminen ei minulta silloin vielä luonnistunut. Tekstipohjaisissa peleissä olisin CoolBasicia ehkä käyttänyt, mutta QBasic oli tähän tarkoitukseen ehkä hieman ”helpompi” ja sitä paitsi ehdottomasti legendaarisempi. Niinpä en CoolBasiciä heti alkanut käyttää, kun minulla ei ollut tarmoa opetella uutta kieltä.

Vuoden 2005 alussa palasin kuitenkin CoolBasicin pariin. Olin muistaakseni *MBnetissä* nähnyt *Päivän Ohjelma* -listauksessa CoolBasicin ja päättänyt jälleen kokeilla sitä. Loppujen lopuksi sen käyttäminen oli varsin helppoa oppia, kun osasin ohjemoida jo entuudestaan QBasic:llä. Manuaali oli läheinen ystävä uusia komentoja opetellessa, ja CoolBasic rikkoi niitä rajoja, joita QBasic oli asettanut. QBasic on siis DOS-pohjainen ohjelmointikieli, jonka Microsoft kehitti 1980-luvulla. Siinä suurin mahdollinen värimäärä oli 256, ja suurin ruudun resoluutio 640x480. Tosin jos halusi käyttää tätä suurinta resoluutiota, oli tyydyttävä vain 16:een väriin.

10.1 Meneillään olevat projektini

Kerron tässä luvussa muutamasta projektistani, joita minulla on tällä hetkellä meneillään CoolBasicilla. Katsoessani nyt projektikansiotani,

näen siellä yhteensä 54-kansiota. Jokainen sisältää yhden projektin. Paljon olen siis projekteja aloitellut. Tässä luvussa otan käsittelyyn vain näistä suurimmat ja sellaiset, jotka todennäköisimmin jatkan vielä loppuun. Projektit ovat aakkosjärjestyksessä ja suurin osa niistä ei ole pelejä, vaan moottoreita tai ohjelmia, jotka on tarkoitettu avustamaan CoolBasic-pelien luomista.

10.1.1 CBDebug

Koska CoolBasic ei sisällä debuggausmahdollisuuksia, päätin luoda ohjelman, joka hieman parantaa tilannetta. Debuggaus tarkoittaa ohjelman ajamista ja ohjelmakoodin tarkkailemista samaan aikaan. Virheen tai muun yllättävän tapauksen sattuessa debug-toiminto osaa kertoa, millä rivillä oltiin ohjelmaa suorittamassa, kun virhe tapahtui. Lisäksi debuggauksessa voidaan tarkkailla ohjelman muuttujien arvoja. CBDebug ei kuitenkaan kykene ajamaan CB-lähdekoodia, mutta se lukee lähdekoodin ja etsii siitä virheitä, joita CoolBasic-kääntäjä ei löydä. Esimerkiksi kirjoitusvirhe muuttujan nimessä voi olla salakavala virhe, joka ei selviä helposti, mutta aiheuttaa virhetilanteen, kun koodi ei saa asetettua tai luettua muuttujan arvoa oikein – kirjoitusvirheen vuoksi. CBDebug laskee, miten monesti kuhunkin muuttujaan asetetaan arvo, ja miten monesti se luetaan. Jos jompikumpi tulos on nolla, on muuttujassa jotakin vikaa. Lisäksi CBDebug löytää muuttujat, jotka muistuttavat nimiltään toisiaan – kuten esimerkiksi *muuttuja* ja *muutuja*, joista jälkimmäisessä on kirjoitusvirhe.

Muuttujien lisäksi CBDebug tekee samantapaisia toimenpiteitä myös käyttäjän määrittelemille funktioille, tyypeille, taulukoille ja vakioille. Se osaa erottaa koodista CoolBasicin sisään rakennetut komennot ja funktiot, ja se löytää käyttämättömät funktiot, vakiot, taulukot, muuttujat ja tyypit.

CBDebugilla ei ole vielä kotisivua, enkä ole kunnolla maininnut siitä muutenkaan missään muualla kuin tässä dokumentissa. Se on vielä erittäin keskeneräinen, mutta koneisto on varsin hyvällä mallilla ja se kirjoittaa jo mielenkiintoisia raportteja.
CBDebug tulee parhaimmassa tapauksessa säästämään tuntien työn virheiden etsinnässä – etenkin silloin, kun sille annetaan tutkittavaksi useita tuhansia koodirivejä sisältävä ohjelma.

CBDebug ei tee lähdekoodiin yhtään muutosta – se ainoastaan huomauttaa käyttäjää löytämistään seikoista selkeässä raporttitiedostossa. Tämän pohjalta käyttäjä voi itse korjata virheet parhaaksi katsomallaan tavalla.

Ohjelma tulee olemaan OpenSourcea, joten kuka tahansa voi myös kehittää sitä eteenpäin.

10.1.2 Cursed Town

Cursed Town tulee olemaan ylhäältäpäin kuvattu roolipeli, jossa tulee pelastaa pienen kylän kohtalo jumalilta, jotka ovat raivostuneet kyläläisille huonosta temppeliaarteiden vartioinnista, jonka seurauksena aarteet on varastettu. Pelin päähenkilön tulee löytää varastetut aarteet ja palauttaa ne kylään – matkalla kukistaen vihollisia. Tämä projekti on todella alussa, eikä tule valmistumaan vielä pitkään aikaan.

10.1.3 Extended Object System

Järjestelmä, joka lisää CoolBasicin objekteihin lisää ominaisuuksia, kuten vakioliikenopeuden (jolloin objekti liikkuu automaattisesti koko ajan), eri suuntiin vaikuttavia eri suuruisia voimia, objektien liimaamisen kiinni toisiin objekteihin ja paljon muuta. Tämäkin projekti on vasta eirittäin alussa.

10.1.4 Janelas

Ikkunamoottori, joka mahdollistaa CoolBasic-ohjelman sisäisten ikkunoiden luomisen. Projekti on varsin suurissa ongelmissa, sillä ikkunat tarvitsisivat omat käyttöliittymäelementit, joiden toteuttamisessa on todella paljon työtä. Projektista on olemassa keskeneräinen versio osoitteessa URL 10.1.

10.1.5 Madot 4

Jatko-osa *Madot*-pelisarjalle, jonka aiemmat osat olen tehnyt pelinteko-ohjelmilla. Kyseessä on reaaliaikainen, sivulta päin kuvattu, *minä vs. maailma* –tyyppinen matosotapeli. Pelaajalla on vastassaan matoarmeija, jonka sotilaita on tapettava mahdollisimman paljon, ennen kuin itse kuolee. Kentät tuhoutuvat räiskeestä samaan tapaan kuin *Worms*-pelissä. Madot naljailevat toisilleen nasevia kommentteja taistelun keskellä – lisäksi tilanneselostuksesta vastaa tv-selostaja, joka tuo piristävän lisän peliin. Pelaaja luo pelissä itselleen pelaajatilin, jolle karttuu kokemusta ja sen mukana uusia aseita ja kenttiä. Pelaaja voi myös kokemuksen perusteella parantaa ominaisuuksiaan. Pelillä on omat kotisivut osoitteessa URL 10.2.

10.1.6 Memory Handling Script

Olen tehnyt koko jouko skriptimoottoreita, jotka tosin kaikki ovat jääneet keskeneräisiksi. *Memory Handling Script* pyrkii rajaamaan skriptauksen muuttujien arvojen muutteluun. Näillä muuttujilla voidaan sitten vaikuttaa pelin toimintaan. Tällä tavoin voidaan osa pelin koodista toteuttaa pelin ulkopuolisella skriptillä, jolloin skriptiä voi vaihtaa lennosta ja täten muuttaa esimerkiksi tietokonepelaajan vaikeusastetta ja älykkyyttä.

Skriptit mahdollistavat myös pientä korjausta itse pelissä oleviin virheisiin. Skriptimoottorin käyttömahdollisuudet riippuvat siitä, miten lujasti se liitetään kiinni itse peliprojektiin. Jos pelin ja skriptimoottorin

väliin tehdään todella monipuolinen liitos, voi skriptissä käytännössä hallita koko pelin kulkua – kaikkia kentällä olevia objekteja ja pelin muuttujia. Esimerkiksi jokaiselle kentälle voitaisiin tehdä oma skriptitiedosto, jolloin pelimoottoriin voisi tulla kenttäkohtaisia muutoksia – esimerkiksi painovoiman, vihollisten määrän, energialaatikoiden määrän ja kentän muodostuksen suhteen.
Tällä projektilla ei ole vielä kotisivua.

10.1.7 SunBEAM

Tämä on niitä harvoja projekteja, jotka olen saanut valmiiksi. SunBEAM on peli, jossa on tarkoitus saada kentän kaikki kohteet valaistuiksi koskettamalla niitä valonsäteellä, jonka kulkua ohjataan peileillä. Peilien kulman saa itse päättää, ja valonsäde ei saa osua kentällä oleviin esteisiin, eikä valonsäde saa myöskään karata kentän rajojen ulkopuolelle. Jos valonsäde ei pääse kaikkiin kohteisiin, saa kenttää jatkaa edelleen samasta tilanteesta, johon on päässytkin.
SunBEAM on jo valmistunut, mutta aion kehittää sitä vielä eteenpäin. Pelistä löytyy lisää tietoa osoitteessa URL 10.3.

Lähteet

André LaMothe, 2000, *Inside peliohjelmointi*, IT Press, Helsinki

FILE 1: Licence.txt-tiedosto CoolBasicin asennuskansiossa.

URL 1: http://www.coolbasic.com/ (luettu 7.4.2008)

URL 3.1: http://www.coolbasic.com/suomi/ (luettu 7.4.2008)
URL 3.2: http://www.coolbasic.com/suomi/coolbasic.php (luettu 7.4.2008)
URL 3.3: http://www.coolbasic.com/phpBB3/viewtopic.php?f=6&t=3&p=3 (luettu 7.4.2008)
URL 3.4: http://www.coolbasic.com/oldforums/index.php?showtopic=130&view=findpost&p=1213 (luettu 7.4.2008)
URL 3.5: http://www.coolbasic.com/suomi/cbrequirements.php (luettu 7.4.2008)

URL 4.1: http://www.coolbasic.com/suomi/tilester.php (luettu 7.4.2008)
URL 4.2: http://www.coolbasic.com/suomi/tilester/Help/fin1.html (luettu 7.4.2008)
URL 4.3: http://www.coolbasic.com/suomi/tilester/Help/fin2.html (luettu 7.4.2008)
URL 4.4: http://www.coolbasic.com/suomi/tilester/Help/fin6.html (luettu 7.4.2008)
URL 4.5: http://www.coolbasic.com/suomi/tilester/Help/fin3.html (luettu 7.4.2008)

URL 5.1: http://www.coolbasic.com/suomi/resources.php (luettu 7.4.2008)

URL 6.1: http://www.coolbasic.com/phpBB3/ (luettu 7.4.2008)
URL 6.2: http://www.coolbasic.com/phpBB3/viewtopic.php?f=6&t=2 (luettu 7.4.2008)
URL 6.3: http://www.coolbasic.com/phpBB3/viewforum.php?f=10 (luettu 7.4.2008)
URL 6.4: http://www.coolbasic.com/phpBB3/viewforum.php?f=11 (luettu 7.4.2008)
URL 6.5: http://www.coolbasic.com/phpBB3/viewforum.php?f=15 (luettu 7.4.2008)
URL 6.6: http://www.coolbasic.com/oldforums/index.php?showtopic=1714 (luettu 7.4.2008)
URL 6.7: http://cbkk.systec.fi/ (luettu 7.4.2008)
URL 6.8: http://www.coolbasic.com/phpBB3/viewtopic.php?f=9&t=74&start=0&st=0&sk=t&sd=a (luettu 7.4.2008)
URL 6.9: http://cbkk.systec.fi/cbsarjis.php (luettu 7.4.2008)
URL 6.10: http://www.cbcorner.net/ (luettu 7.4.2008)
URL 6.11: http://www.cbcorner.net/articles.php?cat=1 (luettu 7.4.2008)
URL 6.12: http://www.cbcorner.net/files.php?cat=1 (luettu 7.4.2008)
URL 6.13: http://www.cbcorner.net/competition.php (luettu 7.4.2008)

URL 7.1: http://koti.mbnet.fi/cbsdk/ (luettu 7.4.2008)
URL 7.2: http://www.coolbasic.com/oldforums/index.php?showtopic=5798 (luettu 7.4.2008)
URL 7.3: http://charged.relaa.net/a_stigmator.php (luettu 7.4.2008)
URL 7.4: http://www.coolbasic.com/phpBB3/viewtopic.php?f=11&t=35 (luettu 2.4.2008)
URL 7.5: http://koti.mbnet.fi/marcoder/tilegen/ (luettu 7.4.2008)
URL 7.6:

http://www.coolbasic.com/oldforums/index.php?showtopic=3199 (luettu 2.4.2008)

URL 8.1: http://www.cbcorner.net/articles.php?id=11 (luettu 2.4.2008)
URL 8.2: http://www.shnetworks4.net/~asciiwor/pcopy/issue20/pcopy20.html#coolbasic (luettu 2.4.2008)
URL 8.3: http://www.coolbasic.com/oldforums/index.php?showtopic=6846 (luettu 8.4.2008)
URL 8.4: http://www.coolbasic.com/oldforums/index.php?showtopic=76 (luettu 2.4.2008)
URL 8.5: http://www.coolbasic.com/oldforums/index.php?showtopic=3 (luettu 2.4.2008)
URL 8.6: http://www.coolbasic.com/suomi/changes.html (luettu 3.4.2008)

URL 9.1: http://www.coolbasic.com/oldforums/index.php?showtopic=3047 (luettu 7.4.2008)
URL 9.2: http://www.coolbasic.com/oldforums/index.php?showtopic=364 (luettu 7.4.2008)
URL 9.3: http://www.coolbasic.com/oldforums/index.php?showtopic=3051 (luettu 7.4.2008)
URL 9.4: http://www.coolbasic.com/oldforums/index.php?showtopic=5809 (luettu 5.4.2008)
URL 9.5: http://www.coolbasic.com/phpBB3/viewtopic.php?f=13&t=77 (luettu 7.4.2008)
URL 9.6: http://www.coolbasic.com/phpBB3/viewtopic.php?f=13&t=77&st=0&sk

=t&sd=a&start=80#p3366 (luettu 7.4.2008)
URL 9.7: http://notepad-plus.sourceforge.net (luettu 7.4.2008)

URL 10.1: http://koti.mbnet.fi/jare1/janelas/ (luettu 5.4.2008)
URL 10.2: http://www.madot4.urli.net (luettu 5.4.2008)
URL 10.3: http://akiky.nwps.ws/pelit/index.php?topic=25.0 (luettu 5.4.2008)

Liitteet

LIITE 1

```
32
33                  'Asetetaan ikkunan koko
34                  SCREEN 640,480
35
36                  //Määritetään pelin asetukset
37
38                  Const PALKIN_LEVEYS = 100
39                  Const PALKIN_PAKSUUS = 5
40                  Const PALKIN_NOPEUS = 10
41                  Const PALLON_HALKAISIJA = 10
42                  Const PALLON_NOPEUS = 4
43
44
45                  //Luodaan mallikuva palkista
46
47                  'Tämä luo kuvan muistiin
48                  palkki = MakeImage(PALKIN_LEVEYS, PALKIN_PAKSUUS)
49
50                  'Ohjataan piirtäminen luomaamme kuvaan
51                  DrawToImage palkki
52
53                  'Piirretään palkki
54                  Color cbDarkRed
55                  Box 0,0, PALKIN_LEVEYS,PALKIN_PAKSUUS, ON
56
57
58                  //Luodaan kuva pallosta
59                  pallo = MakeImage(PALLON_HALKAISIJA,PALLON_HALKAISIJA)
60                  DrawToImage pallo
61                  Color cbBlue
62                  Circle 0,0, PALLON_HALKAISIJA, ON
63
```

```
64
65                  'Ohjataan piirtäminen takaisin näytölle
66                  DrawToScreen
67
68
69                  'Asetetaan pelaaja vaakasuunnassa keskelle ruutua ja
pystysuunnassa lähelle ikkunan alalaitaa
70                  pelaajaX = (ScreenWidth() - PALKIN_LEVEYS)/2
71                  pelaajaY = ScreenHeight()-30
72
73                  'Asetetaan pallo keskelle ruutua
74                  palloX = ScreenWidth()/2
75                  palloY = ScreenHeight()/2
76                  palloOlemassa = True
77
78                  'Asetetaan pallolle suunta. Suunta voi olla liukuluku väliltä
0-360.
79                  '#-merkki tekee muuttujasta liukulukutyyppisen.
80                  palloSuunta# = Rnd(360)
81
82                  //Pelin päälenkki
83
84                  Repeat
85
86                       //Pelaajan ohjaus
87
88                       'Nuolet vasemmalle ja oikealle
89                       If KeyDown(cbKeyLeft) Then pelaajaX = pelaajaX -
PALKIN_NOPEUS
90                       If KeyDown(cbKeyRight) Then pelaajaX = pelaajaX +
PALKIN_NOPEUS
91
92                       'Tarkistetaan, ettei pelaaja mene ulos ruudusta
93                       If pelaajaX < 0 Then pelaajaX = 0
94                       If pelaajaX > ScreenWidth() - PALKIN_LEVEYS Then pelaajaX
= ScreenWidth() - PALKIN_LEVEYS
95
96
97                       //Piirretään pelaaja
98                       DrawImage palkki, pelaajaX,pelaajaY
99
100
101                      If palloOlemassa Then
102                                //Pelipallon liikkuminen
103
104                               'Liikutetaan palloa Sinin ja Cosinin avulla.
```

```
105                                 palloX = palloX +
Cos(palloSuunta)*PALLON_NOPEUS
106                                 palloY = palloY -
Sin(palloSuunta)*PALLON_NOPEUS
107
108                                 'Tarkistetaan törmäykset
109                                 If DotInArea(palloX,palloY,
pelaajaX,pelaajaY, pelaajaX+PALKIN_LEVEYS-1, pelaajaY+PALKIN_PAKSUUS-1) Then
110                                             'Pallo ON törmännyt pelaajaan
111                                             'Laitetaan pallo kimpoamaan
palkista, kun palkin kulma ON 180 (tai 0, miten vain haluaa ajatella)
112                                             palloSuunta =
BounceAngle(palloSuunta,180)
113                                 End If
114
115                                 If palloX <= 0 Then
116                                             'Pallo ON törmännyt ruudun
vasempaan reunaan
117                                             'Laitetaan pallo kimpoamaan
seinästä, jonka kulma ON 90 (tai 270)
118                                             palloSuunta =
BounceAngle(palloSuunta,90)
119                                 End If
120
121                                 If palloX >= ScreenWidth() Then
122                                             'Pallo ON törmännyt ruudun
oikeaan reunaan
123                                             'Laitetaan pallo kimpoamaan
seinästä, jonka kulma ON 90 (tai 270)
124                                             palloSuunta =
BounceAngle(palloSuunta,90)
125                                 End If
126                     End If
127
128                     'Sijoitetaan pallo kentän keskelle, jos pallo ei ole
kentällä, ja jos painetaan Enteriä.
129                     If KeyHit(cbKeyReturn) And palloOlemassa=False Then
130                                 palloOlemassa = True
131                                 palloSuunta = Rnd(360)
132                                 palloX = ScreenWidth()/2
133                                 palloY = ScreenHeight()/2
134                     End If
135
136                     //Piirretään pelipallo
137                     DrawImage pallo, palloX-PALLON_HALKAISIJA/2,palloY-
PALLON_HALKAISIJA/2
138
```

```
139                    'Päivitetään kuva näytölle
140                    DrawScreen
141
142               Forever
143
144               Function DotInArea(dotX,dotY, aX1,aY1, aX2,aY2)
145                    If dotX >= aX1 And dotX <= aX2 Then
146                              If dotY >= aY1 And dotY <= aY2 Then
147                                        Return True
148                              End If
149                    End If
150                    Return False
151               End Function
152
153               Function BounceAngle(angle1#,angle2#)
154                    Return WrapAngle(angle2-(angle1-angle2))
155               End Function
156
```

LIITE 2

```
1
2                 'Asetetaan ikkunan koko
3                 SCREEN 640,480
4
5                 //Määritetään pelin asetukset
6
7                 Const PALKIN_LEVEYS = 100
8                 Const PALKIN_PAKSUUS = 5
9                 Const PALKIN_NOPEUS = 10
10                Const PALLON_HALKAISIJA = 10
11                Const PALLON_NOPEUS = 4
12
13
14                //Luodaan mallikuva palkista
15
16                'Tämä luo kuvan muistiin
17                palkki = MakeImage(PALKIN_LEVEYS, PALKIN_PAKSUUS)
18
19                'Ohjataan piirtäminen luomaamme kuvaan
20                DrawToImage palkki
21
22                'Piirretään palkki
23                Color cbDarkRed
24                Box 0,0, PALKIN_LEVEYS,PALKIN_PAKSUUS, ON
25
```

```
26
27                  //Luodaan kuva pallosta
28                  pallo = MakeImage(PALLON_HALKAISIJA,PALLON_HALKAISIJA)
29                  DrawToImage pallo
30                  Color cbBlue
31                  Circle 0,0, PALLON_HALKAISIJA, ON
32
33
34                  'Ohjataan piirtäminen takaisin näytölle
35                  DrawToScreen
36
37
38                  'Asetetaan pelaaja vaakasuunnassa keskelle ruutua ja
pystysuunnassa lähelle ikkunan alalaitaa
39                  pelaajaX = (ScreenWidth() - PALKIN_LEVEYS)/2
40                  pelaajaY = ScreenHeight()-30
41
42                  'Asetetaan tietokone vaakasuunnassa keskelle ruutua ja
pystysuunnassa lähelle ikkunan ylälaitaa
43                  tietokoneX = (ScreenWidth() - PALKIN_LEVEYS)/2
44                  tietokoneY = 30
45
46                  'Asetetaan pallo keskelle ruutua
47                  palloX = ScreenWidth()/2
48                  palloY = ScreenHeight()/2
49                  palloOlemassa = True
50
51                  'Asetetaan pallolle suunta. Suunta voi olla liukuluku väliltä
0-360.
52                  '#-merkki tekee muuttujasta liukulukutyyppisen.
53                  palloSuunta# = Rnd(360)
54
55                  //Pelin päälenkki
56
57                  Repeat
58
59                       //Pelaajan ohjaus
60
61                       'Nuolet vasemmalle ja oikealle
62                       If KeyDown(cbKeyLeft) Then pelaajaX = pelaajaX -
PALKIN_NOPEUS
63                       If KeyDown(cbKeyRight) Then pelaajaX = pelaajaX +
PALKIN_NOPEUS
64
65                       'Tarkistetaan, ettei pelaaja mene ulos ruudusta
66                       If pelaajaX < 0 Then pelaajaX = 0
```

```
67                      If pelaajaX > ScreenWidth() - PALKIN_LEVEYS Then pelaajaX
= ScreenWidth() - PALKIN_LEVEYS
68
69
70                      //Piirretään pelaaja
71                      DrawImage palkki, pelaajaX,pelaajaY
72
73
74                      //Tietokoneen ohjaus
75
76                      'Liikkuminen vasemmalle ja oikealle
77                      etäisyys_vaaka = Abs(tietokoneX+PALKIN_LEVEYS/2 - palloX)
78                      etäisyys_pysty = Abs(tietokoneY - palloY)
79                      If etäisyys_vaaka > Rand(30,60) And etäisyys_pysty < 200
And palloSuunta < 180 Then
80                                      'Liikutamme tietokonetta kohti palloa
81                                      If (tietokoneX < palloX) Then
82                                                      tietokoneX = tietokoneX +
Min(etäisyys_vaaka, PALKIN_NOPEUS)
83                                      Else
84                                                      tietokoneX = tietokoneX -
Min(etäisyys_vaaka, PALKIN_NOPEUS)
85                                      End If
86                      End If
87
88                      'Tarkistetaan, ettei tietokone mene ulos ruudusta
89                      If tietokoneX < 0 Then tietokoneX = 0
90                      If tietokoneX > ScreenWidth() - PALKIN_LEVEYS Then
tietokoneX = ScreenWidth() - PALKIN_LEVEYS
91
92
93                      //Piirretään tietokone
94                      DrawImage palkki, tietokoneX,tietokoneY
95
96                      If palloOlemassa Then
97                                      //Pelipallon liikkuminen
98
99                                      'Liikutetaan palloa Sinin ja Cosinin avulla.
100                                     palloX = palloX +
Cos(palloSuunta)*PALLON_NOPEUS
101                                     palloY = palloY -
Sin(palloSuunta)*PALLON_NOPEUS
102
103                                     'Tarkistetaan törmäykset
104                                     If DotInArea(palloX,palloY,
pelaajaX,pelaajaY, pelaajaX+PALKIN_LEVEYS-1, pelaajaY+PALKIN_PAKSUUS-1) Then
105                                                     'Pallo ON törmännyt pelaajaan
```

```
106                                         'Laitetaan pallo kimpoamaan
palkista, kun palkin kulma ON 180 (tai 0, miten vain haluaa ajatella)
107                                         palloSuunta =
BounceAngle(palloSuunta,180)
108                                 End If
109
110                                 If DotInArea(palloX,palloY,
tietokoneX,tietokoneY, tietokoneX+PALKIN_LEVEYS-1, tietokoneY+PALKIN_PAKSUUS-1)
Then
111                                         'Pallo ON törmännyt
tietokoneeseen
112                                         'Laitetaan pallo kimpoamaan
palkista, kun palkin kulma ON 180 (tai 0, miten vain haluaa ajatella)
113                                         palloSuunta =
BounceAngle(palloSuunta,180)
114                                 End If
115
116                                 If palloX <= 0 Then
117                                         'Pallo ON törmännyt ruudun
vasempaan reunaan
118                                         'Laitetaan pallo kimpoamaan
seinästä, jonka kulma ON 90 (tai 270)
119                                         palloSuunta =
BounceAngle(palloSuunta,90)
120                                 End If
121
122                                 If palloX >= ScreenWidth() Then
123                                         'Pallo ON törmännyt ruudun
oikeaan reunaan
124                                         'Laitetaan pallo kimpoamaan
seinästä, jonka kulma ON 90 (tai 270)
125                                         palloSuunta =
BounceAngle(palloSuunta,90)
126                                 End If
127
128                                 'Tarkistetaan maalit
129                                 If palloY < 0 Then
130                                         'Pelaaja ON tehnyt maalin
131                                         palloOlemassa = False
132                                         pisteet_pelaaja + 1
133                                 End If
134
135                                 If palloY > ScreenHeight() Then
136                                         'Tietokone ON tehnyt maalin
137                                         palloOlemassa = False
138                                         pisteet_tietokone + 1
139                                 End If
```

```
140                     End If
141
142                     'Sijoitetaan pallo kentän keskelle, jos pallo ei ole
kentällä, ja jos painetaan Enteriä.
143                     If KeyHit(cbKeyReturn) And palloOlemassa=False Then
144                               palloOlemassa = True
145                               palloSuunta = Rnd(360)
146                               palloX = ScreenWidth()/2
147                               palloY = ScreenHeight()/2
148                     End If
149
150                     //Piirretään pelipallo
151                     DrawImage pallo, palloX-PALLON_HALKAISIJA/2,palloY-
PALLON_HALKAISIJA/2
152
153                     //Tulostetaan pisteet
154                     Color cbWhite
155                     Text 10,50, "Tietokoneen pisteet: " + pisteet_tietokone
156                     Text 10,ScreenHeight()-65, "Pelaajan pisteet: " +
pisteet_pelaaja
157
158                     'Päivitetään kuva näytölle
159                     DrawScreen
160
161                Forever
162
163                Function DotInArea(dotX,dotY, aX1,aY1, aX2,aY2)
164                     If dotX >= aX1 And dotX <= aX2 Then
165                               If dotY >= aY1 And dotY <= aY2 Then
166                                         Return True
167                               End If
168                     End If
169                     Return False
170                End Function
171
172                Function BounceAngle(angle1#,angle2#)
173                     Return WrapAngle(angle2-(angle1-angle2))
174                End Function
175
```