XNA 2D ohjelmoinnin perusteet

Paavo Räisänen

www.ohjelmoimaan.net

Sisällysluettelo

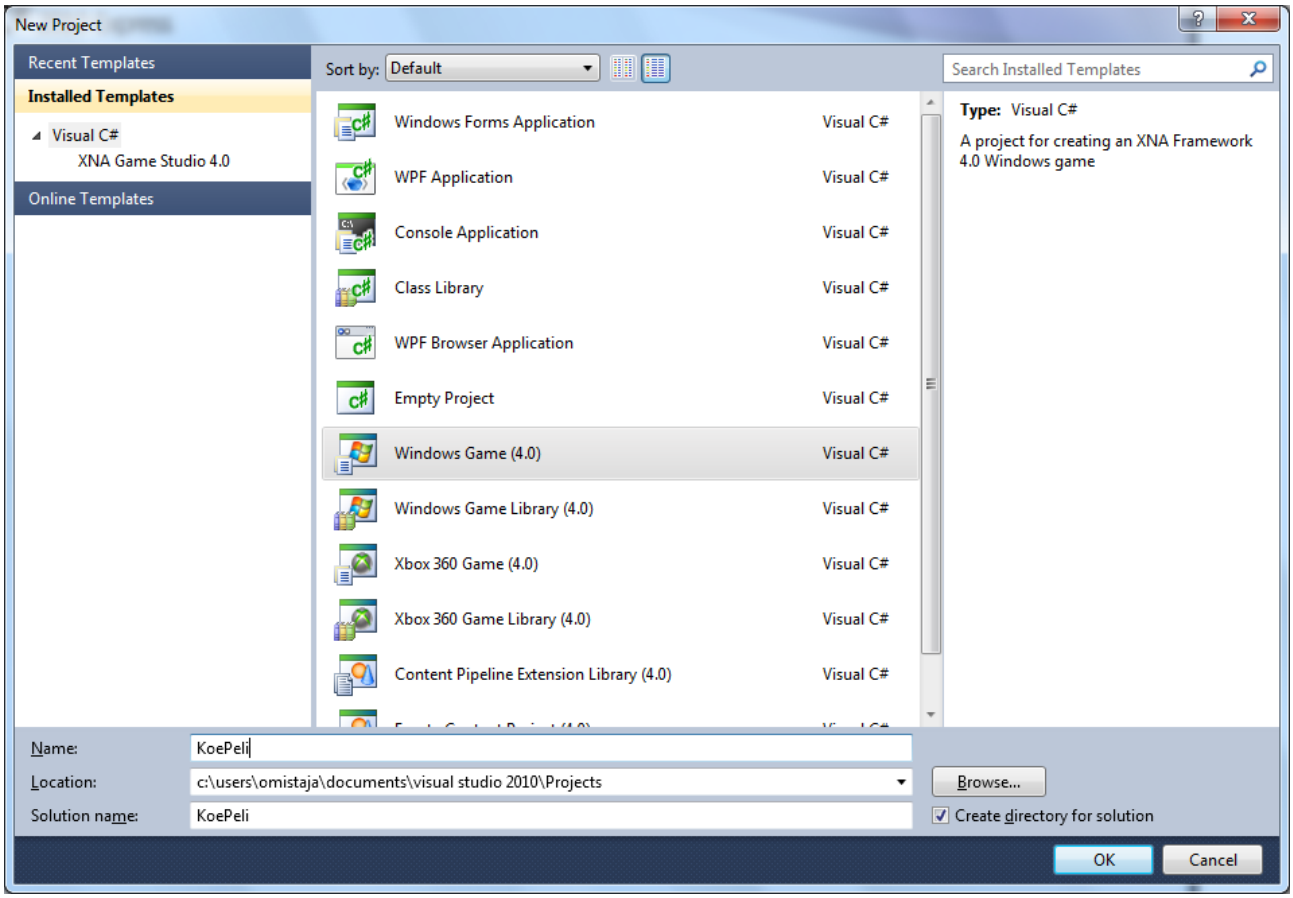
1: Alkusanat

XNA on Microsoftin erityisesti pelejä varten kehittämä ohjelmointiympäristö. XNA:lla voi ohjelmoida sekä perinteisiä tietokonepelejä, että konsoli- ja matkapuhelinpelejä. Ympäristön saa MonoDevelopin kautta toimimaan Windowsin lisäksi myös Linuxilla ja Macillä.
XNA ohjelmointi vaatii Visual Studio ohjelman ja C# ohjelmoinnin perusteiden osaamisen. Riittävät perustaidot saa lukemalla näiden sivujen C# perusoppaan ja tutustumalla hieman C# Oliot oppaaseen. Lähinnä olio-ohjelmoinnista on kuitenkin tässä tiedettävä vain käsitteet luokka, olio ja metodi. Visual Studion asennus neuvotaan C# perusoppaassa.
XNA ympäristö asennetaan Visual Studioon. Ympäristön voi asentaa ilmaiseksi osoitteesta http://www.microsoft.com/download/en/details.aspx?id=23714 .Tässä oppaassa käytetään XNA 4.0 versiota. XNA 4.0 on osittain muuttunut XNA 3.0 versiosta. Ympäristö asentuu helpoilla valinnoilla automaattisesti Visual Studioon.

Tässä oppaassa käydään läpi 2D XNA C# peliohjelmoinnin perusteet, ja opastetaan esimerkkipelin ”Salaman Torjuntaa” tekeminen. Peli on pelattavissa osoitteessa www.pelilakka.com .

2.Ensimmäinen ohjelma

Käynnistä Visual Studio. Klikkaa ”new project”. Sinulle aukeaa tämä näkymä, josta valitse ”Windows Game (4.0)” ja anna nimeksi vaikka ”KoePeli”, ja klikkaa ”OK”.



Voit jo testata ohjelmaa valitsemalla ”Debug/Start degugging”. Se on jo ensimmäinen ”peli”, vaikka näyttö onkin tyhjä. Takana pyörii jo XNA koodi, jonka näet Game1.cs tiedostosta, ja näyttää seuraavalta:

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

namespace KoePeli
{
    /// <summary>
    /// This is the main type for your game
    /// </summary>
```

```
    public class Game1 : Microsoft.Xna.Framework.Game
    {
        GraphicsDeviceManager graphics;
        SpriteBatch spriteBatch;

        public Game1()
        {
            graphics = new GraphicsDeviceManager(this);
            Content.RootDirectory = "Content";
        }

        /// <summary>
        /// Allows the game to perform any initialization it needs to
before starting to run.
        /// This is where it can query for any required services and load
any non-graphic
        /// related content.  Calling base.Initialize will enumerate
through any components
        /// and initialize them as well.
        /// </summary>
        protected override void Initialize()
        {
            // TODO: Add your initialization logic here

            base.Initialize();
        }

        /// <summary>
        /// LoadContent will be called once per game and is the place to
load
        /// all of your content.
        /// </summary>
        protected override void LoadContent()
        {
            // Create a new SpriteBatch, which can be used to draw
textures.
            spriteBatch = new SpriteBatch(GraphicsDevice);

            // TODO: use this.Content to load your game content here
        }

        /// <summary>
        /// UnloadContent will be called once per game and is the place
to unload
        /// all content.
        /// </summary>

        protected override void UnloadContent()
        {
            // TODO: Unload any non ContentManager content here
        }
```

```
        /// <summary>
        /// Allows the game to run logic such as updating the world,
        /// checking for collisions, gathering input, and playing audio.
        /// </summary>
        /// <param name="gameTime">Provides a snapshot of timing
values.</param>
        protected override void Update(GameTime gameTime)
        {
            // Allows the game to exit
            if (GamePad.GetState(PlayerIndex.One).Buttons.Back ==
ButtonState.Pressed)
                this.Exit();

            // TODO: Add your update logic here

            base.Update(gameTime);
        }

        /// <summary>
        /// This is called when the game should draw itself.
        /// </summary>
        /// <param name="gameTime">Provides a snapshot of timing
values.</param>
        protected override void Draw(GameTime gameTime)
        {
            GraphicsDevice.Clear(Color.CornflowerBlue);

            // TODO: Add your drawing code here

            base.Draw(gameTime);
        }
    }
}
```

Toinen takana oleva koodi on ”Program.cs” tiedostossa, eikä se oikeastaan tee muuta kuin käynnistää ”Game1.cs” tiedoston, eikä aina ole edes tarpeen muuttaa ”Program.cs” tiedostota. ”Program.cs” tiedosto näyttää tältä:

```
using System;

namespace KoePeli
{
#if WINDOWS || XBOX
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        static void Main(string[] args)
        {
```

```
            using (Game1 game = new Game1())
            {
                game.Run();
            }
        }
    }
#endif
}
```

”Game1.cs” tiedosto on ohjelmakoodi, missä pelin ”runko” pyörii. Siihen voi sitten lisätä luokkia ym.
Valmiiksi annetaan kaksi luokkatason muuttujaa `GraphicsDeviceManager` ja `SpriteBatch.` `GraphicsDeviceManager` on objekti, jonka aulla käytetään grafiikkalaitteita, ja kaikki näytöllä tapahtuva kulkee sen kautta.
`GraphicsDeviceManager` objektilla on ominaisuus `GraphicsDevice` joka edustaa laitteen, jolla peliä pelataan, todellista grafiikkalaitetta.
`SpriteBatch` on ydinobjekti spritejen piirtämisessä. Tietokonetermi ”sprite” tarkoittaa kuvaa, joka on integroitu osaksi laajenpaanäkumää. XNA:ssa jokseenkin kaikki, mitä näytölle piirretään on spritejä (pelaaja, muut liikkuvat kohteet, tausta, viivat, neliöt, teksti jne.).
`Initialize` metodissa alustetaan ”Game1” objektiin liittyvät muuttujat ja muut objektit.
`LoadContent` metodissa ladataan peliin kuvat, äänet ja tekstifontit. `LoadContent` metodia kutsutaan `Initialize` metodin jälkeen pelin alussa, ja mikäli pelaaja muuttaa näyttöasetuksia.
Seuraavaksi siirrytään itse pelisilmukkaan. `Update` silmukkaan laitetaan pelisilmukka, jossa ohjataan pelin tapahtumia. XNA kutsuu automaattisesti `Update` silmukkaa 60 kertaa sekunnissa. Aina `Update`:n suorittamisen jälkeen peli kutsuu `Draw` metodia. Peli pyörii tässä silmukassa. `Update` ikäänkuin ohjaa peliä, eli siitä kutsutaan muita tapahtumia. `Draw` metodiin pyritään saamaan kaikki pelin piirtämiseen liittyvä, ja mahdollisimman vähän muuta.
Seuraavassa `Draw` metodin lauseessa asetetaan näytön taustaväri:

```
GraphicsDevice.Clear(Color.CornflowerBlue);
```

Tässä näytön väriksi asetetaan `CornflowerBlue` .Värejä saa näkyviin lisää kirjoittamalla `Color` ja sen jälkeen piste, jolloin Visual Studio ehdottaa eri värivaihtoehtoja. Värin voi myös määritellä itse antamalla muuttujalle RGB arvot seuraavasti:

```
Color vari = new Color(255, 255, 0);
```

Silloin `GraphicsDevice` käskyä kutsutaan seuraavasti:

```
GraphicsDevice.Clear(vari);
```

3: Pelikehityksen ero sovelluskehitykseen

Pelikehityksessä peli pyörii silmukassa, jossa pelin pitää jatkuvasti kysyä järjestelmältä, onko pelaaja klikannut hiirtä, painanut jotain näppäintä jne., sensijaan, että järjestelmä kertoisi pelille, että näin on tapahtunut (kuten sovelluskehityksessä). Pelissä siis koko ajan kysytään yleensä `Update` silmukassa, että jos pelaaja on liikuttanut hiirtä, tai painanut vaikka ”nuoli oikealle” näppäintä, tarkistetaan esim. onko pelaaja törmännyt johonkin. Sitten kutsutaan esim. metodia, jossa kerrotaan, mitä esim. törmäyksen sattuaessa tapahtuu. Samaan aikaan peli pyörii koko ajan. Kaikki spritet, jotka on ohjelmoitu liikkumaan, liikkuvat taustalla ja erilaiset ohjelmoidut tapahtumat pyörivät pelisilmukassa koko ajan, samalla kun tarkkaillaan hiirtä, pelikonsolia ja/ tai näppäimistöä.
Idea on, että kaikki liikuttaminen pyritään tekemään Update() metodissa, ja piirtäminen Draw() metodissa, vaikka tässä oppaassa neuvotussa pelissä ammuksen liikuttaminen suurelta osin tehdäänkin Draw() metodissa. Kaikkia pelin tapahtumia, ja tekoälyjä pyritään ohjaamaan Update() metodissa, ja kutsumaan sieltä tarvittavia toimintoja. Draw() suoritaan myös aina Update() metodin jälkeen, eli molempia oletusarvoisesti kutsutaan 60 kertaa sekunnissa, ellet muuta sitä.

4: Kehysnopeus

Oletus kehysnopeus on, että Update silmukkaa kutsutaan 60 kertaa sekunnissa. Sitä voi muuttaa seuraavalla käskyllä:

```
TargetElapsedTime = new TimeSpan(0,0,0,0,50);
```

Tässä kehysnopeudeksi asetetaan, että kehystä kutsutaan joka 50:s millisekuntti, eli 20 kertaa sekunnissa.

5: Esimerkkipelin ”Salaman Torjuntaa” tekemisen aloitus

Tämä opas neuvoo esimerkkipelin tekemisen. waw äänitiedostot on laitettu kommenteiksi, ja on käytetty mp3 tiedostoja. Kuvat joudut lisäämään ja tekemään itse.

Avaa uusi projekti, ja anna nimeksi ”SalamanTorjuntaa”. Muokkaa ”Game1()” konstruktoria seuraavasti:

```
public Game1()
        {

            graphics = new GraphicsDeviceManager(this);
            Content.RootDirectory = "Content";

            IsMouseVisible = true;

            graphics.PreferredBackBufferWidth = 1100;
            graphics.PreferredBackBufferHeight = 800;
#if   !DEBUG
               graphics.IsFullScreen       true;
#endif


        }
```

Tässä asetetaan näytön leveydeksi 1100 ja korkeudeksi 800. #if ja #endif ovat sellaista tilannetta varten, ettei käyttäjän näytön asetukset salli näin isoa näyttöä. `IsMouseVisible = true;` lause saa aikaan sen, että hiiren nuoli piirretään näyttöön. Jos sen asetta false :ksi, nuolta ei näy, vaikka hiiri muuten toimii, ja sitä voi käyttää pelaamiseen.

Lisää Game1 luokkaan luokkatason muuttujat:

```
public class Game1 : Microsoft.Xna.Framework.Game
    {
        GraphicsDeviceManager graphics;
        SpriteBatch spriteBatch;

        //Määritellään pelitilaa varten enum lueteltu tyyppi
        enum peliTila { aloitus, ohje, pelaa, lopetus };

        //Määritellään peliTilaNyt ja asetetaan alkuarvoksi "aloitus"
        peliTila peliTilaNyt = peliTila.aloitus;
```

Lisää `Update` metodiin seuraavat rivit, joissa rakennellaan ohjelman perusrakenne, eli tarkkaillaan, mikä peliTila on kyseessä, ja toimitaan sen mukaisesti. Siihen voisi halutessaan ohjelmoida myös levelit.

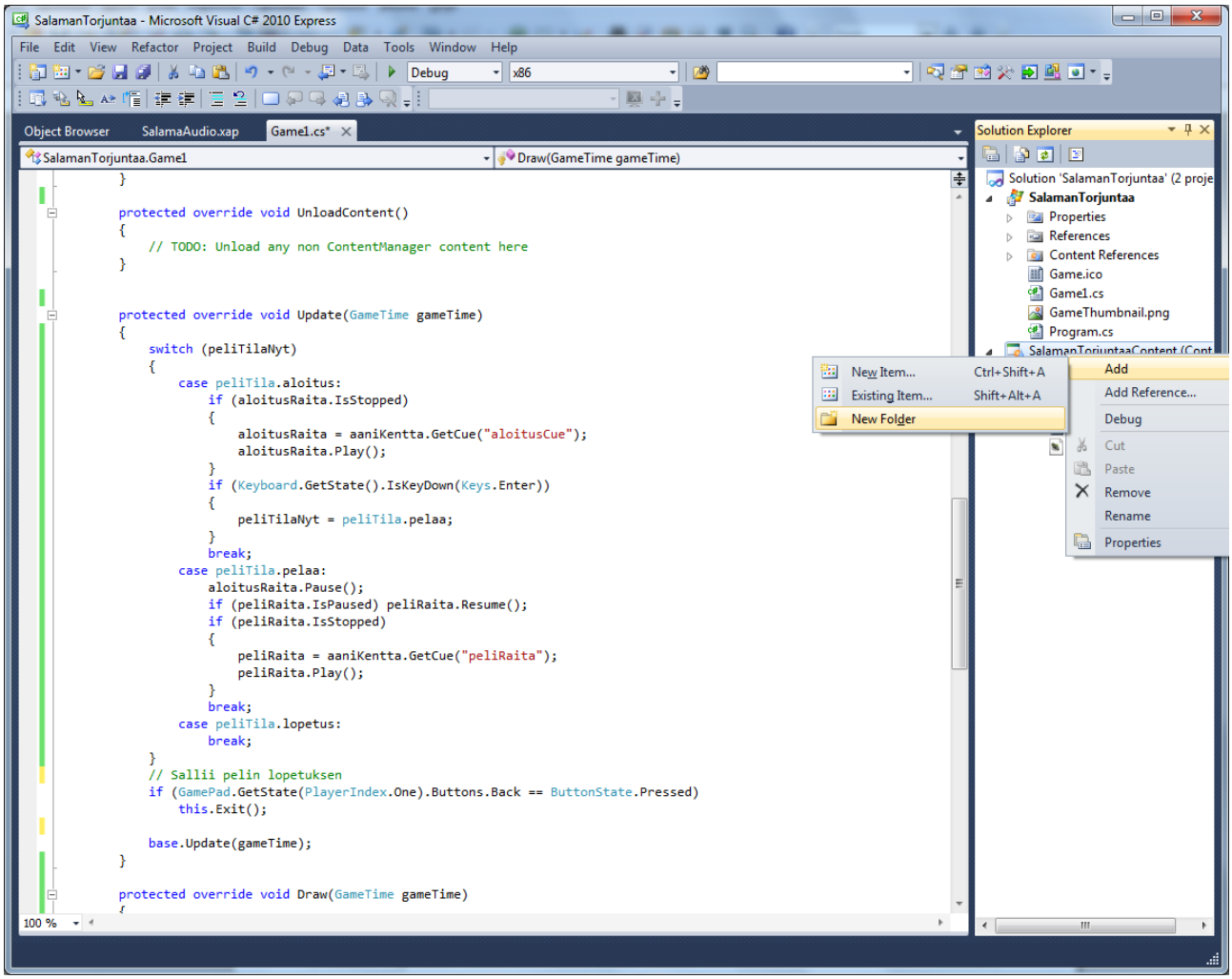
```
 protected override void Update(GameTime gameTime)
        {
            switch (peliTilaNyt)
            {
                case peliTila.aloitus:
                    if (aloitusRaita.IsStopped)
                    {
                        aloitusRaita = aaniKentta.GetCue("aloitusCue");
                        aloitusRaita.Play();
                    }

                    //Enterin painalluksen lukeminen
                    if (Keyboard.GetState().IsKeyDown(Keys.Enter))
                    {
                        peliTilaNyt = peliTila.pelaa;
                    }
                    break;
                case peliTila.pelaa:
                    aloitusRaita.Pause();
                    if (peliRaita.IsPaused) peliRaita.Resume();
                    if (peliRaita.IsStopped)
                    {
                        peliRaita = aaniKentta.GetCue("peliRaita");
                        peliRaita.Play();
                    }
                    break;
                case peliTila.lopetus:
                    break;
            }
            // Sallii pelin lopetuksen
            if (GamePad.GetState(PlayerIndex.One).Buttons.Back ==
ButtonState.Pressed)
                this.Exit();

            base.Update(gameTime);
        }
```

6: Uuden kansion luominen Content solmuun

Klikkaa alla olevan kuvan mukaisesti oikealla valikossa olevaa ”SalamanTorjuntaaContent” solmua hiiren oikealla näppäimellä, ja valitse ”Add” ja ”New Folder”. Content solmuun tulee yleensä ainakin kolme itsetehtyä kansiota, eli ”Audio”, ”Fontit”, ja ”Kuvat”. Voit luoda ne jo valmiiksi.

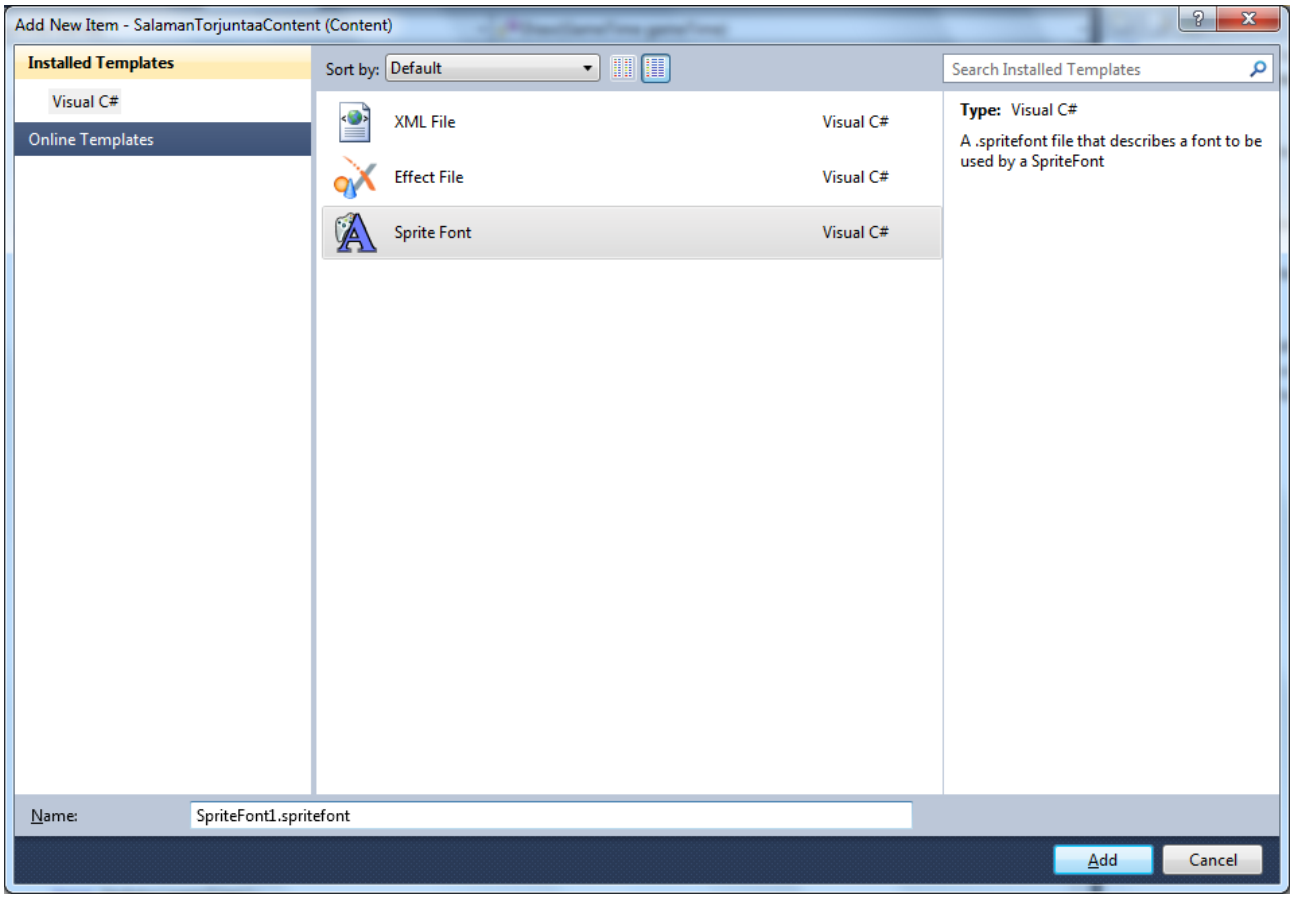


7: Fontin lisääminen

XNA:ssa myös teksti kirjoitaan spriteBach :einä, eli käytetään spritefontteja. Fontit on jokainen eri fontin koko ja tyyli määriteltävä erikseen ja lisättävä äskettäin lisäämääsi ”Fontit” kansioon ja ladattava LoadContent() metodissa ohjelmaan kuuluvaksi.

Uusi fontti lisätään seuraavasti. Klikkaa ”Fontit” kansiota hiiren oikealla näppäimellä. Valitse ”Add” ja ”New Item”.

Sinulle aukeaa seuraava ikkuna:



Valitse kolmesta vaihtoedosta ”Sprite Font”, anna nimeksi Fontti1.spritefont ja paina ”Enter” (tai ”Add” ikkunan alta).

Nyt sinulle aukeaa koodia, johon myöhemmin pääset käsiksi klikkaamalla fontin nimeä ”Fontit” kansiossa. Muutetaan hieman koodia.

XNA:ssa fontit oletusarvoisena eivät hyväksy ääkkösiä. Muutos on kuitenkin helppo tehdä. Muokkaa seuraavaa koodipätkää aivan koodin lopussa:

```
<CharacterRegions>
      <CharacterRegion>
        <Start>&#32;</Start>
        <End>&#126;</End>
      </CharacterRegion>
    </CharacterRegions>
  </Asset>
</XnaContent>
```

Kun <End> </End> tagien väliin suurentaa luvun 255 :een, fonteissa hyväksytään myös ääkköset.

Eli lopullinen koodipätkä on seuraavanlainen:

```
<CharacterRegions>
      <CharacterRegion>
        <Start>&#32;</Start>
        <End>&#255;</End>
      </CharacterRegion>
    </CharacterRegions>
  </Asset>
</XnaContent>
```

Koodissa voi myös tehdä muita muutoksia. Seuraavassa kohdassa voi muutta koodin tyyliä kirjoittamalla tyylin <Style> </Style> tagien väliin:

```
<!--
    Style controls the style of the font. Valid entries are "Regular",
"Bold", "Italic",
    and "Bold, Italic", and are case sensitive.
    -->
    <Style>Regular</Style>
```

Seuraavassa kohdassa taasen voi muuttaa fontin kokoa:

```
<Size>14</Size>
```

8: Tekstin kirjoittaminen

Tulostaaksemme aloitusikkunaan tekstin ”Paina Enter aloittaaksesi, on Draw metodiin lisättävä seuraava koodi

```
protected override void Draw(GameTime gameTime)
{
 GraphicsDevice.Clear(Color.Black);

 spriteBatch.Begin();

 if (peliTilaNyt == peliTila.aloitus)
 {
  spriteBatch.DrawString(spriteFontti, "Paina Enter aloittaaksesi",
     new Vector2(400, 300), Color.Yellow, 0, Vector2.Zero,
     1, SpriteEffects.None, 0);
 }

 spriteBatch.End();

 base.Draw(gameTime);
}
```

Kaiken piirettävän ja kirjoitettavan on oltava `spriteBatch.Begin()` ja `spriteBatch.End()` käskyjen välissä ennen base.Draw(gameTime) kutsua. Tässä esimerkissä on myös lisätty if lause tarkistamaan, onko peli aloitustilassa. Draw metodin idea on, että se piirtää 60 kertaa sekunnissa koko näytön uudelleen. Se siis yhtä monesti tyhjentää näytön, ja piirtää sen, mitä sille uudelleen syötetään. Kun siirrytään peliTila.pelaa tilaan, tätä koodin kohtaa ei enään suoriteta, eli teksti häipyy näytöltä.
`spriteBatch.DrawString` käskyn ensimmäisenä parametrina on fontin nimi, sitten kirjoitettava teksti. Jos haluat kirjoittaa esim. ”Pisteet pistemaara”, on kirjoitettava ”Pisteet” + pistemaara, tai jos haluat lattaa pelkän numeron, on laitettava silti eteen tyhjät lainausmerkit, eli on kirjoitettava ””+numeromuutuja.
Seuraava parametri on Vector2 tyyppinen, ja siinä anntaan koordinantit (x,y), josta alkaen teksti kirjoitetaan. Sitten on väri. Loput parametrit ovat harvemmin käytössä.

Ennenkuin uusi fontti toimii, ja ohjelman voi ajaa, on kuitenkin lisättävä Game1 luokkaan luokkatason muuttuja

```
SpriteFont spriteFontti;
```

ja ladattava uusi fontti LoadContent() metodissa:

```
spriteFontti = Content.Load<SpriteFont>(@"Fontit\Fontti1");
```

9: Taustakuvan lisääminen

Aloituskuvan aloitusnäkymään saat seuraavasti. Aiemmin mainittiin jo, että Content solmuun lisätään kansio ”Kuvat”. Jos et ole vielä lisännyt sitä, lisää nyt. Klikkaa sitten ”Kuvat” kansiota hiiren oikealla, ja valitse ”Add” ja ”Existing Item”. Valitse sitten tietokoneeltasi haluttu oikean kokoinen kuva. Tässä kuva on nimetty nimellä ”Oulujoki”.

Kuva liitetään ohjelmaan laittamalla Game1 luokkaan luokkatason muuttuja
`Texture2D aloituskuva;`

Luva ladataan peliin sisältöputkessa LoadContent() metodissa seuaavasti:
`aloituskuva = Content.Load<Texture2D>(@"Kuvat/Oulujoki");`

Näyttöön sen saa, kun lisää Draw() metodiin seuraavan lauseen `if (peliTilaNyt == peliTila.aloitus)` lausekken sisään:

```
spriteBatch.Draw(aloituskuva, Vector2.Zero, null, Color.White, 0,
Vector2.Zero, 1, SpriteEffects.None, 0);
```

Kuvan saa halutessaan eri paikkaan antamalla eri arvot toiselle parametrille. Oletus `Vector2.Zero` piirtää kuvan vasemmasta yläkulmasta alkaen, mutta siihen voi antaa parametreinä x ja y koordinantit näin: new Vector2(100,200) ,kuten tässä ohjelmassa teen.

10: Näppäimistösyötöt

Näppäimistösyötöissä tutkitaan, onko joku näppäin alhaalla, vai ylhäällä. Yleensä pelissä olennaista on, onko näppäintä painettu (alhaalla). Esim. onko nuolta painettu vasemmalle, voi lukea:

```
if (Keyboard.GetState().IsKeyDown(Keys.Left))
```

vastaavasti oikealle painallus luetaan:

```
if (Keyboard.GetState().IsKeyDown(Keys.Right))
```

Välityöntinäppäimen painallus luetaan:

```
if (Keyboard.GetState().IsKeyDown(Keys.Space))
```

Kirjainten painallukset ohjelmoidaan käyttämällä ”Keys” sanan jälkeen pistettä ja isoa kirjainta. XNA ei erottele näppäimistöltä luettaessa isoja ja pieniä kirjaimia, mutta ohejlmoitaessa kirjaimet on kirjoitettava isolla. Peli silti lukee myös pienet kirjaimet. Kun kirjoittaa ”Key” ja piste, antaa Visual Studio erilaisia vaihtoehtoja, joihin kannattaa tutustua.
Seuraava koodi tutkii ”S” kirjaimen painalluksen:

```
if (Keyboard.GetState().IsKeyDown(Keys.S))
```

ja seuraava F11 näppäimen painalluksen:

```
if (Keyboard.GetState().IsKeyDown(Keys.F11))
```

Numerojen painallukset voi lukea antamalla arvon `Key.D2` (lukee onko näppäin 2 painettu, D4 onko 4 jne.)

esc näppäimen painallus tarkistetaan seuraavasti:

```
if (Keyboard.GetState().IsKeyDown(Keys.Escape))
```

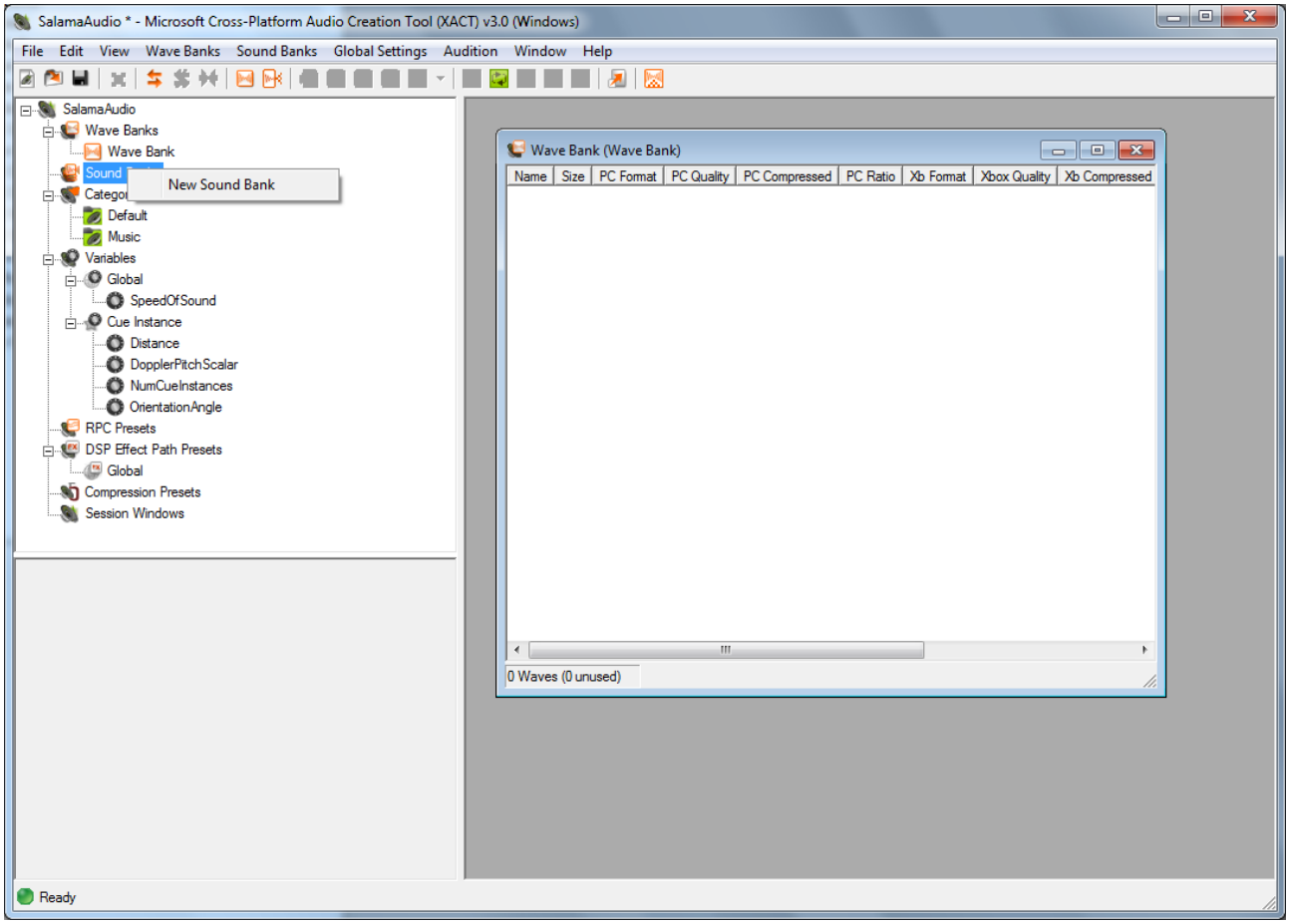
Mikäli halutaan tietää, onko joku näppäin ylhäällä, sen voi lukea kuten seuraavassa esimerkissä tutkitaan, onko välilyönti ylhäällä:

```
if (Keyboard.GetState().IsKeyUp(Keys.Space))
```

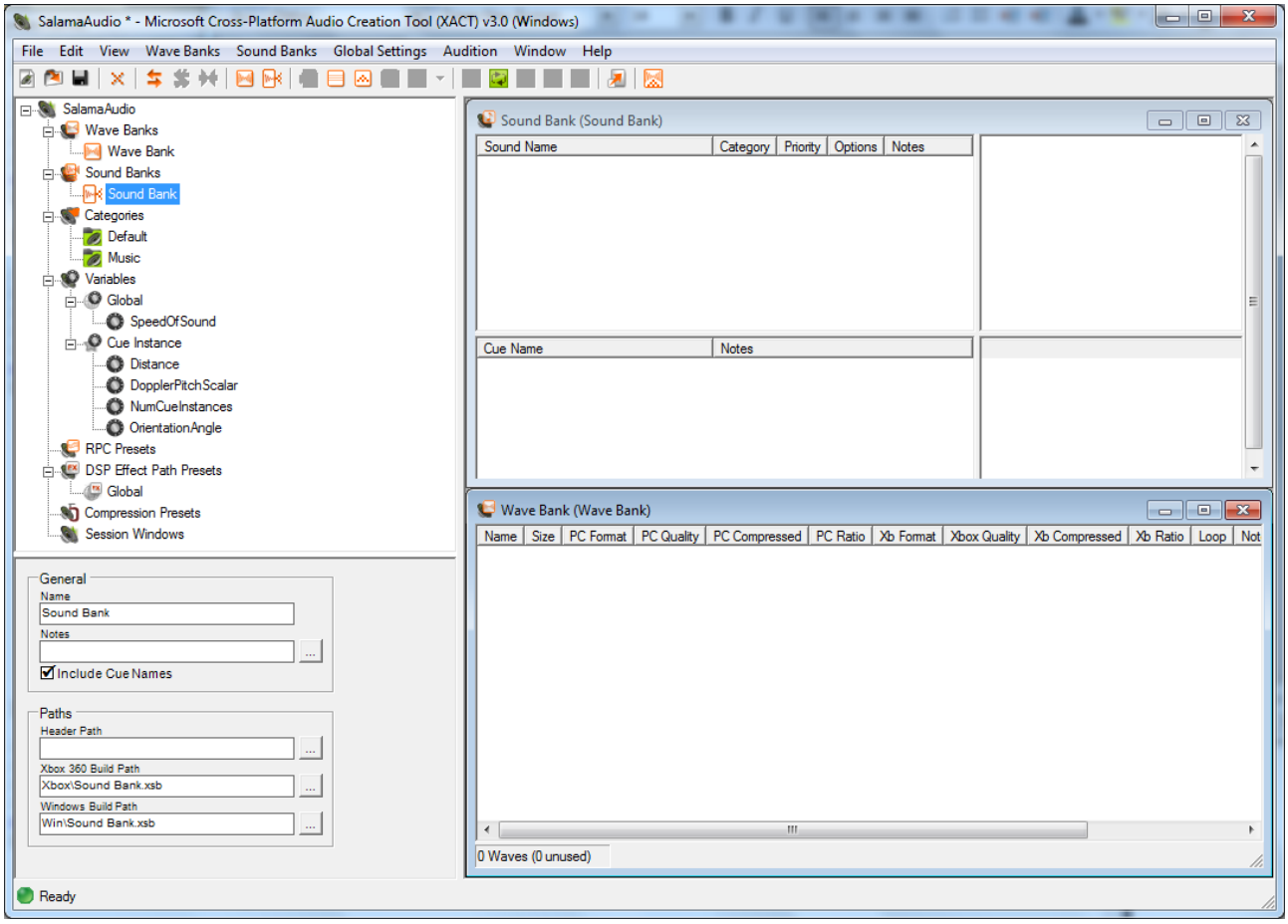
11: Musiikin ja äänitiedostojen lisääminen

Musiikki ja äänitehosteet lisätään XNA ohjelmaan XACT sisältöputken avulla, jos käytetään wav tiedostoja. XNA:ssa voi käyttää myös mp3 tiedostoja, jotka vievät n. kymmenen kertaa vähemmän tilaa, eli ovat suositeltavia eritoten verkossa julkaistavissa peleissä. wav on kuitenkin joissain asioissa, kuten äänitehosteissa kätevä, ja jos pelin julkaisee CD:llä, muutamalla kymmenellä ylimääräisellä megatavulla ei ole merkitystä.
XACT ohjelma on asentunut koneellesi XNA laajennuksen asennuksen yhteydessä. Pääset editoriin menemällä Windowsin ”Käynnistä” valikkoon, ja valitsemalla ”Kaikki ohjelmat/ Microsoft XNA Game Studio 4.0/ Tools/ Microsoft Cross-Platfom Creation Tool (XACT3).

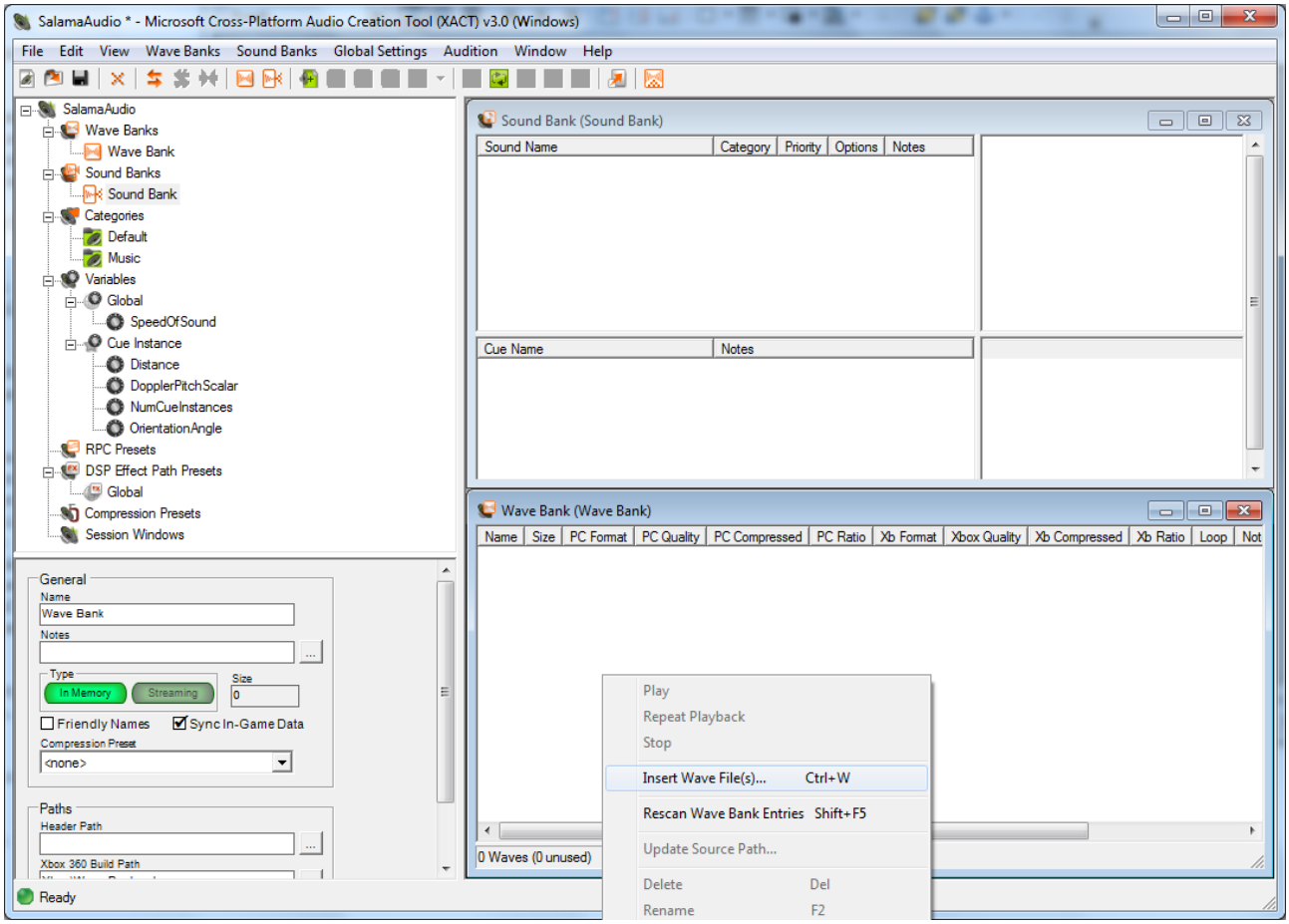
Sinulle aukeaa XACT editori, jossa valitse ”new project” ja anna sille nimeksi vaikkapa ”SalamaAudio”. Klikkaa hiiren oikealla vasemmasta valikosta ”Wave Banks” kohtaa ja valitse ”new wave bank” Tee sitten sama ”Sound Banks” Kodassa. Seuraava kuva selventää hieman.



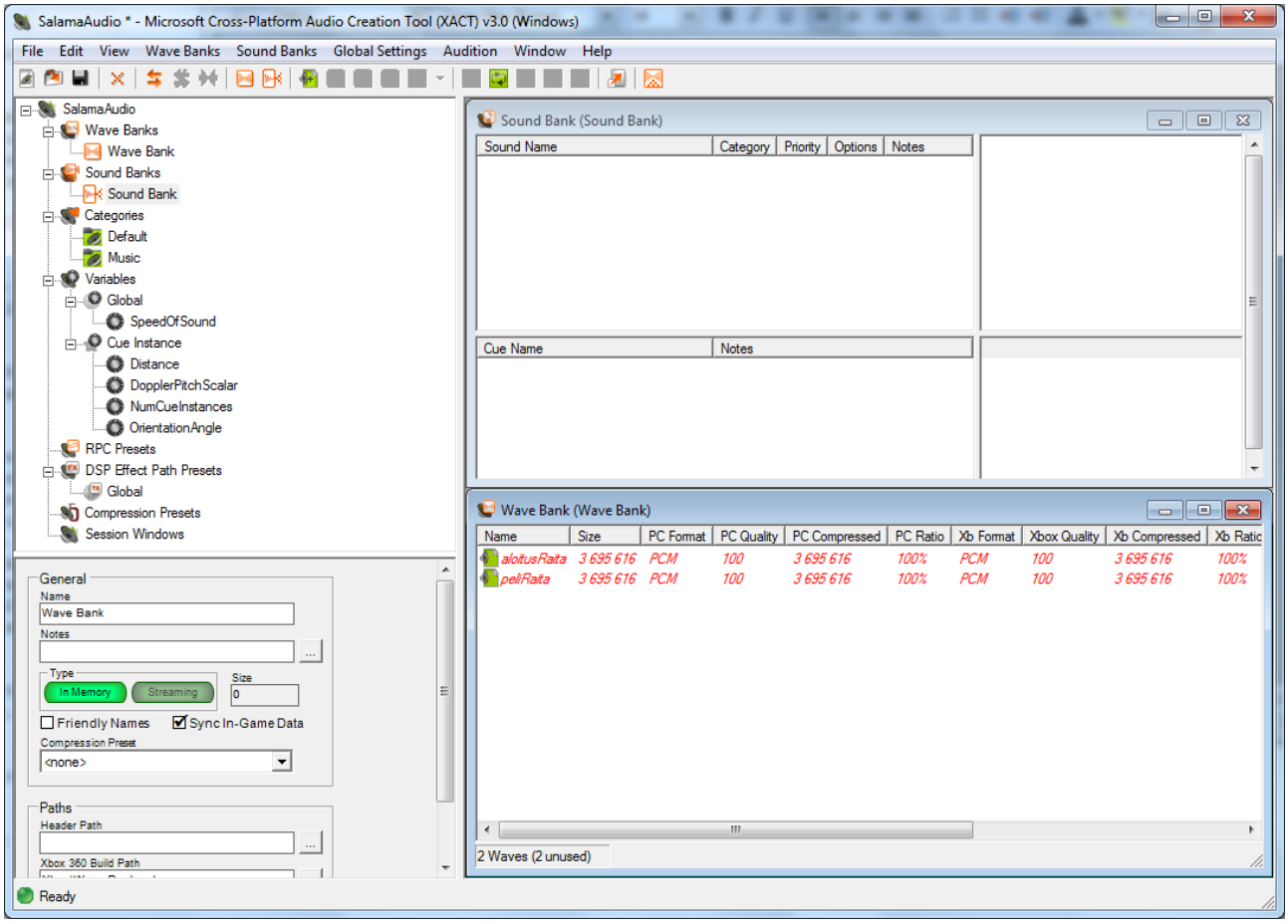
”Wave Bank” ja ”Soud Bank” ovat hankalasti näytöllä päällekäin, mutta ne on helppo järjestää ”Window” kohdasta ”Tile Horizontally”. Jolloin näkymä on seuraava:



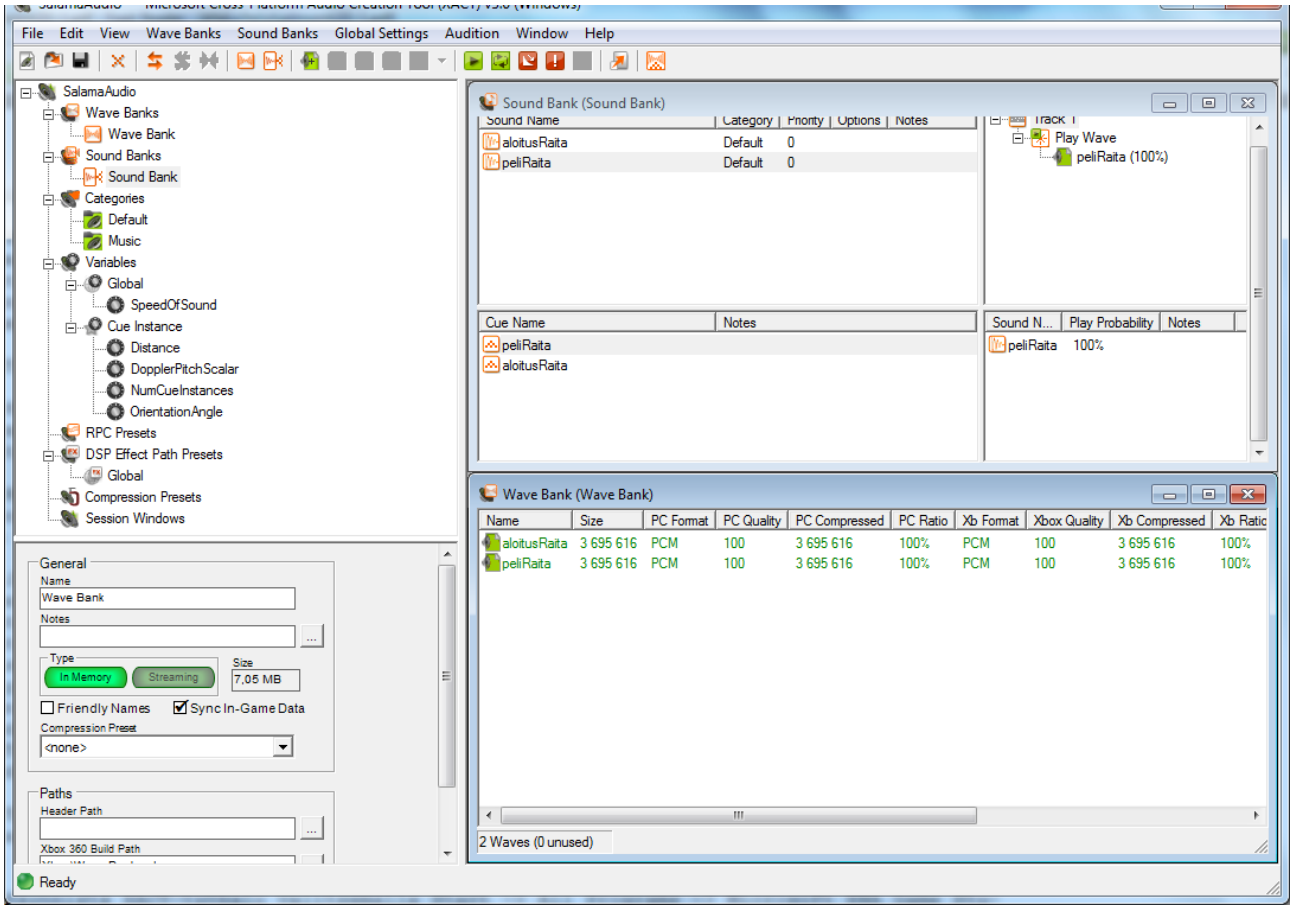
Mene nyt ”Wave Bank kohtaan oikealla ikkunassa, ja klikkaa hiiren oikealla, ja valitse ”Insert Wave Files” ja valitse sitten tiedostoistasi soitettava musiikki tai äänitehoste wave tiedostona. Huomioitavaa on, että XNA:ssa yleensä käytetään Wave tiedostoja, ja vaikka MP3 tiedostot periaatteessa toimivat, niiden kanssa voi tulla ongelmia. Waven huono puoli on, että jos jollain Conventterillä (löytyy netistä useita, kun Googletat MP3 to Wave) muutetaan MP3:sta Waviksi, tiedoston koko voi jopa kymmenkertaistua, ja näin ohjelman koko kasvaa. Wave formaatti on eniten tilaa vievä formaatti. Coventtereissä voi olla jotain ilmaisia kokeilumahdollisuuksia, mutta tyypillisesti ne kääntävät ilmaisversiona vain esim. 60% tiedostosta. Kovin kalliita ne eivät kuitenkaan ole. Joitain kymmeniä euroja ohjelmasta riippuen.
Alla oleva kuva näyttää tiedoston lisäämisen.



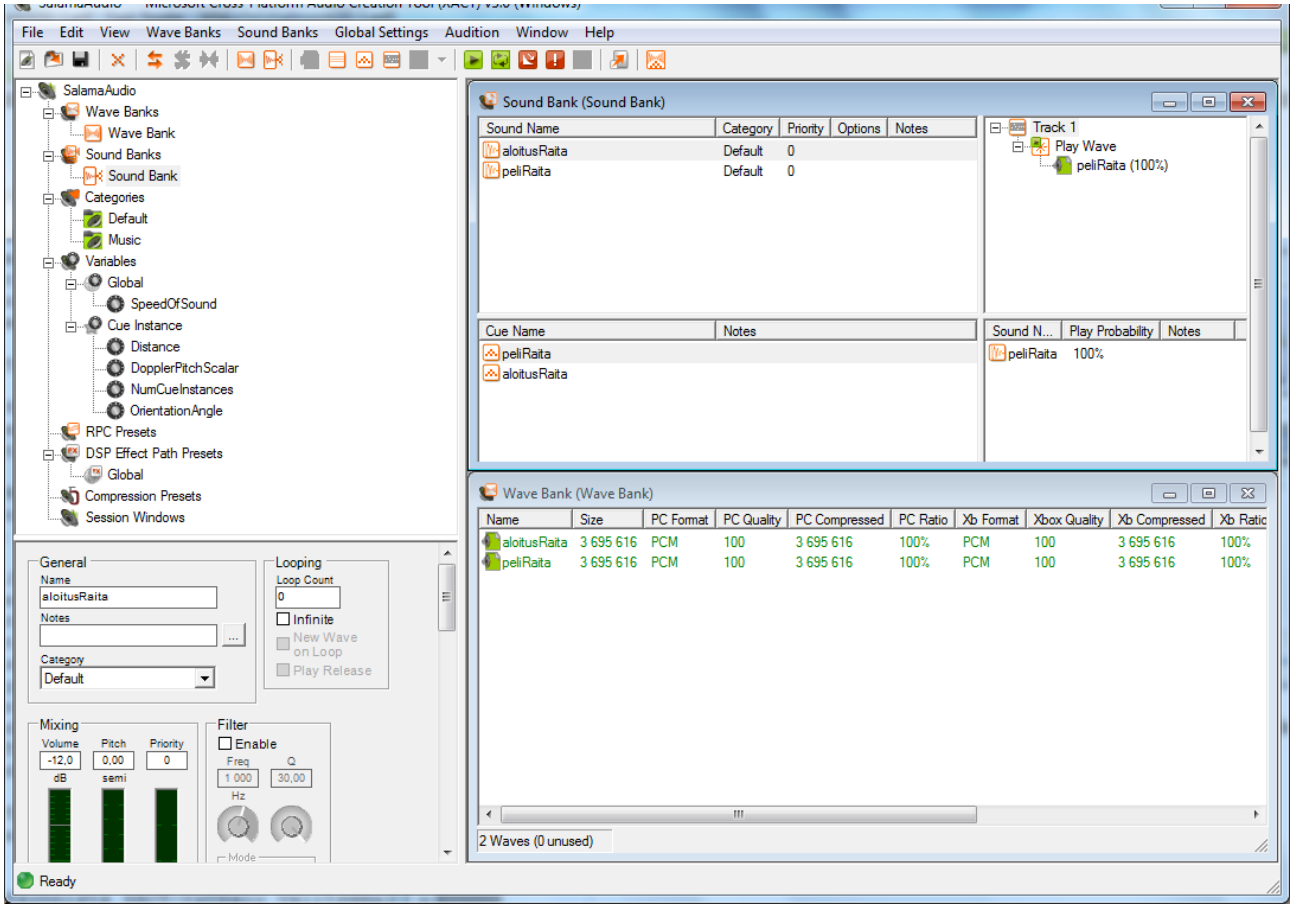
Jos laitoit kaksi raitaa, eli vaikka kaksi musiikkikappaletta, tai vaikkapa musiikkikappaleen ja äänitehosteen (törmäysääni jne.), Wave Bank näyttää nyt tältä:



Seuraavaksi siirrä kumpikin raita Sound Bankin alempaan osioon, jossa lukee ”Cue Name”. Tiedostot kopioituvat automaattisesti myös ”Sound Name” osioon. Sinulla pitäisi olla nyt alla oleva näkymä:



Muokataan nyt hieman äänen vopimakkuutta. Klikkaa Soun Bankissa ylemmässä ikkunassa raidan nimeä. Vasemmalla alhaalla näkyy seuraava editori:



Jos valitset vasemmalta alhaalta ”Looping” kohdan alta ”Infinite” peli soi automaattisesti toistuvasti koko ajan takana, ellet sitten pelissä vaienna sitä. Myös äänen voimakkuutta voi halutessaan muuttaa. Oletuksena on -12 ja se on usein sopiva, joskin aika voimakas. Itse olen käyttänyt sitä aloitus raidassa, mutta pelin aikana pyörivään raitaan olen laittanut voimakkuudeksi -30. Tässä harjoituspelissä valitsemme myös ”Infinite” valinnan. Muitakin muokkausmahdollisuuksia on, ja niihin voi perehtyä, mutta näillä pärjää jo hyvin.
Äänitiedosto on nyt valmis tallennettavaksi.

Seuraavaksi lisätään äänitiedosto. Tee ”Content” kansioon ”Audio” kansio. Lataa siihen wave ääniraidat (hiiren oikealla Audio kansiota, ja Add/Existing Item), jotka laitoit myös XACT:ssa ja sitten tekemäsi ”SalamaAudio” tiedosto. XACT:ssa lisätyt ääniraidat (Cue) ovat tässä pelissä nimeltään aloitusRaita ja peliRaita.

Sitten ”Game1” luokan yläosaan on lisättävä luokkatason muuttujat:

```
AudioEngine audioMoottori;
WaveBank aaltoKentta;
SoundBank aaniKentta;
Cue aloitusRaita;
Cue peliRaita;
```

Ensimmäistä muuttujaa audioMoottori käytetään Wave Bank ja Sound Bank objektien luomisessa, ja se edustaa äänimoottoria. Näitä tarvitaan myöhemmin määrittelyissä. Cue-objektia käytetään soittamisessa ja soittamisen kontrolloinnista. Ilman Cue:takin voi ääniä XNA:ssa tuottaa, mutta silloin niitä ei voi ohjelmassa kontrolloida, kuten käyttää paussia.

Muuttujat alustetaan LoadContent metodissa, jonne laitetaan seuraava koodi:

```
audioMoottori = new AudioEngine(@"Content\Audio\SalamaAudio.xgs");
aaltoKentta = new WaveBank(audioMoottori, @"Content\Audio\Wave Bank.xwb");
aaniKentta = new SoundBank(audioMoottori, @"Content\Audio\Sound Bank.xsb");
```

Musiikkia soitetaan pelissä seuraavalla määritteellä:

```
peliRaita = aaniKentta.GetCue("peliRaita");
peliRaita.Play(); //aloittaa soittamisen
```

Toiminnolla peliRaita.Pause() musiikin voi laittaa paussiin ja peliRaita.Stop() pysäyttää. Usein myös tarve tutkia, onko raita paussissa sillä hetkellä, ja jos on käynnistetään uudelleen. Se tapahtuu seuraavasti:

```
if (peliRaita.IsPaused) peliRaita.Resume();
```

Taasen jos halutaan tietää, onko raita pysäytetty, ja käynnistää uudelleen, mikäli on pysäytetty, toimii seuraava koodi:

```
if (peliRaita.IsStopped)
{
 peliRaita = aaniKentta.GetCue("peliRaita");
 peliRaita.Play();
}
```

12: Musiikki mp3 tiedostoina

Edellä oli kerrottu, kuinka musiikki kannattaa tehdä, jos levittää peliä esim. CD:llä. Samoin wav tiedostot ovat käteviä esim. äänitehosteisiin. Netissä pelattavan pelin musiikiksi siinä on se ongelma, etä wav on eniten tilaa vievä tallennusmuoto. Minuutin ympärisoitettu sävellys, joka mp3 tiedostona on n. 1MB, voi olla wav tiedostona liitettynä yli 10MB. Sellainen syö paljon siirtokaistaa, mikä on verkkopalveluissa kalleinta.
XNA osaa soittaa myös mp3 tiedostoja. Siinä on vain rajoiteensa. Olen saanut toimimaan vain yhden ympärisoitetun sävelmän per peli. En siis erikseen aloitus- ja peliajan musiikkia. Sävelmä nimittäin on käynnistettävä LoadContent() metodissa. Sävelmä liitettään peliin seuraavasti:

Lataa Audio kansioon musiikki, joka tässä on tiedostonimellä ”Raita”.

Lisää luokkatason muuttuja `Song musiikki;`

Sitten LoadContent() metodiin:

```
musiikki = Content.Load<Song>(@"Audio/Raita");
MediaPlayer.IsRepeating = true; //Sallii ympäritoiston
MediaPlayer.Play(musiikki);
```

Voit katkaista musiikin pelisilmukassa: `MediaPlayer.Stop();`
Tai laittaa tauolle `MediaPlayer.Pause();`

Ja käynnistää uudelleen: `MediaPlayer.Resume();`

Tuota `MediaPlayer.Play(musiikki);` käskyä en yrityksistä huolimatta ole saanut toimimaan muualla kuin LoadContent() metodissa.

Lisätietoa mp3 musiikin soittamisesta XNA:lla on:

http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.media.mediaplayer.aspx

13: Aluksen liikuttaminen

Aluksen piirtämistä ja liikuttelua varten määrittele ensin Game1 luokkatason muuttujat:

```
Texture2D alus;
Vector2 aluksenPaikka = new Vector2();
double alusX = 550, alusY = 350;
double suuntaX, suuntaY;
double nopeus=1, nop;
```

aluksenPaikka on muuttuja, jolla lopulta määritellään paikka, mihin alusta liikutetaan. Siihen viitataan aluksenPaikka.X ja aluksenPaikka.Y. AlusX ja AlusY koordinanteille annetaan alkuarvot. Niitä käytetään liikuttamisen laskemisessa. SuuntaX ja suuntaY muuttujia käytetään apuna määrittelemään liikutettava suunta. Nopeus muuttujalle annetaan alkuarvo, jota voi pelin aikana muuttaa. Nop on apuna nopeuden laskemisessa.

Alus piirretään Draw metodissa ”pelaa” pelitilassa seuraavasti:

```
aluksenPaikka.X = (int)alusX;
aluksenPaikka.Y = (int)alusY;
spriteBatch.Draw(alus, aluksenPaikka, null, Color.White, 0, Vector2.Zero,
1, SpriteEffects.None, 0);
```

Sitä ennen kuva on pitänyt ladata ”Kuvat” kansioon Content solmussa ja ladata sisältöputken kautta ohjelmaan LoadContent() metodissa seuraavasti:

```
alus = Content.Load<Texture2D>(@"Kuvat/Kori");
```

`spriteBatch` käskystä mainittakoon, että viimeinen parametri on kerrossyvyys, jolla voi määritellä, mitkä kuvat näkyvät päälimmäisenä.

Kori on tässä aluksen kuvan nimi.

```
//Metodi, joka ohjaa aluksen liikuttelun näppäintoimintoja
public void AluksenSiirto()
{
 //Tarkistetaan, mihin suuntaan mahdollisesti nuolta on painettu
 //ja valitaan sen mukainen suunta
 if (Keyboard.GetState().IsKeyDown(Keys.Right))
 {
  suuntaX = 6;
  suuntaY = 0;
 }
 if (Keyboard.GetState().IsKeyDown(Keys.Left))
 {
  suuntaY = 0;
  suuntaX = -6;
 }
 if (Keyboard.GetState().IsKeyDown(Keys.Up))
 {
   suuntaY = -6;
```

```
    suuntaX = 0;
   }
   if (Keyboard.GetState().IsKeyDown(Keys.Down))
   {
    suuntaY = 6;
    suuntaX = 0;
   }
   if (Keyboard.GetState().IsKeyDown(Keys.Right)
     && Keyboard.GetState().IsKeyDown(Keys.Up))
   {
    suuntaX = 6;
    suuntaY = -6;
   }
   if (Keyboard.GetState().IsKeyDown(Keys.Right)
     && Keyboard.GetState().IsKeyDown(Keys.Down))
   {
    suuntaY = 6;
    suuntaX = 6;
   }
   if (Keyboard.GetState().IsKeyDown(Keys.Left)
     && Keyboard.GetState().IsKeyDown(Keys.Up))
   {
    suuntaX = -6;
    suuntaY = -6;
   }
   if (Keyboard.GetState().IsKeyDown(Keys.Left)
     && Keyboard.GetState().IsKeyDown(Keys.Down))
   {
    suuntaX = -6;
    suuntaY = 6;
   }

//Tarkistetaan että alus ei liiku näytön ulkopuolelle
//ja sallitaan liikuttaminen muihin suuntiin kuin näytön ulkopuolelle
if (aluksenPaikka.X < 0 && Keyboard.GetState().IsKeyDown(Keys.Left))
 nop = 0;
 else if (aluksenPaikka.X > 1080 &&
  Keyboard.GetState().IsKeyDown(Keys.Right))
  nop = 0;
 else if (aluksenPaikka.Y < 0 && Keyboard.GetState().IsKeyDown(Keys.Up))
  nop = 0;
 else if (aluksenPaikka.Y > 500 &&
  Keyboard.GetState().IsKeyDown(Keys.Down))
  nop = 0;
 else
  nop = nopeus;

//Kasvatetaan koordinantteja, mikäli jotain nuolinäppäintä on painettu
if (Keyboard.GetState().IsKeyDown(Keys.Right)
 || Keyboard.GetState().IsKeyDown(Keys.Left)
  || Keyboard.GetState().IsKeyDown(Keys.Up)
```

```
  || Keyboard.GetState().IsKeyDown(Keys.Down))
 {
  alusX = alusX + suuntaX * nop;
  alusY = alusY + suuntaY * nop;
 }
}
```

14: Salaman liikuttelu

Salaman lisäämiseen ja siitelyyn tarvitset seuraavat luokka tason muuttujat:

```
Texture2D salamaKuva;
Vector2 salamaPaikka = new Vector2();
double salamaY;
bool salama;
```

Laita sitten tekemästäsi ”Kuvat” kansiosta LoadContent() sisältöputkeen seuraava lause:

```
salamaKuva = Content.Load<Texture2D>(@"Kuvat/Salama");
```

Salaman liikutteluun tarvitset seuraavaan metodin:

```
public void Salama()
{
 //Jos salamaa ei ole näytöllä, arvotaan uusi salaman X koordinantti
 if (salama == false)
 {
  salamaPaikka.X = satLuku.Next(20, 1079);
  salama = true;
 }

 if (salamaPaikka.Y > 500)
 {
  salama = false;
  salamaY = 0;
  elamat--;
  laskuri = 0;
  peliTilaNyt = peliTila.virhe;
 }

 salamaY += 1.5 * nopeus;
}
```

Update() metodiin tarvitset vain seuraavan lauseen tämän PeliTilaNyt switch lausekkeen `case peliTila.pelaa kohtaan:`

```
//Salaman liikutus
```

```
Salama();
```

Ja Draw() metodiin:

```
//Piirretään salama
salamaPaikka.Y = (int)salamaY;
spriteBatch.Draw(salamaKuva, salamaPaikka, null, Color.White, 0,
Vector2.Zero, 1, SpriteEffects.None, 0);
```

Muistathan, että satunnaisluvun C#:ssa saa `Random satLuku = new Random()` luokan avulla, ja siitä luotua oliota satLuku voi käyttää:
`int satLuku = satLuku.Next(0, 4)`, joka palauttaa jonkun luvuista 0, 1, 2, 3 (ei siis enään neljää).

15: Tormäysten tarkkailu

Lisätään luokkatason muuttuja:

```
bool salama;
```

Seuraavaksi metodi:

```
  //Tarkistetaan törmääkö alus ja salama
        public bool SalamaTormaus()
        {
            Rectangle alusSuorakaide = new
Rectangle((int)aluksenPaikka.X, (int)aluksenPaikka.Y, 40, 40);
            Rectangle salamaSuorakaide = new
Rectangle((int)salamaPaikka.X, (int)salamaPaikka.Y, 20, 40);

            return alusSuorakaide.Intersects(salamaSuorakaide);
        }
```

Update() metodiin `peliTila.pelaa` kohtaan lisätään rivit:

```
 //Tarkistetaan törmäys salamaan
if (SalamaTormaus())
{
 pisteet += 100;
 salama = false;
 salamaY = 0;
}
```

16: Koodivinkki aluksen liikutteluun

Tämä vinkki ei liity nyt tehtävään peliin, mutta voi olla hyödyllinen esim. vene ja autopeleissä. Aina ei haluta, että vene tai auto liikkuu tarkalleen näppäimien mukaan, vaan sillä on suunta ja nopeus, johon se vielä liukuu, vaikka näppäintä toisaalle painettaisiinkin. Tässä ohjelmassa on suuntaViive, jota muuttamalla alukselle saa erilaisia liukuja.

```
//Metodi, joka ohjaa aluksen liikuttelun näppäintoimintoja
        public void AluksenSiirto()
        {
            if (Keyboard.GetState().IsKeyDown(Keys.Right))
            {
                suuntaX = 8;
                suuntaY = 0;
                suuntaViiveX += 0.01;
            }
            if (Keyboard.GetState().IsKeyDown(Keys.Left))
            {
                suuntaY = 0;
                suuntaX = -8;
                suuntaViiveY -= 0.01;
            }
            if (Keyboard.GetState().IsKeyDown(Keys.Up))
            {
                suuntaY = -8;
                suuntaX = 0;
                suuntaViiveY -= 0.01;
            }

            if (Keyboard.GetState().IsKeyDown(Keys.Down))
            {
                suuntaY = 8;
                suuntaX = 0;
                suuntaViiveY += 0.01;
            }
            if (Keyboard.GetState().IsKeyDown(Keys.Right) &&
Keyboard.GetState().IsKeyDown(Keys.Up))
            {
                suuntaX = 8;
                suuntaY = -8;
                suuntaViiveX += 0.01;
                suuntaViiveY -= 0.01;
            }




            if (Keyboard.GetState().IsKeyDown(Keys.Right) &&
```

```
Keyboard.GetState().IsKeyDown(Keys.Down))
            {
                suuntaY = 8;
                suuntaX = 8;
                suuntaViiveY += 0.01;
                suuntaViiveY += 0.01;
            }

            if (Keyboard.GetState().IsKeyDown(Keys.Left) &&
Keyboard.GetState().IsKeyDown(Keys.Up))
            {
                suuntaX = -8;
                suuntaY = -8;
                suuntaViiveX -= 0.01;
                suuntaViiveY -= 0.01;
            }
            if (Keyboard.GetState().IsKeyDown(Keys.Left) &&
Keyboard.GetState().IsKeyDown(Keys.Down))
            {
                suuntaX = -8;
                suuntaY = 8;
                suuntaViiveX -= 0.01;
                suuntaViiveY += 0.01;
            }
            if (suuntaViiveX >= 0.6) suuntaViiveX = 0.6;
            if (suuntaViiveY >= 0.6) suuntaViiveY = 0.6;
            if (suuntaViiveX <= -0.6) suuntaViiveX = -0.6;
            if (suuntaViiveY <= -0.6) suuntaViiveY = -0.6;

                alusX = alusX + suuntaX * nopeus + suuntaViiveX;
                alusY = alusY + suuntaY * nopeus + suuntaViiveY;

        }
```

17: Hieman tekoälyä

Tekoäly eli englanniksi Artifical Intelligence (AI) on olennainen osa peliohjelmointia. Erilaisia näytöllä olevia spritejä liikutetaan erilaisten tekoälyjen avulla. Samoin pelin muuhun koodiin voi sisältyä monenkinlaista tekoälyä, kuten shakkitietokoneeseen ohjelma, joka päättelee tietokoneen seuraavan siirron.

Lisätään luokkatason muuttujat rakeiden liikuttamiseen:

```
Vector2[] raePaikka = new Vector2[4];

double[] raeX = new double[4];
double[] raeY = new double[4];
double[] k = new double[4];

bool[] rae = new bool[4];
```

Lisätään seuraava metodi:

```
//Rakeiden luonti ja liikutus
public void Rae()
{
 for (int i = 0; i < 1 + taso; i++)
 {
  //Jos raetta ei ole, luodaan
  if (rae[i] == false)
  {
   raeX[i] = satLuku.Next(20, 1079);
   rae[i] = true;

  //Lasketaan rakeelle kulmakerroin niin, että se suuntautuu alusta kohti
   k[i] = (aluksenPaikka.X - raeX[i]) / aluksenPaikka.Y;

   //Rajoitetaan että kulmakerroin on välillä -2  < k < 2
   if (k[i] > 2) k[i] = 2;
   if (k[i] < -2) k[i] = -2;
  }

  //Jos rae on saavuttanut alarajan, tai siirtynyt X suunnassa pois
näytöltä
   if (raePaikka[i].Y > 500 || raePaikka[i].X < 0
     || raePaikka[i].X > 1100)
   {
    rae[i] = false;
    raeY[i] = 0;
   }

   //Lasketaan X
   raeX[i] += 1 * nopeus * k[i];

 //Lasketaan Y
```

```
   raeY[i] += 1 * nopeus;
  }
}
```

Tarkistetaan aluksen törmäykset rakeisiin:

```
//Tarkistetaan törmätäänkö rakeisiin
public bool RaeTormaus(int i)
{
 Rectangle alusSuorakaide = new Rectangle((int)aluksenPaikka.X,
  (int)aluksenPaikka.Y, 40, 40);
 Rectangle raeSuorakaide = new Rectangle((int)raePaikka[i].X,
  (int)raePaikka[i].Y, 20, 20);

 return alusSuorakaide.Intersects(raeSuorakaide);
}
```

Seuraavaksi Update() metodiin seuraava koodi

```
//Tarkistetaan törmäykset rakeisiin
for (int i = 0; i < 4; i++)
{
 if (RaeTormaus(i))
 {
  for (int j = 0; j < 4; j++)
  {
   rae[j] = false;
   raeY[i] = 0;
  }
   elamat--;
   laskuri = 0;
   peliTilaNyt = peliTila.virhe;
   break;
 }
}
```

Rakeet piirretään Draw() metodissa seuraavasti:

```
//Piirretään rakeet
for (int i = 0; i < 1 + taso; i++)
{
 raePaikka[i].X = (int)raeX[i];
 raePaikka[i].Y = (int)raeY[i];
 spriteBatch.Draw(raeKuva, raePaikka[i], null, Color.White, 0,
Vector2.Zero, 1, SpriteEffects.None, 0);
}
```

Rakeita liikutetaan Update() metodissa yksinkertaisesti:

```
//Rakeiden liikutus
Rae();
```

Lokit lisätään seuraavasti.

Ensin luokkatason muuttujat:

```
Texture2D lokkiKuva;
Vector2[] lokkiPaikka = new Vector2[6];

Point kehysKokoLokki = new Point(40, 20);
Point nykyinenKehysLokki = new Point(0, 0);
Point arkkiKokoLokki = new Point(3, 1);

double[] k = new double[6];
double[] lokkiX = new double[6];
double[] lokkiY = new double[6];
double[] lokkiSuuntaX = new double[6];
double[] lokkiSuuntaY = new double[6];


lokkiKuva = Content.Load<Texture2D>(@"Kuvat/Lokit");
```

”Lokit” kuvatiedoston takana on spritearkki. Samaan kuvaan on laitettu tarkalleen saman kokoisia kuvia, tässä 40*20 pixeliä. XNA:n spriteBach käskyllä on helppo animoida tällainen arkki. Kätevä keino tällaisten piirtämiseen on piirtää kuva kuvankäsittelyohjelmalla (esim. ilmaisella Gimpillä) kymmenen kertaa suuremmaksi. Näin saa tarkemman grafiikan. Sitten vain pienentää kuvaa lopulliseen versioon kymmenen kertaa pienemmäksi.

Seraava kehyksien vaihtamiskoodi on otettu lähes suoraan oppaasta ”Learning XNA 4.0” (Aaron Reed, O'Reilly 2010). Jos osaa englantia, oppaaseen kannattaa ilman muuta tutustua. Kyseisen teoksen XNA olio-ohjelmointitekniikka on asia, mihin en tässä oppaassa perehdy, mutta se on taitavaa.

```
aikaViimeKehyksesta += gameTime.ElapsedGameTime.Milliseconds;
if (aikaViimeKehyksesta > millisekunnitKehystaKohti)
{
 aikaViimeKehyksesta -= millisekunnitKehystaKohti;
 ++nykyinenKehysLokki.X;
```

```
 if (nykyinenKehysLokki.X >= arkkiKokoLokki.X)
 {
  nykyinenKehysLokki.X = 0;
  ++nykyinenKehysLokki.Y;
  if (nykyinenKehysLokki.Y >= arkkiKokoLokki.Y)
   nykyinenKehysLokki.Y = 0;
 }
}
```

Lokien luontiin ja liikutteluun tehdään metodi:

```
//Lokkien luonti ja liikutus
public void Lokit()
{
 //Tässä [i] indeksillä osoitetaan lokin numeroa, joka voi olla 2-4
 int rnd;
 for (int i = 0; i < 1 + taso; i++)
 {
  //Jos alku luodaan lokki
  if (alku[i] == true)
  {
   tormays[i] = false;

   lokkiY[i] = 20;
   lokkiX[i] = satLuku.Next(50, 1000);
   lokkiSuuntaY[i] = 1;

   rnd = satLuku.Next(1, 3);
   switch (rnd)
   {
    case 1: lokkiSuuntaX[i] = 1;
     lokkiY[i] = 1;
     break;
    case 2: lokkiSuuntaX[i] = -1;
     lokkiSuuntaY[i] = 1;
     break;
   }

   alku[i] = false;
  }
 if (lokkiPaikka[i].X < 1)
 {
  lokkiSuuntaX[i] *= -1;
  rnd = satLuku.Next(0, 2);
  if (rnd == 0)
  {
   lokkiSuuntaY[i] *= -1;
   lokkiX[i] += 10;
  }
 }
 if (lokkiPaikka[i].X > 1045)
```

```
 {
   lokkiSuuntaX[i] *= -1;
   rnd = satLuku.Next(0, 2);
   if (rnd == 0)
   {
    lokkiSuuntaY[i] *= -1;
    lokkiX[i] -= 10;
   }
  }
   if (lokkiPaikka[i].Y < 1)
   {
    lokkiSuuntaY[i] *= -1;
    rnd = satLuku.Next(0, 2);
    if (rnd == 1)
    {
     lokkiSuuntaX[i] *= -1;
     lokkiX[i] += 10;
    }
  }
  if (lokkiPaikka[i].Y > 500)
  {
   lokkiSuuntaY[i] *= -1;
   rnd = satLuku.Next(0, 2);
   if (rnd == 1)
   {
    lokkiSuuntaX[i] *= -1;
    lokkiX[i] -= 10;
   }
  }

  lokkiX[i] += lokkiSuuntaX[i] * nopeus;
  lokkiY[i] += lokkiSuuntaY[i] * nopeus;
 }
}
```

Lokit piirretään Draw() metodissa seuraavasti:

```
//Piirretään lokit
for (int i = 0; i < 1 + taso; i++)
{
 lokkiPaikka[i].X = (int)lokkiX[i];
 lokkiPaikka[i].Y = (int)lokkiY[i];

 spriteBatch.Draw(lokkiKuva, lokkiPaikka[i], new
Rectangle(nykyinenKehysLokki.X * kehysKokoLokki.X,
 nykyinenKehysLokki.Y * kehysKokoLokki.Y, kehysKokoLokki.X,
 kehysKokoLokki.Y), Color.White, 0, Vector2.Zero, 1,
 SpriteEffects.None, 0);
}

//Tarkistetaan törmätäänkö lokkeihin
```

```
public bool LokkiTormaus(int i)
{
 Rectangle alusSuorakaide = new Rectangle((int)aluksenPaikka.X,
  (int)aluksenPaikka.Y, 40, 40);
 Rectangle lokkiSuorakaide = new Rectangle((int)lokkiPaikka[i].X,
  (int)lokkiPaikka[i].Y, 40, 20);

 return alusSuorakaide.Intersects(lokkiSuorakaide);
}
```

Update() metodiin lisätään seuraava koodipätkä:

```
//Tarkistetaan törmäykset lokkeihin
for (int i = 0; i < 1 + taso; i++)
{
 if (LokkiTormaus(i))
 {
  for (int j = 0; j < 1 + taso; j++)
  {
   alku[j] = true;
   Lokit();
  }
  elamat--;
  laskuri = 0;
  peliTilaNyt = peliTila.virhe;
  break;
 }
}
```

18: Ammuksen luonti ja liikuttaminen

```
//Jos painetaan välilyöntinäppäin alas, asetetaan laukaus arvoon ”true”
if (Keyboard.GetState().IsKeyDown(Keys.Space) && ammusMaara > 0)
                        laukaus = true;

//Kun vapautetaan välilyöntinäppäin ylös, ammutaan
if (Keyboard.GetState().IsKeyUp(Keys.Space) && laukaus == true)
{
 apuLaukaus = true;
 ammusAlku = true;
}
```

Itse ammuksen elinkaaren tarkkailu ja liikuttaminen on laitettu Draw() metodiin seuraavasti:

```
if (apuLaukaus == true)
{
 if (ammusAlku == true)
 {
  ammusX = aluksenPaikka.X;
  ammusY = aluksenPaikka.Y;
  ammusMaara--;
  ammusAlku = false;
  laukaus = false;
  ammusSuunta2X = ammusSuuntaX;
  ammusSuunta2Y = ammusSuuntaY;
 }
 ammusX += ammusSuunta2X;
 ammusY += ammusSuunta2Y;

 ammusPaikka.X = (int)ammusX;
 ammusPaikka.Y = (int)ammusY;

 if (ammusPaikka.X > 1100 || ammusPaikka.X < 0 || ammusPaikka.Y > 550
  || ammusPaikka.Y < 0)
 {
  apuLaukaus = false;
 }
 spriteBatch.Draw(ammus, ammusPaikka, null, Color.White, 0,
  Vector2.Zero, 1, SpriteEffects.None, 0);
}
```

`Update() metodissa on ”peliTila.pelaa”` tilassa ainoastaa kutsu ”Ammus()”, joka tarkkailee koko alan, mihin suuntaan nuolta on viimeksi painettu, ja antaa ammukselle sen jälkeen suunnan. Tulos piiretään ylläolevasti Draw() metodissa, jos ammus on näytöllä.

Ammuksen lentosuunnan valintametodi on seuraava:

```
//Ammuksen lentosuunnan valinta
public void Ammus()
{
 if (Keyboard.GetState().IsKeyDown(Keys.Right))
 {
  ammusSuuntaX = 10;
  ammusSuuntaY = 0;
 }
 if (Keyboard.GetState().IsKeyDown(Keys.Left))
 {
  ammusSuuntaY = 0;
  ammusSuuntaX = -10;
 }
 if (Keyboard.GetState().IsKeyDown(Keys.Up))
 {
  ammusSuuntaY = -10;
  ammusSuuntaX = 0;
 }
 if (Keyboard.GetState().IsKeyDown(Keys.Down))
 {
  ammusSuuntaY = 10;
  ammusSuuntaX = 0;
 }
 if (Keyboard.GetState().IsKeyDown(Keys.Right)
  && Keyboard.GetState().IsKeyDown(Keys.Up))
 {
  ammusSuuntaX = 10;
  ammusSuuntaY = -10;
 }
 if (Keyboard.GetState().IsKeyDown(Keys.Right)
  && Keyboard.GetState().IsKeyDown(Keys.Down))
 {
  ammusSuuntaY = 10;
  ammusSuuntaX = 10;
 }
 if (Keyboard.GetState().IsKeyDown(Keys.Left)
  && Keyboard.GetState().IsKeyDown(Keys.Up))
 {
  ammusSuuntaX = -10;
  ammusSuuntaY = -10;
 }
 if (Keyboard.GetState().IsKeyDown(Keys.Left)
  && Keyboard.GetState().IsKeyDown(Keys.Down))
 {
  ammusSuuntaX = -10;
  ammusSuuntaY = 10;
 }
}
```

Kun ammus halutaan pois näytöltä, se vain jätetään piirtämättä. Eli, kun olen ohjelmoinut, että ammusta liikutetaan ja se piirretään aina, kun ”laukaus==true” ja apuLaukaus == true, ammus poistetaan yksinkertaisesti kirjoittamalla

```
laukaus = false;
apuLaukaus = false;

//Tarkistetaan törmääkö ammus ja lokki
public bool AmmusTormaus(int i)
{
 Rectangle ammusSuorakaide = new Rectangle((int)ammusPaikka.X,
(int)ammusPaikka.Y, 10, 10);
 Rectangle lokkiSuorakaide = new Rectangle((int)lokkiPaikka[i].X,
(int)lokkiPaikka[i].Y, 40, 20);

 return ammusSuorakaide.Intersects(lokkiSuorakaide);
}
```

Lisää tekoälyä löydät oppaan lopussa julkaistusta lähdekoodista. Tutustu esim. Perhonen() ja Kotka() metodeihin. Perhonen arpoo liikkumissuunnan ja matkan minkä liikkuu, ja myös nopeuden. Kotka taasen liikkuu tietyn matkan, ja tarkistaa sitten suunnan alusta kolkee taasen jonkin matkaa, ja tarkistaa alusta kohti suunnan. Kotka kuitenkin liikkuu vain vasemmalta oikealle, eli kotkan vasemmalla puolella saa olla rauhassa.

19: Ennätystaulukot internettiin

XNA ennätystaulukot voidaan tehdä joko nettiin tai kovalevylle. Käsittelen tässä, miten ennätystaulukot tehdään nettiin. Jos haluat tallentaa tulokset kovalevylle, tutustu tähän englanninkieliseen verkko-oppaaseen
http://xnaessentials.com/tutorials/highscores.aspx

Tämä netti ennätystaulukkoluku pohjautuu http://brightside-games.com/?p=125&all=1 sivuihin, joihin kannattaa tutustua. Olen tässä tehnyt hieman yksinkertaisemman version. webPost metodi on otettu suoraan em. sivuilta copy pastella, ja vain hieman muokattu. Muu koodi on suurelta osin omaa.

Lisää ensin seuraavat using lauseet:

```
using System.Xml.Serialization;
using System.IO;
using System.Net;
using System.Text;
```

Lisää seuraava koodi (siis lähes suoraan em. sivuilta kopioituna):

```
 private string webPost(string _URI, string _postString)
 {
  const string REQUEST_METHOD_POST = "POST";
  const string CONTENT_TYPE = "application/x-www-form-urlencoded";
  Stream dataStream = null;
  StreamReader reader = null;
  WebResponse response = null;
  string responseString = null;
  // Create a request using a URL that can receive a post.
  WebRequest request = WebRequest.Create(_URI);
  // Set the Method property of the request to POST.
  request.Method = REQUEST_METHOD_POST;
  // Create POST data and convert it to a byte array.
  string postData = _postString;

  // Set the ContentType property of the WebRequest.
  request.ContentType = CONTENT_TYPE;
  // Set the ContentLength property of the WebRequest.

  byte[] byteArray = Encoding.UTF8.GetBytes(postData);
  request.ContentLength = byteArray.Length;

  try
  {
   // Get the request stream.
   dataStream = request.GetRequestStream();
   // Write the data to the request stream.
   dataStream.Write(byteArray, 0, byteArray.Length);
```

```
    // Close the Stream object.
    dataStream.Close();
    // Get the response.

    response = request.GetResponse();
    // Display the status.
    Console.WriteLine(((HttpWebResponse)response).StatusDescription);
    // Get the stream containing content returned by the server.
    dataStream = response.GetResponseStream();
    // Open the stream using a StreamReader for easy access.
    reader = new StreamReader(dataStream);
    // Read the content.
    responseString = reader.ReadToEnd();
    // Display the content.
    //Console.WriteLine(responseFromServer);

    // Clean up the streams.
    if (reader != null) reader.Close();
    if (dataStream != null) dataStream.Close();
    if (response != null) response.Close();
    }

    catch
    {
     yhtvirhe = true;
    }
     return responseString;
   }
```

Nettitaulukoissa olevat tiedot, jos haluat antaa parametrejä, tai lukea XNA ohjelmasta ennätystaulukoita, tehdään Xna ohjelmassa lisäämällä seuraava koodi:

```
public string getScores()
{
 try
 {
  return
webPost("http://www.sinunOsoitteesi.com/Kansio/requestscores.php",
postString);
 }
  catch
  {
   return "";
  }
}
```

requestscores.php koodi, joka lähettää kutsusta tiedot XNA:lle on esim.

seuravanlainen:

```
<?PHP
  $fl="Tiedostonimi.txt";
   $fp=fopen($fl, "r") or die ("Tiedostoa ei voida luoda");
   while(! feof($fp))
   {
    flock ($fp, 2);
   $ennatukset = fgets($fp, 2000);
      }
  flock($fp, 1);
  fclose($fp);
    print $ennatukset;
?>
```

Update() metodissa kutsuva koodi on yksinkertaisesti `postString = getScores();`

Nimi ja pistemäärä tulevat yhteen lauseeseen peräjälkeen. Jätä syötettäessa välilyonti tietojen perään, niin voit myöhemmin C# XNA:n Split() metodilla paloitella lauseen eri osiksi käytettäväksi. Voit lukea saatuja tietoja suoraan `postString` muuttujan avulla.

Koodi, jolla pisteet XNA ohjelmasta lähetetään PHP:lle palvelimelle on seuraava:

```
public string sendScore(string ennatukset)
{
 try
 {
  string postString = "&Hash=" + "SinunSalainenAvainkoodisi" +
"&Ennatukset=" + ennatukset;
  return webPost("http://www.sinunOsoitteesi.com/Kansio/newscore.php",
postString);

 }
 catch
 {
  return "";
 }
}
```

Sitä lukeva newscore.php ohjelma on seuraava:

```
<?PHP
if ($_POST["Hash"]=="SinunSalainenAvainkoodisi") {
$ennatukset=$_POST["Ennatukset"];
$fl="Tiedostonimi.txt";

  $fp=fopen($fl, "w") or die ("Tiedostoa ei voida luoda");
  flock ($fp, 2);
  fputs($fp, $ennatukset);
  flock($fp, 1);
  fclose($fp);
   }
?>
```

XNA koodia kutsutaan Update() metosissa antamalla parametsiksi nimi ja pistemäärä, eli esim. seuraavasti:

```
sendScore("Jaakko",4030);
```

Voit nyt tehdä PHP:llä seuraavan pienen testiohjelman, jolla voi testata, että XNA ohjelman internetistä ajamisen jälkeen tiedot ovat palvelimella.

```
<?php
$fn="Tiedostonimi.txt";
$fp=fopen($fn, "r") or die("Tiedostoa ei voinut avata");
$ennatykset=fgets($fp,20);
print "Nimi ja pisteet: $ennatukset";
flock($fp, 1);
fclose($fp);
?>
```

Olen lisännyt nimen kirjoittamisen yksinkertaisesti lukemalla painettuja näppäimiä, eli esim.

```
 if (Keyboard.GetState().IsKeyDown(Keys.A))
 {
 if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
     || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] =
'A';
  else nimi[pituus] = 'a';
  pituus++;
 }
 if (Keyboard.GetState().IsKeyDown(Keys.B))
 {
  if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
   || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'B';
```

```
   else nimi[pituus] = 'b';
   pituus++;
  }
 if (Keyboard.GetState().IsKeyDown(Keys.F3))
 {
  if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
      || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] =
'Ö';
      else nimi[pituus] = 'ö';
      pituus++;
 }
 if (Keyboard.GetState().IsKeyDown(Keys.D1))
 {
  nimi[pituus] = '1';
  pituus++;
 }
 if (Keyboard.GetState().IsKeyDown(Keys.D2))
 {
  nimi[pituus] = '2';
  pituus++;
 }
```

Nämä olen laittanut omaan metodiin. Sensijaan BS näppäimen panalluksen, jolla voi ”syödä” kirjaimia (takaisin) olen laittanut Update() metodiin seraavanlaisesti:

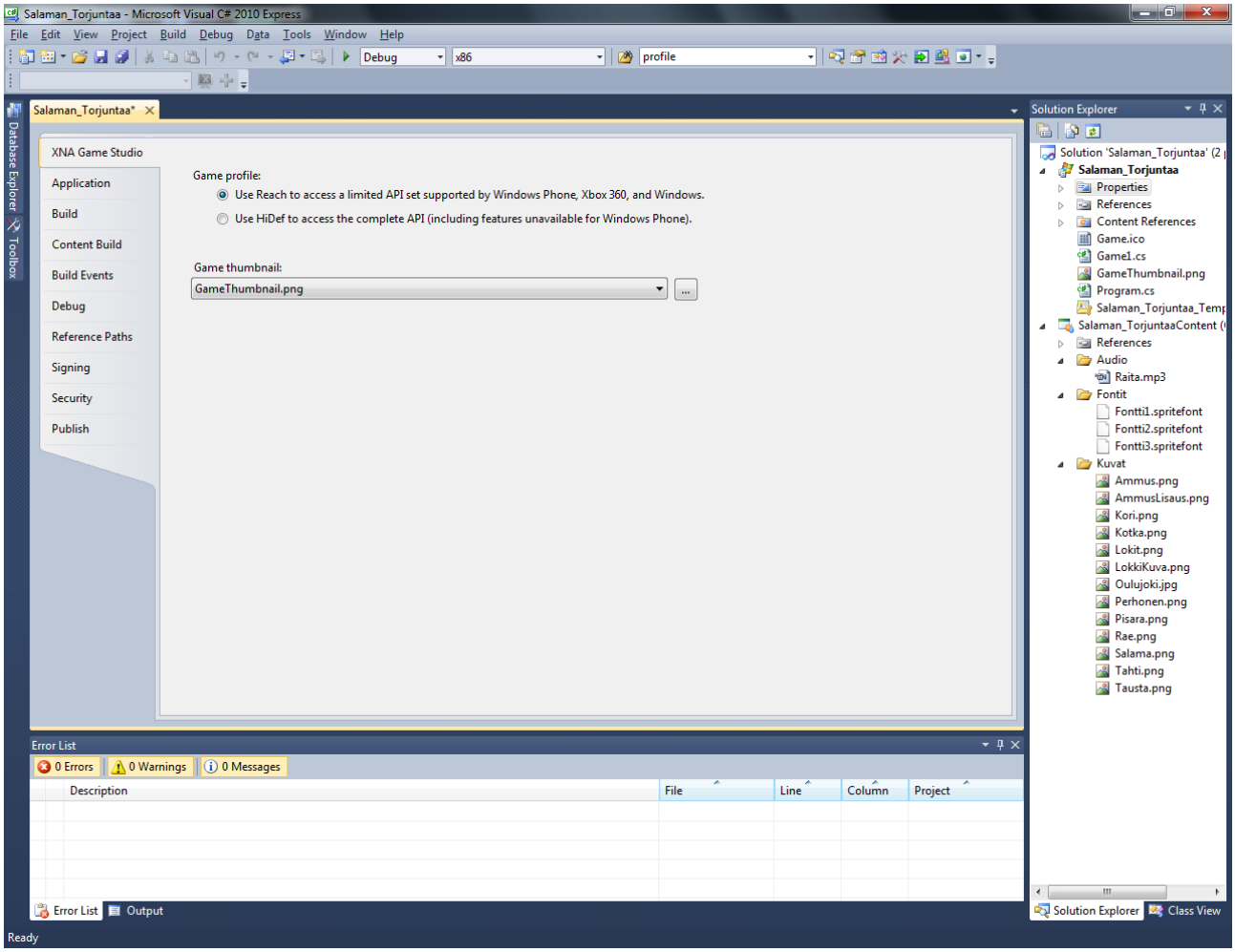
```
 if (Keyboard.GetState().IsKeyDown(Keys.Back) && aika % 10 == 0
  && pituus >= 1)
 {
  pituus--;
  nimi[pituus] = ' ';
  pelaajaNimi = Kirjoita();
 }
```

Löydät lopussa olevasta lähdekoodista tarkemmin nämä lisäykset. Jos haluat ne ohjelmaasi, ei muuta kuin copy/paste.

20: Pelin julkaisu

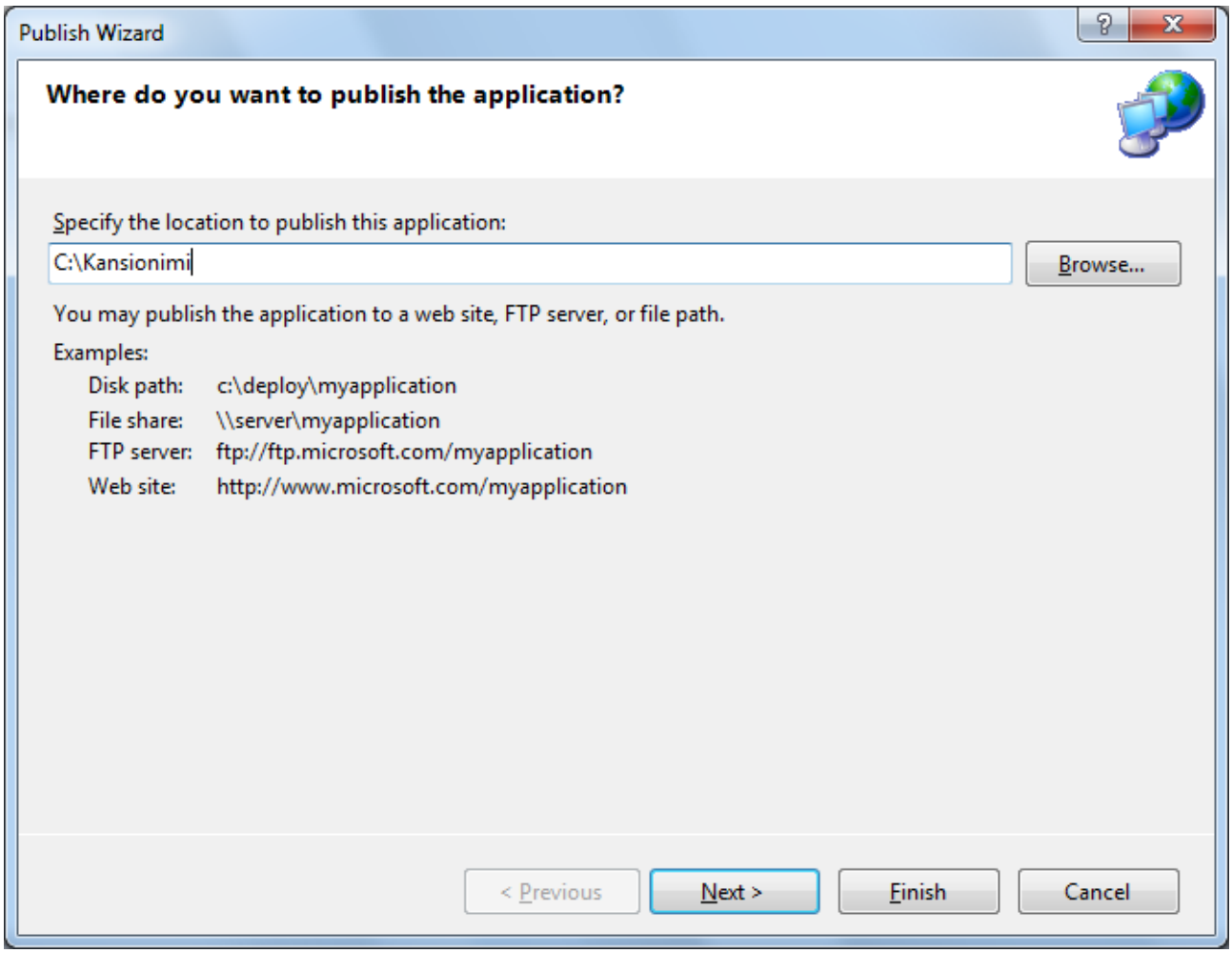
Pelin julkaisu on helppoa. Kuitenkaan XNA 4.0 (jota tässä oppaassa käytetään) versiolla tehdyt ohjelmat eivät toimi vanhimpien XP koneiden grafiikkakorteilla. Erityisesti voi tulla ongelmia vanhojen kannettavien kanssa. Ylipäätäänkään pelit eivät toimi oletusasetuksena XP koneissa. XNA ohjelmointiympäristössä on kuitenkin asetus, jolla saa toimimaan itse käyttöjärjestelmän puolesta, mikäli koneen grafiikkakortti on tarpeeksi hyvä.

Asetus muutetaan seuraavasti: oikealla Solution Explorerissa on ”Properties” kohta, kun tuplaklikkaat sitä, saa seuraavan näkymän.

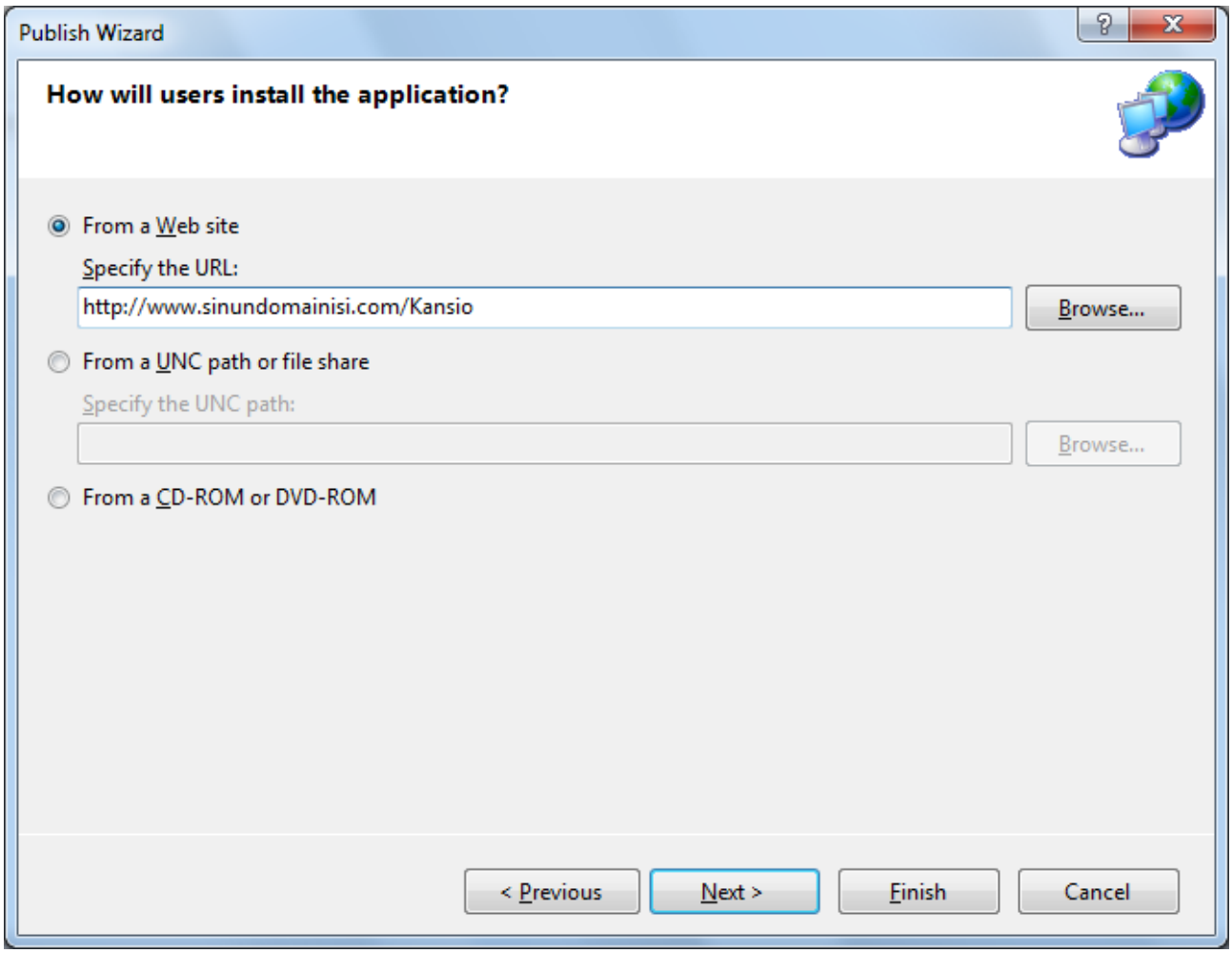


Valitse kahdesta vaihtoehdosta ylenpi, eli ”Reach” valinta. Oletusarvoisena on ”HiDef”, joka ei toimi vanhoissa koneissa.

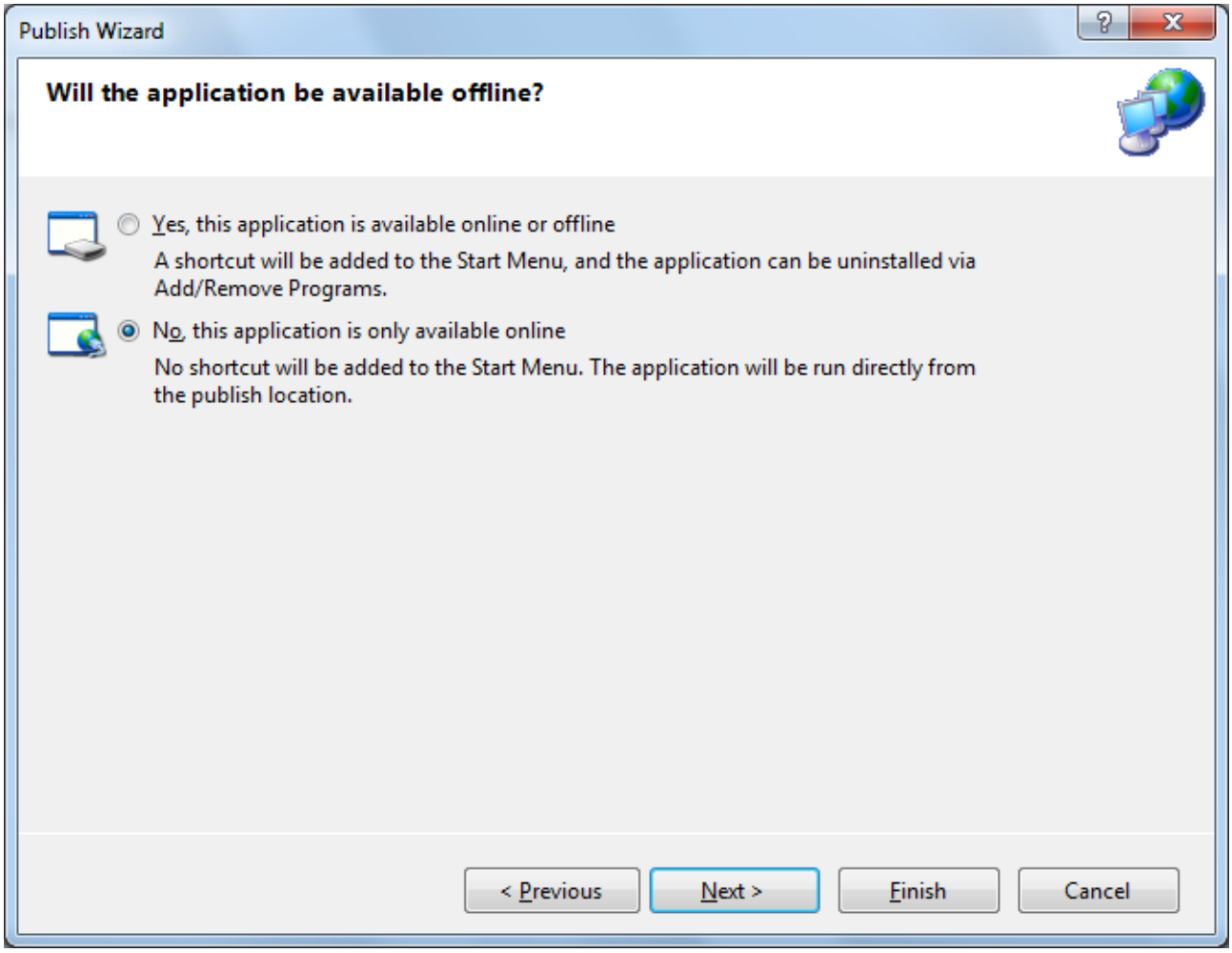
Itse julkaisu tapahtuu valitsemalla ”Build/ Publish” valinnat. Sieltä aukeaa seuraava ikkuna:



Kirjoita kansion nimi, mihin haluat pelin pakata, tai valitse kansio koneeltasi ”Browse” valinnalla. Klikkaa ”next”. Aukeaa seuraava ikkuna:

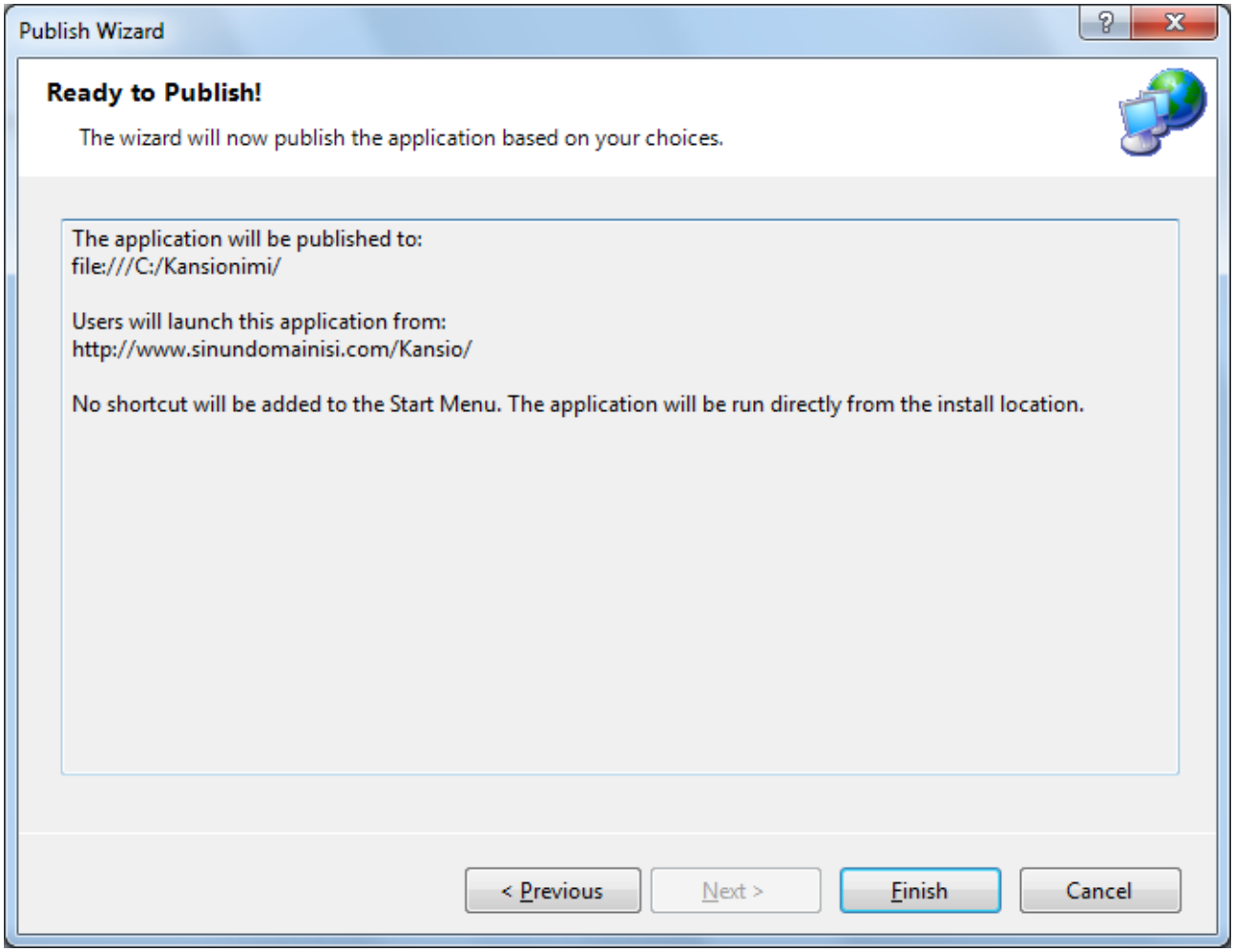

Publish Wizard
How will users install the application?
From a Web site
Specify the URL:
http://www.sinundomainisi.com/Kansio
Browse...
From a UNC path or file share
Specify the UNC path:
Browse...
From a CD-ROM or DVD-ROM
< Previous
Next >
Finish
Cancel

Jos julkaiset netissä, anna tarkka osoite, mihin ohjelman sijoitat, eli sinun verkkohotellisi domain ja kansionimet (mutta huom. ei esim. index tiedoston tai muun html, tai php tiedoston ym. nimeä). Voit valita myös muun julkaisutavan. Huomaa, että eteen laitettava http:// on pakollinen. Klikattuasi ”next” aukeaa seuraava näyttö:



Jo valitset alemman vaihtoehdon (no), pelin voi käynnistää vain netistä. Ylemmällä valinnalla (yes) pelin voi lataamisen jälkeen käynnistää myös suoraan koneelta ilman internettiä.

Viimeinen ikkuna on:



Tässä ei enään tarvitse muuta, kuin klikata ”Finish”, ja pelistä tulee projekti siihen kansioon kovalevyllesi, minkä aluksi annoit. Jos julkaisit kuten tässä edettiin, eli netiä varten, kanioon tulee ”publish.htm” tiedosto, mistä peli kuuluu nettiin laiton jälkeen ladata ja asentaa. Asennuksen jälkeen pelin voi käynnistää seuraavilla pelikerroilla application tiedostosta, eli esim. tiedostosta ”Salaman_Torjuntaa.application” (tässä oppaassa opastetun pelin nimeä käyttäen).

XNA:ssa pelin varastaminen on pyritty tekemään vaikeaksi juuri esim. sillä, että se asennetaan tiettyyn verkko-osoitteeseen, joka asentuu ohjelmaan julkaistaessa.

21: Koko pelin lähdekoodi

Tässä on lopuksi julkaistu koko pelin lähdekoodi ilman php tiedostoja, jotka ovat nähtävillä ”Ennätystaulukot” kohdassa. Pelistä tietenkin puuttuvat kaikki kuvat, musiikki ja fonttien määrittelyt, jotka on tehtävä Content solmussa. Peli on pelattavissa näillä sivuilla esimerkkipeli kategoriassa. Jos haluat testata myös ennätystaulukkoja, tarvitset tietenkin nettihotellin tai omalle koneelle asennetun palvelimen PHP tuella, ja joudut muuttamaan koodissa olevaa osoitetta grtScores() ja sendScore() metodeissa, sekä keksimään oman avainkoodin, jolla vain sinun julkaisemassa peliversiossa pisteitä voi syöttää sinun piste tiedostoon palvelimelle.

```
/*************************************************************************
 * (c)Paavo Räisänen 2012
 * www.pelilakka.com
 * ***********************************************************************
 */

using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;
using System.Xml.Serialization;
using System.IO;
using System.Net;
using System.Text;

namespace Salaman_Torjuntaa
{

    public class Game1 : Microsoft.Xna.Framework.Game
    {
        GraphicsDeviceManager graphics;
        SpriteBatch spriteBatch;

        //Määritellään pelitilaa varten enum lueteltu tyyppi
        enum peliTila { aloitus, ohje, tekijat, pelaa, virhe, ennatukset, yhteysvirhe,
lopetus };

        //Määritellään peliTilaNyt ja asetetaan alkuarvoksi "aloitus"
        peliTila peliTilaNyt = peliTila.aloitus;

        SpriteFont spriteFontti;
        SpriteFont font2;
        Song musiikki;

        Texture2D aloituskuva;
        Texture2D alus;
        Texture2D salamaKuva;
        Texture2D raeKuva;
        Texture2D lokkiKuva;
        Texture2D tausta;
        Texture2D ammus;
        Texture2D perhonen;
        Texture2D kotka;
```

```
Texture2D tahti;
Texture2D pisara;
Texture2D ammusLisaus;
Texture2D lokkiKuvaPieni;

Vector2 aluksenPaikka = new Vector2();
Vector2 salamaPaikka = new Vector2();
Vector2[] raePaikka = new Vector2[4];
Vector2[] lokkiPaikka = new Vector2[6];
Vector2 ammusPaikka = new Vector2();
Vector2 paikka;
Vector2 nopeus2;
Vector2 paikkaKotka;
Vector2 paikkaTahti;
Vector2 ammusLisausPaikka;
Vector2 pisaraPaikka;

Point kehysKokoLokki = new Point(40, 20);
Point nykyinenKehysLokki = new Point(0, 0);
Point arkkiKokoLokki = new Point(3, 1);

Point kehysKokoPerhonen = new Point(30, 30);
Point nykyinenKehysPerhonen = new Point(0, 0);
Point arkkiKokoPerhonen = new Point(2, 1);

Point kehysKokoKotka = new Point(40, 20);
Point nykyinenKehysKotka = new Point(0, 0);
Point arkkiKokoKotka = new Point(3, 1);

int pituus;
char[] nimi = new char[20];
string sana;
string[] jono;
string[] jonoApu = new string[20];
bool loytyi;
bool alku2 = true;
string pelaajaNimi;
string postString;
bool yhtvirhe;
int yhtvir;
bool ennatus;
string ennatukset;

//Määritelmät musiikkia varten
/*AudioEngine audioMoottori;
WaveBank aaltoKentta;
SoundBank aaniKentta;
Cue aloitusRaita;
Cue peliRaita;
*/
double alusX, alusY;
double suuntaX, suuntaY;
double ammusSuuntaX, ammusSuuntaY;
double suuntaKotkaX, suuntaKotkaY;
double nopeus, nop;
double salamaY;
double[] raeX = new double[4];
double[] raeY = new double[4];
double[] k = new double[6];
double[] lokkiX = new double[6];
double[] lokkiY = new double[6];
double[] lokkiSuuntaX = new double[6];
double[] lokkiSuuntaY = new double[6];
double ammusX, ammusY;
```

```
        int pisteet;
        int elamat;
        int laskuri;
        int ammusMaara;
        int bonus;


        Random satLuku = new Random();

        bool salama;
        bool[] rae = new bool[4];
        bool[] alku = new bool[6];
        bool laukaus, apuLaukaus;
        bool suuntaKotkaBool;
        bool liikeMuutos;
        bool tahtiBool;
        bool ammusLisaysBool;
        bool pisaraBool;
        bool bonusLaskuri;
        bool[] tormays = new bool[4];
        bool ammusAlku;
        bool syotetty;

        int aikaViimeKehyksesta;
        int millisekunnitKehystaKohti;
        int taso;
        double aika;

        int perhonenRnd;
        int laskuri2;
        int laskuri3;
        int aloitus;
        int lokitMaara;

        int h;
        double kulma;

        double ammusSuunta2X, ammusSuunta2Y;

        public Game1()
        {

            graphics = new GraphicsDeviceManager(this);
            Content.RootDirectory = "Content";

            IsMouseVisible = true;

            //Tarkistetaan onko näyttö riittävän suuri, eli vähintään 1100*800 pixeliä
            graphics.PreferredBackBufferWidth = 1100;
            graphics.PreferredBackBufferHeight = 800;

#if    !DEBUG
            graphics.IsFullScreen       true;
#endif


        }

        protected override void Initialize()
        {
            base.Initialize();
        }

        protected override void LoadContent()
        {
```

```
        spriteFontti = Content.Load<SpriteFont>(@"Fontit\Fontti1");
        font2 = Content.Load<SpriteFont>(@"Fontit\Fontti2");

        aloituskuva = Content.Load<Texture2D>(@"Kuvat/Oulujoki");
        alus = Content.Load<Texture2D>(@"Kuvat/Kori");
        salamaKuva = Content.Load<Texture2D>(@"Kuvat/Salama");
        raeKuva = Content.Load<Texture2D>(@"Kuvat/Rae");
        lokkiKuva = Content.Load<Texture2D>(@"Kuvat/Lokit");
        tausta = Content.Load<Texture2D>(@"Kuvat/Tausta");
        ammus = Content.Load<Texture2D>(@"Kuvat/Ammus");
        perhonen = Content.Load<Texture2D>(@"Kuvat/Perhonen");
        kotka = Content.Load<Texture2D>(@"Kuvat/Kotka");
        tahti = Content.Load<Texture2D>(@"Kuvat/Tahti");
        pisara = Content.Load<Texture2D>(@"Kuvat/Pisara");
        ammusLisaus = Content.Load<Texture2D>(@"Kuvat/AmmusLisaus");
        lokkiKuvaPieni = Content.Load<Texture2D>(@"Kuvat/LokkiKuva");

        /*
        audioMoottori = new AudioEngine(@"Content\Audio\SalamaAudio.xgs");
        aaltoKentta = new WaveBank(audioMoottori, @"Content\Audio\Wave Bank.xwb");
        aaniKentta = new SoundBank(audioMoottori, @"Content\Audio\Sound Bank.xsb");
        */
        musiikki = Content.Load<Song>(@"Audio/Raita");
        MediaPlayer.IsRepeating = true; //Sallii ympäritoiston
        MediaPlayer.Play(musiikki);
        /*
         peliRaita = aaniKentta.GetCue("peliRaita");
         peliRaita.Play();
         peliRaita.Pause();

         aloitusRaita = aaniKentta.GetCue("aloitusRaita");
         aloitusRaita.Play();
         * */
        // Create a new SpriteBatch, which can be used to draw textures.
        spriteBatch = new SpriteBatch(GraphicsDevice);

    }

    protected override void UnloadContent()
    {
        // TODO: Unload any non ContentManager content here
    }


    protected override void Update(GameTime gameTime)
    {
        switch (peliTilaNyt)
        {
            case peliTila.aloitus:
                /* if (aloitusRaita.IsStopped)
                 {
                     aloitusRaita = aaniKentta.GetCue("aloitusCue");
                     aloitusRaita.Play();
                 }*/

                MediaPlayer.Resume();
                //Muuttujien alkumääritykset
                pisteet = 0;
                elamat = 6;
                alusX = 550;
                alusY = 250;
                nopeus = 0.5;
                salamaPaikka.Y = 0;
                salama = false;
                taso = 1;
```

```
laskuri = 0;
aika = 0;
laukaus = false;
paikka.X = 550;
paikka.Y = 500;
aloitus = 1;
liikeMuutos = true;
laskuri3 = 0;
ammusPaikka.X = 1200;
ammusPaikka.Y = 1000;
tahtiBool = false;
ammusLisaysBool = false;
pisaraBool = false;
ammusMaara = 10;
lokitMaara = 0;
bonus = 0;
bonusLaskuri = false;
ammusSuuntaX = 10;
ammusSuuntaY = 0;
millisekunnitKehystaKohti = 80;
aikaViimeKehyksesta = 0;
ammusAlku = false;
syotetty = false;
ennatus = false;
pituus = 0;
sana = "";
loytyi = false;
ennatukset = "";
postString = "";
yhtvirhe = false;
yhtvir=0;
pituus = 0;
loytyi = false;


if (alku2)
{
    try
    {

        postString = getScores();
        alku2 = false;
        jono = postString.Split(',');
    }
    catch { }
}
if (yhtvirhe)
{
    yhtvir = 1;
    peliTilaNyt = peliTila.yhteysvirhe;
}
for (int i = 0; i < 4; i++) rae[i] = false;
for (int i = 0; i < 6; i++) alku[i] = true;
for (int i = 0; i < 4; i++)
{
    lokkiY[i] = 0;
    lokkiX[i] = 0;
    tormays[i] = false;
}


//Enterin painalluksen lukeminen. Jos Enter panettu, aloitetaan peli.
if (Keyboard.GetState().IsKeyDown(Keys.Enter))
{
    peliTilaNyt = peliTila.pelaa;
```

```
        }

        //Luetaan, onko O näppäin painettu, jolloin mennään ohjeeseen
        if (Keyboard.GetState().IsKeyDown(Keys.O))
        {
            peliTilaNyt = peliTila.ohje;
        }

        //Luetaan, onko Z näppäin painettu, jolloin mennään ohjeeseen
        if (Keyboard.GetState().IsKeyDown(Keys.Z))
        {
            peliTilaNyt = peliTila.tekijat;
        }

        if (Keyboard.GetState().IsKeyDown(Keys.E))
        {
            peliTilaNyt = peliTila.ennatukset;
        }
        break;

    case peliTila.ohje:
        //Jos painetaan T, palataan alkuun.
        if (Keyboard.GetState().IsKeyDown(Keys.T))
        {
            peliTilaNyt = peliTila.aloitus;
        }
        break;
    case peliTila.tekijat:
        //Jos painetaan T, palataan alkuun.
        if (Keyboard.GetState().IsKeyDown(Keys.T))
        {
            peliTilaNyt = peliTila.aloitus;
        }
        break;
    case peliTila.ennatukset:

        //Jos painetaan T, palataan alkuun.
        if (Keyboard.GetState().IsKeyDown(Keys.T))
        {
            peliTilaNyt = peliTila.aloitus;
        }
        break;
    case peliTila.yhteysvirhe:
        {
            if (Keyboard.GetState().IsKeyDown(Keys.T) && yhtvir==1)
            {
                yhtvirhe = false;
                alku2 = true;
                peliTilaNyt = peliTila.aloitus;
            }
            if (Keyboard.GetState().IsKeyDown(Keys.T) && yhtvir == 2)
            {
                yhtvirhe = false;
                postString = "";
                try
                {

                    postString = getScores();

                    jono = postString.Split(',');
                }
                catch
                {

                }
```

```
                    peliTilaNyt = peliTila.lopetus;
                }
                break;

            }
        case peliTila.pelaa:
            aika++;
            /*aloitusRaita.Pause();
            if (peliRaita.IsPaused) peliRaita.Resume();
            if (peliRaita.IsStopped)
            {
                peliRaita = aaniKentta.GetCue("peliRaita");
                peliRaita.Play();
            }*/
            if (aika == 3600 || aika == 7200) taso++;
            if (aika > 4000 && aika % 1800 == 0) nopeus += 0.1;

            //Jos painetaan välilyöntinäppäin alas, asetetaan laukaus arvoon ”true”
            if (Keyboard.GetState().IsKeyDown(Keys.Space) && ammusMaara > 0)
                laukaus = true;

            //Kun vapautetaan välilyöntinäppäin ylös, ammutaan
            if (Keyboard.GetState().IsKeyUp(Keys.Space) && laukaus == true)
            {
                apuLaukaus = true;
                ammusAlku = true;
            }

            Ammus();

            //Mikäli käytetään bonusta, siiretään perhosta,
            //ja tyhjennetään näyttö linnuista ja rakeista
            if (Keyboard.GetState().IsKeyDown(Keys.Tab) && bonusLaskuri == false &&
bonus > 0)
            {
                bonusLaskuri = true;
                Tyhjennys();
            }
            if (Keyboard.GetState().IsKeyUp(Keys.Tab) && bonusLaskuri == true &&
bonus > 0)
            {
                bonus--;
                bonusLaskuri = false;
            }


            //Näppäinkontrollit aluksen siirtoon
            AluksenSiirto();

            //Salaman liikutus
            Salama();

            //Rakeiden liikutus
            Rae();

            //Lokkien liikutus
            Lokit();

            //Perhosen liikutus
            Perhonen();

            //Kotkan liikutus
            Kotka();

            //Tähti, eli lisäalus
```

```
Tahti();

//Ammuslisäyspallo
AmmusLisaus();

//Pisara, eli lisäpisteitä
Pisara();

//Tarkistetaan törmäys salamaan
if (SalamaTormaus())
{
    pisteet += 100;
    salama = false;
    salamaY = 0;
}

//Tarkistetaan, poimittiinko tähti
if (TahtiAlus())
{
    elamat++;
    tahtiBool = false;
    paikkaTahti.Y = 1000;
}

//Tarkistetaan, poimittiinko lisäammukset
if (AmmusLisaysAlus())
{
    ammusMaara += 10;
    ammusLisaysBool = false;
    ammusLisausPaikka.Y = 1000;
}

//Tarkistetaan, poimittiinko pisara
if (PisaraAlus())
{
    pisteet += 400;
    pisaraBool = false;
    pisaraPaikka.Y = 1000;
}

//Tarkistetaan törmäykset rakeisiin
for (int i = 0; i < 4; i++)
{
    if (RaeTormaus(i))
    {
        for (int j = 0; j < 4; j++)
        {
            rae[j] = false;
            raeY[i] = 0;
        }
        elamat--;
        laskuri = 0;
        peliTilaNyt = peliTila.virhe;
        break;
    }
}

//Tarkistetaan törmäykset lokkeihin
for (int i = 0; i < 1 + taso; i++)
{
    if (LokkiTormaus(i))
    {
        for (int j = 0; j < 1 + taso; j++)
        {
            alku[j] = true;
```

```
            Lokit();
        }
        elamat--;
        laskuri = 0;
        peliTilaNyt = peliTila.virhe;
        break;
    }
}

//Tarkistetaan aluksen tai ammuksen törmäys perhoseen
if (PerhonenTormaus() || PerhonenAmmus())
{
    elamat--;
    laskuri = 0;
    laukaus = false;
    apuLaukaus = false;
    ammusPaikka.X = -200;
    ammusPaikka.Y = -100;
    laukaus = false;
    peliTilaNyt = peliTila.virhe;
    PerhosenSiirto();
}

//Tarkistetaan aluksen tai ammuksen törmäys kotkaan
if (KotkaAlus() || KotkaAmmus())
{
    elamat--;
    laskuri = 0;
    laskuri3 = 0;
    laukaus = false;
    apuLaukaus = false;
    ammusPaikka.X = -200;
    ammusPaikka.Y = -100;
    peliTilaNyt = peliTila.virhe;

    //Siirretään kotka alkuun
    suuntaKotkaBool = false;
    paikkaKotka.X = -50;
}

//Tarkistetaan ammuksen törmäykset lokkeihin
for (int i = 0; i < 1 + taso; i++)
{
    for (int j = 0; j < 1 + taso; j++)
    {
        tormays[j] = AmmusTormaus(j);

    }
    if (tormays[i] == true)
    {
        lokitMaara++;
        laukaus = false;
        apuLaukaus = false;
        ammusPaikka.X = -200;
        ammusPaikka.Y = -100;
        if (lokitMaara == 4)
        {
            lokitMaara = 0;
            bonus++;
        }
        for (int j = 0; j < 1 + taso; j++)
        {
            laukaus = false;
            alku[j] = true;
            Lokit();
```

```
                }
                pisteet += 100;
                break;
            }
        }

        //Lokkien animaatiota ohjaava koodipätkä
        aikaViimeKehyksesta += gameTime.ElapsedGameTime.Milliseconds;
        if (aikaViimeKehyksesta > millisekunnitKehystaKohti)
        {
            aikaViimeKehyksesta -= millisekunnitKehystaKohti;
            ++nykyinenKehysLokki.X;
            if (nykyinenKehysLokki.X >= arkkiKokoLokki.X)
            {
                nykyinenKehysLokki.X = 0;
                ++nykyinenKehysLokki.Y;
                if (nykyinenKehysLokki.Y >= arkkiKokoLokki.Y)
                    nykyinenKehysLokki.Y = 0;
            }
        }

        //Perhosen animaatiota ohjaava koodipätkä
        aikaViimeKehyksesta += gameTime.ElapsedGameTime.Milliseconds;
        if (aikaViimeKehyksesta > millisekunnitKehystaKohti)
        {
            aikaViimeKehyksesta -= millisekunnitKehystaKohti;
            ++nykyinenKehysPerhonen.X;
            if (nykyinenKehysPerhonen.X >= arkkiKokoPerhonen.X)
            {
                nykyinenKehysPerhonen.X = 0;
                ++nykyinenKehysPerhonen.Y;
                if (nykyinenKehysPerhonen.Y >= arkkiKokoPerhonen.Y)
                    nykyinenKehysPerhonen.Y = 0;
            }
        }

        //Kotkan animaatiota ohjaava koodipätkä
        aikaViimeKehyksesta += gameTime.ElapsedGameTime.Milliseconds;
        if (aikaViimeKehyksesta > millisekunnitKehystaKohti)
        {
            aikaViimeKehyksesta -= millisekunnitKehystaKohti;
            ++nykyinenKehysKotka.X;
            if (nykyinenKehysKotka.X >= arkkiKokoKotka.X)
            {
                nykyinenKehysKotka.X = 0;
                ++nykyinenKehysKotka.Y;
                if (nykyinenKehysKotka.Y >= arkkiKokoKotka.Y)
                    nykyinenKehysKotka.Y = 0;
            }
        }

        //Tarkistetaan loppuuko peli
        if (elamat == 0) peliTilaNyt = peliTila.lopetus;
        break;
    case peliTila.virhe:
        Tyhjennys();
        if (laskuri > 60) peliTilaNyt = peliTila.pelaa;
        break;
    case peliTila.lopetus:
        //peliRaita.Pause();
        aika++;
        MediaPlayer.Stop();
        if (Keyboard.GetState().IsKeyDown(Keys.F12))
        {
            alku2 = true;
```

```
                    peliTilaNyt = peliTila.aloitus;
                }

                if (Keyboard.GetState().IsKeyDown(Keys.Escape)) Exit();

                if (int.Parse(jono[19]) <= pisteet && pisteet>0)
                {
                    ennatus = true;
                    if (pituus < 20 && aika % 10 == 0) pelaajaNimi = Kirjoita();
                    if (Keyboard.GetState().IsKeyDown(Keys.Back) && aika % 10 == 0 &&
pituus >= 1)
                    {
                        pituus--;
                        nimi[pituus] = ' ';
                        pelaajaNimi = Kirjoita();
                    }

                    if (Keyboard.GetState().IsKeyDown(Keys.Enter) && !syotetty)
                    {
                        try
                        {
                            //Ladataan vielä ennen tietojen tallennusta tuore
ennätystaulukko
                            //palvelimelta, ettei nimeä kirjoittaessa ole joku jo
rikkonut
                            //ennätystulosta
                            postString = "";
                            postString = getScores();
                            jono = postString.Split(',');

                            for (int i = 0; i < 20; i++) jonoApu[i] = jono[i];
                            for (int i = 1; i < 20; i = i + 2)
                            {
                                if (int.Parse(jono[i]) <= pisteet)
                                {
                                    loytyi = true;
                                    for (int j = i + 1; j < 20; j = j + 2)
                                    {
                                        jono[j] = jonoApu[j - 2];
                                        jono[j + 1] = jonoApu[j - 1];
                                    }
                                    jono[i - 1] = pelaajaNimi;
                                    jono[i] = pisteet.ToString();
                                    break;
                                }
                                if (loytyi)
                                {
                                    h = i + 2;

                                    break;
                                }
                            }

                            ennatukset = "";
                            for (int i = 0; i < 20; i++) ennatukset = ennatukset +
jono[i] + ",";
                            try
                            {
                                sendScore(ennatukset);
                                syotetty = true;



                                postString = "";
                                postString = getScores();
```

```
                                jono = postString.Split(',');
                            }
                            catch { for (int i = 0; i < 20; i++) jono =
ennatukset.Split(','); }

                            if (yhtvirhe)
                            {
                                yhtvir = 2;
                                peliTilaNyt = peliTila.yhteysvirhe;
                            }
                        }

                        catch { }
                    }
                    if (yhtvirhe)
                    {
                        yhtvir = 2;
                        peliTilaNyt = peliTila.yhteysvirhe;
                    }
                }
                break;
            }
            // Sallii pelin lopetuksen
            if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
                this.Exit();

            base.Update(gameTime);
        }

        protected override void Draw(GameTime gameTime)
        {
            GraphicsDevice.Clear(Color.Black);

            spriteBatch.Begin();

            if (peliTilaNyt == peliTila.aloitus)
            {
                spriteBatch.Draw(aloituskuva, new Vector2(100, 200), null, Color.White, 0,
Vector2.Zero, 1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "peliLakka.com",
                        new Vector2(10, 10), Color.Green, 0, Vector2.Zero,
                        1, SpriteEffects.None, 0);
                spriteBatch.DrawString(font2, "Salaman Torjuntaa",
                  new Vector2(40, 60), Color.Blue, 0, Vector2.Zero,
                  1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "Valitse:",
                 new Vector2(400, 220), Color.Yellow, 0, Vector2.Zero,
                 1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "E ennätykset",
                new Vector2(400, 260), Color.Yellow, 0, Vector2.Zero,
                1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "O ohjeet",
                 new Vector2(400, 300), Color.Yellow, 0, Vector2.Zero,
                 1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "Z tekijöistä",
                 new Vector2(400, 340), Color.Yellow, 0, Vector2.Zero,
                 1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "Enter aloittaa",
                  new Vector2(400, 380), Color.Yellow, 0, Vector2.Zero,
                  1, SpriteEffects.None, 0);
            }

            if (peliTilaNyt == peliTila.ohje)
            {
```

```
                spriteBatch.Draw(aloituskuva, new Vector2(100, 200), null, Color.White, 0,
Vector2.Zero, 1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "peliLakka.com",
                       new Vector2(10, 10), Color.Green, 0, Vector2.Zero,
                       1, SpriteEffects.None, 0);
                spriteBatch.DrawString(font2, "Salaman Torjuntaa",
                  new Vector2(40, 60), Color.Blue, 0, Vector2.Zero,
                  1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "T takaisin",
                  new Vector2(400, 750), Color.Yellow, 0, Vector2.Zero,
                  1, SpriteEffects.None, 0);

                Int32[] pixel = { 0xffffff };
                Texture2D viiva = new Texture2D(GraphicsDevice, 1, 1, false,
                  SurfaceFormat.Color);
                viiva.SetData<Int32>(pixel, 0, viiva.Width * viiva.Height);
                Color vari = Color.Blue;
                Rectangle nelio = new Rectangle(300, 200, 800, 600);
                spriteBatch.DrawString(spriteFontti, "Pelissä pyrit torjumaan salamat niin,
että ne",
                       new Vector2(320, 220), Color.Red, 0, Vector2.Zero,
                       1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "eivät pääse alas asti.",
                      new Vector2(320, 245), Color.Red, 0, Vector2.Zero,
                      1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "Mikäli salama pääsee kaupunkiin, tulee
virhe.",
                       new Vector2(320, 270), Color.Red, 0, Vector2.Zero,
                       1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "Lokkeja, eli valkoisia lintuja voi
ampua.",
                       new Vector2(320, 310), Color.Red, 0, Vector2.Zero,
                       1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "Neljästä ammutusta lokista saa
bonuspisteen,",
                       new Vector2(320, 335), Color.Red, 0, Vector2.Zero,
                       1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "jonka voi hyödyntää painamalla
sarkainta(tab).",
                       new Vector2(320, 360), Color.Red, 0, Vector2.Zero,
                       1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "Bonus tyhjentää virhemahdollisuuksista
muuten",
                     new Vector2(320, 385), Color.Red, 0, Vector2.Zero,
                     1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "näytön, mutta vain siirtää perhosta.",
                     new Vector2(320, 410), Color.Red, 0, Vector2.Zero,
                     1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "Ampuminen tapahtuu
välilyöntinäppäimestä ja",
                      new Vector2(320, 435), Color.Red, 0, Vector2.Zero,
                      1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "ammutusta lokista saa 100 pistettä.",
                      new Vector2(320, 470), Color.Red, 0, Vector2.Zero,
                      1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "Ruskeat linnut eli kotkat ja perhoset
ovat",
                      new Vector2(320, 505), Color.Red, 0, Vector2.Zero,
                      1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "rauhoitettuja. Jos osut niihin, tulee
virhe.",
                      new Vector2(320, 530), Color.Red, 0, Vector2.Zero,
                      1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "Jos alus osuu lokkiin, kotkaan,
perhoseen",
```

```
                    new Vector2(320, 555), Color.Red, 0, Vector2.Zero,
                    1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "tai rakeeseen (valkoinen), tulee
virhe.",
                    new Vector2(320, 580), Color.Red, 0, Vector2.Zero,
                    1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "Putoavasta tähdestä saat lisäelämän,
ja",
                    new Vector2(320, 605), Color.Red, 0, Vector2.Zero,
                    1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "putoavasta ruskeasta pallosta 10
lisäammusta.",
                   new Vector2(320, 630), Color.Red, 0, Vector2.Zero,
                   1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "Putoavasta vesipisarasta saa 400
pistettä.",
                    new Vector2(320, 655), Color.Red, 0, Vector2.Zero,
                    1, SpriteEffects.None, 0);
                spriteBatch.Draw(viiva, nelio, vari);
            }

            if (peliTilaNyt == peliTila.tekijat)
            {
                spriteBatch.Draw(aloituskuva, new Vector2(100, 200), null, Color.White, 0,
Vector2.Zero, 1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "peliLakka.com",
                       new Vector2(10, 10), Color.Green, 0, Vector2.Zero,
                       1, SpriteEffects.None, 0);
                spriteBatch.DrawString(font2, "Salaman Torjuntaa",
                  new Vector2(40, 60), Color.Blue, 0, Vector2.Zero,
                  1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "T takaisin",
                  new Vector2(400, 750), Color.Yellow, 0, Vector2.Zero,
                  1, SpriteEffects.None, 0);

                Int32[] pixel = { 0xffffff };
                Texture2D viiva = new Texture2D(GraphicsDevice, 1, 1, false,
                  SurfaceFormat.Color);
                viiva.SetData<Int32>(pixel, 0, viiva.Width * viiva.Height);
                Color vari = Color.Black;
                Rectangle nelio = new Rectangle(300, 200, 800, 600);
                spriteBatch.DrawString(spriteFontti, "Ohjelmointi ja grafiikka:",
                       new Vector2(320, 220), Color.Red, 0, Vector2.Zero,
                       1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "Paavo Räisänen",
                       new Vector2(320, 250), Color.Red, 0, Vector2.Zero,
                       1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "Musiikki:",
                       new Vector2(320, 280), Color.Red, 0, Vector2.Zero,
                       1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "Mozart by (thanks to)
www.archive.org",
                       new Vector2(320, 310), Color.Red, 0, Vector2.Zero,
                       1, SpriteEffects.None, 0);
            }

            if (peliTilaNyt == peliTila.ennatukset)
            {
                spriteBatch.Draw(aloituskuva, new Vector2(100, 200), null, Color.White, 0,
Vector2.Zero, 1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "peliLakka.com",
                       new Vector2(10, 10), Color.Green, 0, Vector2.Zero,
                       1, SpriteEffects.None, 0);
                spriteBatch.DrawString(font2, "Salaman Torjuntaa",
                  new Vector2(40, 60), Color.Blue, 0, Vector2.Zero,
```

```
                  1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "T takaisin",
                  new Vector2(400, 750), Color.Yellow, 0, Vector2.Zero,
                  1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "Nimi",
                  new Vector2(100, 175), Color.Yellow, 0, Vector2.Zero,
                  1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "Pisteet",
                       new Vector2(460, 175), Color.Yellow, 0, Vector2.Zero,
                       1, SpriteEffects.None, 0);
                for (int i = 0; i < 20; i = i + 2)
                {
                    spriteBatch.DrawString(spriteFontti, i / 2 + 1 + ":",
                       new Vector2(40, i * 15 + 200), Color.Yellow, 0, Vector2.Zero,
                       1, SpriteEffects.None, 0);
                    spriteBatch.DrawString(spriteFontti, jono[i],
                       new Vector2(100, i * 15 + 200), Color.Yellow, 0, Vector2.Zero,
                       1, SpriteEffects.None, 0);
                   spriteBatch.DrawString(spriteFontti, jono[i + 1],
                       new Vector2(480, i * 15 + 200), Color.Yellow, 0, Vector2.Zero,
                       1, SpriteEffects.None, 0);
                }
            }
            if (peliTilaNyt == peliTila.pelaa)
            {
                //Piirretään tausta
                spriteBatch.Draw(tausta, Vector2.Zero, null, Color.White, 0, Vector2.Zero,
1, SpriteEffects.None, 0);

                //Piirretään alus
                aluksenPaikka.X = (int)alusX;
                aluksenPaikka.Y = (int)alusY;
                spriteBatch.Draw(alus, aluksenPaikka, null, Color.White, 0, Vector2.Zero, 1,
SpriteEffects.None, 0);

                //Piirretään rakeet
                for (int i = 0; i < 1 + taso; i++)
                {
                    raePaikka[i].X = (int)raeX[i];
                    raePaikka[i].Y = (int)raeY[i];
                    spriteBatch.Draw(raeKuva, raePaikka[i], null, Color.White, 0,
Vector2.Zero, 1, SpriteEffects.None, 0);
                }

                //Piirretään salama
                salamaPaikka.Y = (int)salamaY;
                spriteBatch.Draw(salamaKuva, salamaPaikka, null, Color.White, 0,
Vector2.Zero, 1, SpriteEffects.None, 0);

                //Piirretään tähti
                if (tahtiBool == true)
                    spriteBatch.Draw(tahti, paikkaTahti, null, Color.White, 0, Vector2.Zero,
1, SpriteEffects.None, 0);

                //Piirretään lisäammus pallo
                if (ammusLisaysBool == true)
                    spriteBatch.Draw(ammusLisaus, ammusLisausPaikka, null, Color.White, 0,
Vector2.Zero, 1, SpriteEffects.None, 0);

                //Piirretään Pisara
                if (pisaraBool == true)
                    spriteBatch.Draw(pisara, pisaraPaikka, null, Color.White, 0,
Vector2.Zero, 1, SpriteEffects.None, 0);

                //Piirretään perhonen
```

```
                spriteBatch.Draw(perhonen, paikka, new Rectangle(nykyinenKehysPerhonen.X *
kehysKokoPerhonen.X,
                        nykyinenKehysPerhonen.Y * kehysKokoPerhonen.Y, kehysKokoPerhonen.X,
                          kehysKokoPerhonen.Y), Color.White, 0, Vector2.Zero, 1,
SpriteEffects.None, 0);

                //Piirretään kotka
                paikkaKotka.X = (int)suuntaKotkaX;
                paikkaKotka.Y = (int)suuntaKotkaY;
                spriteBatch.Draw(kotka, paikkaKotka, new Rectangle(nykyinenKehysKotka.X *
kehysKokoKotka.X,
                       nykyinenKehysKotka.Y * kehysKokoKotka.Y, kehysKokoKotka.X,
                         kehysKokoKotka.Y), Color.White, 0, Vector2.Zero, 1,
SpriteEffects.None, 0);

                //Piirretään lokit
                for (int i = 0; i < 1 + taso; i++)
                {
                    lokkiPaikka[i].X = (int)lokkiX[i];
                    lokkiPaikka[i].Y = (int)lokkiY[i];

                    spriteBatch.Draw(lokkiKuva, lokkiPaikka[i], new
Rectangle(nykyinenKehysLokki.X * kehysKokoLokki.X,
                        nykyinenKehysLokki.Y * kehysKokoLokki.Y, kehysKokoLokki.X,
                          kehysKokoLokki.Y), Color.White, 0, Vector2.Zero, 1,
SpriteEffects.None, 0);
                }
                for (int i = 0; i < lokitMaara; i++)
                    spriteBatch.Draw(lokkiKuvaPieni, new Vector2(10 + i * 25, 10), null,
Color.White, 0, Vector2.Zero, 1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "Ammukset:" + ammusMaara + "
Pisteet:" + pisteet + "    Elämät:" + elamat + "    Bonukset:" + bonus,
                  new Vector2(100, 10), Color.Blue, 0, Vector2.Zero,
                  1, SpriteEffects.None, 0);


                if (apuLaukaus == true)
                {
                    if (ammusAlku == true)
                    {
                        ammusX = aluksenPaikka.X;
                        ammusY = aluksenPaikka.Y;
                        ammusMaara--;
                        ammusAlku = false;
                        laukaus = false;
                        ammusSuunta2X = ammusSuuntaX;
                        ammusSuunta2Y = ammusSuuntaY;
                    }

                    ammusX += ammusSuunta2X;
                    ammusY += ammusSuunta2Y;

                    ammusPaikka.X = (int)ammusX;
                    ammusPaikka.Y = (int)ammusY;

                    if (ammusPaikka.X > 1100 || ammusPaikka.X < 0 || ammusPaikka.Y > 550 ||
ammusPaikka.Y < 0)
                    {
                        apuLaukaus = false;
                    }
                    spriteBatch.Draw(ammus, ammusPaikka, null, Color.White, 0, Vector2.Zero,
1, SpriteEffects.None, 0);
                }
            }
```

```
            if (peliTilaNyt == peliTila.virhe)
            {

                laskuri++;
                if (laskuri % 20 == 0)
                {
                    //Piirtää valkoisen neliön välähdyksenä
                    Int32[] pixel = { 0xffffff };
                    Texture2D viiva = new Texture2D(GraphicsDevice, 1, 1, false,
                      SurfaceFormat.Color);
                    viiva.SetData<Int32>(pixel, 0, viiva.Width * viiva.Height);
                    Color vari = Color.White;
                    Rectangle nelio = new Rectangle(0, 0, 1100, 800);
                    spriteBatch.Draw(viiva, nelio, vari);

                }
                else spriteBatch.Draw(tausta, Vector2.Zero, null, Color.White, 0,
Vector2.Zero, 1, SpriteEffects.None, 0);
            }

            if (peliTilaNyt == peliTila.lopetus)
            {
                GraphicsDevice.Clear(Color.Blue);
                spriteBatch.DrawString(font2, "Salaman Torjuntaa",
                  new Vector2(300, 40), Color.Yellow, 0, Vector2.Zero,
                  1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "Peli loppui",
                  new Vector2(450, 130), Color.Yellow, 0, Vector2.Zero,
                  1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "Pisteesi:" + pisteet,
                   new Vector2(450, 170), Color.Yellow, 0, Vector2.Zero,
                   1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "F12 Aloitusikkunaan",
                  new Vector2(380, 200), Color.Yellow, 0, Vector2.Zero,
                  1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "esc lopettaa",
                  new Vector2(450, 225), Color.Yellow, 0, Vector2.Zero,
                  1, SpriteEffects.None, 0);
                if (ennatus == true)
                {
                    spriteBatch.DrawString(spriteFontti, "Pääsit ennätystaulukoihin",
                new Vector2(360, 260), Color.Yellow, 0, Vector2.Zero,
                1, SpriteEffects.None, 0);
                    spriteBatch.DrawString(spriteFontti, "Anna nimesi: " + pelaajaNimi,
                new Vector2(410, 290), Color.Yellow, 0, Vector2.Zero,
                1, SpriteEffects.None, 0);
                    spriteBatch.DrawString(spriteFontti, "Ohje: ääkköset F1 å, F2 ä ja F3
ö",
                      new Vector2(285, 750), Color.Yellow, 0, Vector2.Zero,
                      1, SpriteEffects.None, 0);
                }
                else
                {
                    spriteBatch.DrawString(spriteFontti, "Et päässyt ennätystaulukoihin,
mutta yritä uudelleen",
                   new Vector2(200, 260), Color.Yellow, 0, Vector2.Zero,
                   1, SpriteEffects.None, 0);
                }
                spriteBatch.DrawString(spriteFontti, "Nimi",
                  new Vector2(400, 330), Color.Yellow, 0, Vector2.Zero,
                  1, SpriteEffects.None, 0);
                spriteBatch.DrawString(spriteFontti, "Pisteet",
                       new Vector2(780, 330), Color.Yellow, 0, Vector2.Zero,
                       1, SpriteEffects.None, 0);
```

```
                for (int i = 0; i < 20; i = i + 2)
                {
                    spriteBatch.DrawString(spriteFontti, i / 2 + 1 + ":",
                       new Vector2(320, i * 15 + 360), Color.Yellow, 0, Vector2.Zero,
                       1, SpriteEffects.None, 0);
                    spriteBatch.DrawString(spriteFontti, jono[i],
                       new Vector2(380, i * 15 + 360), Color.Yellow, 0, Vector2.Zero,
                       1, SpriteEffects.None, 0);
                    spriteBatch.DrawString(spriteFontti, jono[i + 1],
                       new Vector2(800, i * 15 + 360), Color.Yellow, 0, Vector2.Zero,
                       1, SpriteEffects.None, 0);
                }


            }

            if (peliTilaNyt == peliTila.yhteysvirhe)
            {
                GraphicsDevice.Clear(Color.Blue);
                spriteBatch.DrawString(font2, "Tarkista internet yhteytesi",
                  new Vector2(150, 40), Color.Yellow, 0, Vector2.Zero,
                  1, SpriteEffects.None, 0);
                spriteBatch.DrawString(font2, "ja paina T",
                  new Vector2(300, 100), Color.Yellow, 0, Vector2.Zero,
                  1, SpriteEffects.None, 0);
            }
            spriteBatch.End();

            base.Draw(gameTime);
        }

        //Metodi, joka ohjaa aluksen liikuttelun näppäintoimintoja
        public void AluksenSiirto()
        {
            //Tarkistetaan, mihin suuntaan mahdollisesti nuolta on painettu
            //ja valitaan sen mukainen suunta
            if (Keyboard.GetState().IsKeyDown(Keys.Right))
            {
                suuntaX = 6;
                suuntaY = 0;
            }
            if (Keyboard.GetState().IsKeyDown(Keys.Left))
            {
                suuntaY = 0;
                suuntaX = -6;
            }
            if (Keyboard.GetState().IsKeyDown(Keys.Up))
            {
                suuntaY = -6;
                suuntaX = 0;
            }

            if (Keyboard.GetState().IsKeyDown(Keys.Down))
            {
                suuntaY = 6;
                suuntaX = 0;
            }
            if (Keyboard.GetState().IsKeyDown(Keys.Right) &&
Keyboard.GetState().IsKeyDown(Keys.Up))
            {
                suuntaX = 6;
                suuntaY = -6;
            }
            if (Keyboard.GetState().IsKeyDown(Keys.Right) &&
Keyboard.GetState().IsKeyDown(Keys.Down))
```

```
            {
                suuntaY = 6;
                suuntaX = 6;
            }

            if (Keyboard.GetState().IsKeyDown(Keys.Left) &&
Keyboard.GetState().IsKeyDown(Keys.Up))
            {
                suuntaX = -6;
                suuntaY = -6;
            }
            if (Keyboard.GetState().IsKeyDown(Keys.Left) &&
Keyboard.GetState().IsKeyDown(Keys.Down))
            {
                suuntaX = -6;
                suuntaY = 6;
            }

            //Tarkistetaan että alus ei liiku näytön ulkopuolelle
            //ja sallitaan liikuttaminen muihin suuntiin kuin näytön ulkopuolelle
            if (aluksenPaikka.X < 0 && Keyboard.GetState().IsKeyDown(Keys.Left))
                nop = 0;
            else if (aluksenPaikka.X > 1080 && Keyboard.GetState().IsKeyDown(Keys.Right))
                nop = 0;
            else if (aluksenPaikka.Y < 0 && Keyboard.GetState().IsKeyDown(Keys.Up))
                nop = 0;
            else if (aluksenPaikka.Y > 500 && Keyboard.GetState().IsKeyDown(Keys.Down))
                nop = 0;
            else
                nop = nopeus;

            //Kasvatetaan koordinantteja, mikäli jotain nuolinäppäintä on painettu
            if (Keyboard.GetState().IsKeyDown(Keys.Right) ||
Keyboard.GetState().IsKeyDown(Keys.Left)
                || Keyboard.GetState().IsKeyDown(Keys.Up) ||
Keyboard.GetState().IsKeyDown(Keys.Down))
            {
                alusX = alusX + suuntaX * nop;
                alusY = alusY + suuntaY * nop;
            }
        }

        public void Salama()
        {
            //Jos salamaa ei ole näytöllä, arvotaan uusi salaman X koordinantti
            if (salama == false)
            {
                salamaPaikka.X = satLuku.Next(20, 1079);
                salama = true;
            }

            if (salamaPaikka.Y > 500)
            {
                salama = false;
                salamaY = 0;
                elamat--;
                laskuri = 0;
                peliTilaNyt = peliTila.virhe;
            }

            salamaY += 1.5 * nopeus;
        }

        //Rakeiden luonti ja liikutus
        public void Rae()
```

```
{
    for (int i = 0; i < 1 + taso; i++)
    {

        //Jos raetta ei ole, luodaan
        if (rae[i] == false)
        {
            raeX[i] = satLuku.Next(20, 1079);
            rae[i] = true;

            //Lasketaan rakeelle kulmakerroin niin, että se suuntautuu alusta kohti
            k[i] = (aluksenPaikka.X - raeX[i]) / aluksenPaikka.Y;

            //Rajoitetaan että kulmakerroin on välillä -2  < k < 2
            if (k[i] > 2) k[i] = 2;
            if (k[i] < -2) k[i] = -2;
        }

        //Jos rae on saavuttanut alarajan, tai siirtynyt X suunnassa pois näytöltä
        if (raePaikka[i].Y > 500 || raePaikka[i].X < 0 || raePaikka[i].X > 1100)
        {
            rae[i] = false;
            raeY[i] = 0;
        }

        //Lasketaan X
        raeX[i] += 1 * nopeus * k[i];

        //Lasketaan Y
        raeY[i] += 1 * nopeus;
    }
}

//Lokkien luonti ja liikutus
public void Lokit()
{
    //Tässä [i] indeksillä osoitetaan lokin numeroa, joka voi olla 2-4
    int rnd;
    for (int i = 0; i < 1 + taso; i++)
    {
        //Jos alku luodaan lokki
        if (alku[i] == true)
        {
            tormays[i] = false;

            lokkiY[i] = 20;
            lokkiX[i] = satLuku.Next(50, 1000);

            lokkiSuuntaY[i] = 1;

            rnd = satLuku.Next(1, 3);

            switch (rnd)
            {
                case 1: lokkiSuuntaX[i] = 1;
                    lokkiY[i] = 1;
                    break;
                case 2: lokkiSuuntaX[i] = -1;
                    lokkiSuuntaY[i] = 1;
                    break;
            }

            alku[i] = false;
        }
```

```
            if (lokkiPaikka[i].X < 1)
            {
                lokkiSuuntaX[i] *= -1;
                rnd = satLuku.Next(0, 2);
                if (rnd == 0)
                {
                    lokkiSuuntaY[i] *= -1;
                    lokkiX[i] += 10;
                }
            }

            if (lokkiPaikka[i].X > 1045)
            {
                lokkiSuuntaX[i] *= -1;
                rnd = satLuku.Next(0, 2);
                if (rnd == 0)
                {
                    lokkiSuuntaY[i] *= -1;
                    lokkiX[i] -= 10;

                }
            }

            if (lokkiPaikka[i].Y < 1)
            {
                lokkiSuuntaY[i] *= -1;
                rnd = satLuku.Next(0, 2);
                if (rnd == 1)
                {
                    lokkiSuuntaX[i] *= -1;
                    lokkiX[i] += 10;
                }
            }

            if (lokkiPaikka[i].Y > 500)
            {
                lokkiSuuntaY[i] *= -1;
                rnd = satLuku.Next(0, 2);
                if (rnd == 1)
                {
                    lokkiSuuntaX[i] *= -1;
                    lokkiX[i] -= 10;
                }
            }

            lokkiX[i] += lokkiSuuntaX[i] * nopeus;
            lokkiY[i] += lokkiSuuntaY[i] * nopeus;
        }
    }

    //Ammuksen liikuttaminen
    public void Ammus()
    {

        if (Keyboard.GetState().IsKeyDown(Keys.Right))
        {
            ammusSuuntaX = 10;
            ammusSuuntaY = 0;
        }
        if (Keyboard.GetState().IsKeyDown(Keys.Left))
        {
            ammusSuuntaY = 0;
            ammusSuuntaX = -10;
        }
        if (Keyboard.GetState().IsKeyDown(Keys.Up))
```

```
            {
                ammusSuuntaY = -10;
                ammusSuuntaX = 0;            }

            if (Keyboard.GetState().IsKeyDown(Keys.Down))
            {
                ammusSuuntaY = 10;
                ammusSuuntaX = 0;
            }
            if (Keyboard.GetState().IsKeyDown(Keys.Right) &&
Keyboard.GetState().IsKeyDown(Keys.Up))
            {
                ammusSuuntaX = 10;
                ammusSuuntaY = -10;
            }
            if (Keyboard.GetState().IsKeyDown(Keys.Right) &&
Keyboard.GetState().IsKeyDown(Keys.Down))
            {
                ammusSuuntaY = 10;
                ammusSuuntaX = 10;
            }

            if (Keyboard.GetState().IsKeyDown(Keys.Left) &&
Keyboard.GetState().IsKeyDown(Keys.Up))
            {
                ammusSuuntaX = -10;
                ammusSuuntaY = -10;
            }
            if (Keyboard.GetState().IsKeyDown(Keys.Left) &&
Keyboard.GetState().IsKeyDown(Keys.Down))
            {
                ammusSuuntaX = -10;
                ammusSuuntaY = 10;
            }
        }

        public void Perhonen()
        {
            //Liikutetaan ensin spritea sen suuntavektorin suuntaan
            paikka.X += nopeus2.X;
            paikka.Y += nopeus2.Y;

            //asetetaan nopeus
            int nX = satLuku.Next(1, 3);
            int nY = satLuku.Next(1, 3);

            //Mikäli perhonen on kääntymispisteessä, eli kulkenut
            //tarkoitetun matkan, arvotaan uusi suunta ja asetetaan uusi nopeus
            if (aloitus == 1 && laskuri2 >= perhonenRnd)
            {
                laskuri2 = 0;
                int snt = satLuku.Next(0, 8);
                switch (snt)
                {
                    case 0: nopeus2.X = nX;
                        nopeus2.Y = nY * -1;
                        break;
                    case 1: nopeus2.X = 0;
                        nopeus2.Y = nY * -1;
                        break;
                    case 2: nopeus2.X = nX;
                        nopeus2.Y = 0;
                        break;
                    case 3: nopeus2.X = nX;
                        nopeus2.Y = nY;
```

```
                break;
            case 4: nopeus2.X = 0;
                nopeus2.Y = nY;
                break;
            case 5: nopeus2.X = nX * -1;
                nopeus2.Y *= nY;
                break;
            case 6: nopeus2.X = nX * -1;
                nopeus2.Y = 0;
                break;
            case 7: nopeus2.X = nX * -1;
                nopeus2.Y = nY * -1;
                break;
        }
    }

    //Jos perhonen meinaa karata näytöltä, muutetaan suuntaa
    if (paikka.X < 10)
    {
        nopeus2.X *= -1;
        paikka.X += 10;
    }
    if (paikka.X >= 1090)
    {
        nopeus2.X *= -1;
        paikka.X -= 10;
    }
    if (paikka.Y < 10)
    {
        nopeus2.Y *= -1;
        paikka.Y += 10;
    }
    if (paikka.Y > 500)
    {
        nopeus2.Y *= -1;
        paikka.Y -= 10;
    }

    //Arvotaan pituus, kuinka pitkästi kerrallaan yhteen suuntaan liikutaan
    if (laskuri2 == 0) perhonenRnd = satLuku.Next(30, 60);
    laskuri2++;
}

public void Kotka()
{
    //Jos kotkaa ei ole, luodaan
    if (suuntaKotkaBool == false)
    {
        suuntaKotkaX = -50;
        suuntaKotkaY = alusY;
        suuntaKotkaBool = true;
        liikeMuutos = true;
    }

    if (liikeMuutos == true)
    {
        //Lasketaan kotkalle kulmakerroin niin, että se suuntautuu alusta kohti
        kulma = (alusY - suuntaKotkaY) / alusX;

        //Rajoitetaan että kulmakerroin on välillä -2  < k < 2
        if (kulma > 4) kulma = 4;
        if (kulma < -4) kulma = -4;

        liikeMuutos = false;
    }
```

```
    if (laskuri3 >= 20)
    {
        liikeMuutos = true;
        laskuri3 = 0;
    }
    else laskuri3++;

    if (paikkaKotka.X < 1100)
    {
        //Liikutetaan kotkaa sen suuntavektorin suuntaan
        suuntaKotkaX += 1.5 * nopeus;
        suuntaKotkaY += 4 * kulma * nopeus;
    }
    else suuntaKotkaBool = false;
}

public void Tahti()
{
    if (aika % 1800 == 0)
    {
        paikkaTahti.X = satLuku.Next(50, 1050);
        paikkaTahti.Y = 20;
        tahtiBool = true;
    }

    paikkaTahti.Y++;
    if (paikkaTahti.Y > 550)
    {
        paikkaTahti.Y = 1000;
        tahtiBool = false;
    }

}

public void Pisara()
{
    if ((aika % 1800) - 1200 == 0)
    {
        pisaraPaikka.X = satLuku.Next(50, 1050);
        pisaraPaikka.Y = 20;
        pisaraBool = true;
    }

    pisaraPaikka.Y++;
    if (pisaraPaikka.Y > 550)
    {
        pisaraPaikka.Y = 1000;
        pisaraBool = false;
    }

}

public void AmmusLisaus()
{
    if ((aika % 1800) - 600 == 0)
    {
        ammusLisausPaikka.X = satLuku.Next(50, 1050);
        ammusLisausPaikka.Y = 20;
        ammusLisaysBool = true;
    }

    ammusLisausPaikka.Y++;
    if (ammusLisausPaikka.Y > 550)
    {
        ammusLisausPaikka.Y = 1000;
```

```
                ammusLisaysBool = false;
            }

        }

        //Tarkistetaan törmääkö alus ja salama
        public bool SalamaTormaus()
        {
            Rectangle alusSuorakaide = new Rectangle((int)aluksenPaikka.X,
(int)aluksenPaikka.Y, 40, 40);
            Rectangle salamaSuorakaide = new Rectangle((int)salamaPaikka.X,
(int)salamaPaikka.Y, 20, 40);

            return alusSuorakaide.Intersects(salamaSuorakaide);
        }

        //Tarkistetaan törmätäänkö rakeisiin
        public bool RaeTormaus(int i)
        {
            Rectangle alusSuorakaide = new Rectangle((int)aluksenPaikka.X,
(int)aluksenPaikka.Y, 40, 40);
            Rectangle raeSuorakaide = new Rectangle((int)raePaikka[i].X,
(int)raePaikka[i].Y, 20, 20);

            return alusSuorakaide.Intersects(raeSuorakaide);
        }

        //Tarkistetaan törmätäänkö lokkeihin
        public bool LokkiTormaus(int i)
        {
            Rectangle alusSuorakaide = new Rectangle((int)aluksenPaikka.X,
(int)aluksenPaikka.Y, 40, 40);
            Rectangle lokkiSuorakaide = new Rectangle((int)lokkiPaikka[i].X,
(int)lokkiPaikka[i].Y, 40, 20);

            return alusSuorakaide.Intersects(lokkiSuorakaide);
        }

        //Tarkistetaan törmääkö ammus ja lokki
        public bool AmmusTormaus(int i)
        {
            Rectangle ammusSuorakaide = new Rectangle((int)ammusPaikka.X,
(int)ammusPaikka.Y, 10, 10);
            Rectangle lokkiSuorakaide = new Rectangle((int)lokkiPaikka[i].X,
(int)lokkiPaikka[i].Y, 40, 20);

            return ammusSuorakaide.Intersects(lokkiSuorakaide);
        }

        //Tarkistetaan perhosen ja aluksen törmäys
        public bool PerhonenTormaus()
        {
            Rectangle perhonenSuorakaide = new Rectangle((int)paikka.X, (int)paikka.Y, 30,
30);
            Rectangle alusSuorakaide = new Rectangle((int)aluksenPaikka.X,
(int)aluksenPaikka.Y, 40, 40);

            return perhonenSuorakaide.Intersects(alusSuorakaide);
        }

        //Tarkistetaan perhosen ja ammuksen törmäys
        public bool PerhonenAmmus()
        {
            Rectangle perhonenSuorakaide = new Rectangle((int)paikka.X, (int)paikka.Y, 30,
30);
```

```
            Rectangle ammusSuorakaide = new Rectangle((int)ammusPaikka.X,
(int)ammusPaikka.Y, 10, 10);

            return perhonenSuorakaide.Intersects(ammusSuorakaide);
        }

        //Tarkistetaan kotkan ja aluksen törmäys
        public bool KotkaAlus()
        {
            Rectangle kotkaSuorakaide = new Rectangle((int)suuntaKotkaX, (int)suuntaKotkaY,
40, 20);
            Rectangle alusSuorakaide = new Rectangle((int)aluksenPaikka.X,
(int)aluksenPaikka.Y, 40, 40);

            return kotkaSuorakaide.Intersects(alusSuorakaide);
        }

        //Tarkistetaan kotkan ja ammuksen törmäys
        public bool KotkaAmmus()
        {
            Rectangle kotkaSuorakaide = new Rectangle((int)suuntaKotkaX, (int)suuntaKotkaY,
40, 20);
            Rectangle ammusSuorakaide = new Rectangle((int)ammusPaikka.X,
(int)ammusPaikka.Y, 10, 10);

            return kotkaSuorakaide.Intersects(ammusSuorakaide);
        }

        //Tarkistetaan kotkan ja aluksen törmäys
        public bool TahtiAlus()
        {
            Rectangle tahtiSuorakaide = new Rectangle((int)paikkaTahti.X,
(int)paikkaTahti.Y, 20, 20);
            Rectangle alusSuorakaide = new Rectangle((int)aluksenPaikka.X,
(int)aluksenPaikka.Y, 40, 40);

            return tahtiSuorakaide.Intersects(alusSuorakaide);
        }

        //Tarkistetaan kotkan ja aluksen törmäys
        public bool AmmusLisaysAlus()
        {
            Rectangle ammusLisaysSuorakaide = new Rectangle((int)ammusLisausPaikka.X,
(int)ammusLisausPaikka.Y, 20, 20);
            Rectangle alusSuorakaide = new Rectangle((int)aluksenPaikka.X,
(int)aluksenPaikka.Y, 40, 40);

            return ammusLisaysSuorakaide.Intersects(alusSuorakaide);
        }

        //Tarkistetaan kotkan ja aluksen törmäys
        public bool PisaraAlus()
        {
            Rectangle pisaraSuorakaide = new Rectangle((int)pisaraPaikka.X,
(int)pisaraPaikka.Y, 20, 20);
            Rectangle alusSuorakaide = new Rectangle((int)aluksenPaikka.X,
(int)aluksenPaikka.Y, 40, 40);

            return pisaraSuorakaide.Intersects(alusSuorakaide);
        }

        public void Tyhjennys()
        {
            salama = false;
            salamaY = 0;
```

```
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            rae[j] = false;
            raeY[i] = 0;
        }
    }
    for (int i = 0; i < 1 + taso; i++)
    {
        for (int j = 0; j < 1 + taso; j++)
        {
            alku[j] = true;
            Lokit();
        }
        suuntaKotkaBool = false;
        paikkaKotka.X = -50;
    }
    alusX = 550;
    alusY = 350;
    PerhosenSiirto();
}

public void PerhosenSiirto()
{
    /*//Siirretään perhonen näytön toiseen laitaan kuin alus
    if (aluksenPaikka.X < 500) paikka.X = 900;
    else paikka.X = 200;*/
    paikka.X = 400;
    paikka.Y = 200;
}


public string Kirjoita()
{
    if (Keyboard.GetState().IsKeyDown(Keys.A))
    {

        if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
            || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'A';
        else nimi[pituus] = 'a';
        pituus++;
    }
    if (Keyboard.GetState().IsKeyDown(Keys.B))
    {
        if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
            || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'B';
        else nimi[pituus] = 'b';
        pituus++;
    }
    if (Keyboard.GetState().IsKeyDown(Keys.C))
    {
        if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
            || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'C';
        else nimi[pituus] = 'c';
        pituus++;
    }
    if (Keyboard.GetState().IsKeyDown(Keys.D))
    {
        if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
            || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'D';
        else nimi[pituus] = 'd';
        pituus++;
    }
    if (Keyboard.GetState().IsKeyDown(Keys.E))
```

```
{
    if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
        || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'E';
    else nimi[pituus] = 'e';
    pituus++;
}
if (Keyboard.GetState().IsKeyDown(Keys.F))
{
    if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
        || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'F';
    else nimi[pituus] = 'f';
    pituus++;
} if (Keyboard.GetState().IsKeyDown(Keys.G))
{
    if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
        || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'G';
    else nimi[pituus] = 'g';
    pituus++;
}
if (Keyboard.GetState().IsKeyDown(Keys.H))
{
    if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
        || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'H';
    else nimi[pituus] = 'h';
    pituus++;
}
if (Keyboard.GetState().IsKeyDown(Keys.I))
{
    if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
        || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'I';
    else nimi[pituus] = 'i';
    pituus++;
}
if (Keyboard.GetState().IsKeyDown(Keys.J))
{
    if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
        || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'J';
    else nimi[pituus] = 'j';
    pituus++;
}
if (Keyboard.GetState().IsKeyDown(Keys.K))
{
    if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
        || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'K';
    else nimi[pituus] = 'k';
    pituus++;
}
if (Keyboard.GetState().IsKeyDown(Keys.L))
{
    if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
        || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'L';
    else nimi[pituus] = 'l';
    pituus++;
}
if (Keyboard.GetState().IsKeyDown(Keys.M))
{
    if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
        || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'M';
    else nimi[pituus] = 'm';
    pituus++;
}
if (Keyboard.GetState().IsKeyDown(Keys.N))
{
    if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
        || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'N';
```

```
        else nimi[pituus] = 'n';
        pituus++;
    }
    if (Keyboard.GetState().IsKeyDown(Keys.O))
    {
        if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
            || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'O';
        else nimi[pituus] = 'o';
        pituus++;
    }
    if (Keyboard.GetState().IsKeyDown(Keys.P))
    {
        if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
            || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'P';
        else nimi[pituus] = 'p';
        pituus++;
    }
    if (Keyboard.GetState().IsKeyDown(Keys.Q))
    {
        if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
            || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'Q';
        else nimi[pituus] = 'q';
        pituus++;
    }
    if (Keyboard.GetState().IsKeyDown(Keys.R))
    {
        if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
            || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'R';
        else nimi[pituus] = 'r';
        pituus++;
    }
    if (Keyboard.GetState().IsKeyDown(Keys.S))
    {
        if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
            || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'S';
        else nimi[pituus] = 's';
        pituus++;
    }
    if (Keyboard.GetState().IsKeyDown(Keys.T))
    {
        if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
            || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'T';
        else nimi[pituus] = 't';
        pituus++;
    }

    if (Keyboard.GetState().IsKeyDown(Keys.U))
    {
        if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
            || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'U';
        else nimi[pituus] = 'u';
        pituus++;
    }
    if (Keyboard.GetState().IsKeyDown(Keys.V))
    {
        if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
            || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'V';
        else nimi[pituus] = 'v';
        pituus++;
    }
    if (Keyboard.GetState().IsKeyDown(Keys.W))
    {
        if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
            || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'W';
        else nimi[pituus] = 'w';
```

```
        pituus++;
    }
    if (Keyboard.GetState().IsKeyDown(Keys.X))
    {
        if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
            || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'X';
        else nimi[pituus] = 'x';
        pituus++;
    }
    if (Keyboard.GetState().IsKeyDown(Keys.Y))
    {
        if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
            || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'Y';
        else nimi[pituus] = 'y';
        pituus++;
    }
    if (Keyboard.GetState().IsKeyDown(Keys.Z))
    {
        if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
            || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'Z';
        else nimi[pituus] = 'z';
        pituus++;
    }
    if (Keyboard.GetState().IsKeyDown(Keys.F1))
    {
        if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
            || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'Å';
        else nimi[pituus] = 'å';
        pituus++;
    }
    if (Keyboard.GetState().IsKeyDown(Keys.F2))
    {
        if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
            || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'Ä';
        else nimi[pituus] = 'ä';
        pituus++;
    }
    if (Keyboard.GetState().IsKeyDown(Keys.F3))
    {
        if (Keyboard.GetState().IsKeyDown(Keys.LeftShift)
            || Keyboard.GetState().IsKeyDown(Keys.RightShift)) nimi[pituus] = 'Ö';
        else nimi[pituus] = 'ö';
        pituus++;
    }
    if (Keyboard.GetState().IsKeyDown(Keys.D1))
    {
        nimi[pituus] = '1';
        pituus++;
    }
    if (Keyboard.GetState().IsKeyDown(Keys.D2))
    {
        nimi[pituus] = '2';
        pituus++;
    }
    if (Keyboard.GetState().IsKeyDown(Keys.D3))
    {
        nimi[pituus] = '3';
        pituus++;
    }
    if (Keyboard.GetState().IsKeyDown(Keys.D4))
    {
        nimi[pituus] = '4';
        pituus++;
    }
    if (Keyboard.GetState().IsKeyDown(Keys.D5))
```

```
        {
            nimi[pituus] = '5';
            pituus++;
        }
        if (Keyboard.GetState().IsKeyDown(Keys.D6))
        {
            nimi[pituus] = '6';
            pituus++;
        }
        if (Keyboard.GetState().IsKeyDown(Keys.D7))
        {
            nimi[pituus] = '7';
            pituus++;
        }
        if (Keyboard.GetState().IsKeyDown(Keys.D8))
        {
            nimi[pituus] = '8';
            pituus++;
        }
        if (Keyboard.GetState().IsKeyDown(Keys.D9))
        {
            nimi[pituus] = '9';
            pituus++;
        }
        if (Keyboard.GetState().IsKeyDown(Keys.D0))
        {
            nimi[pituus] = '0';
            pituus++;
        }
        if (Keyboard.GetState().IsKeyDown(Keys.Space))
        {
            nimi[pituus] = ' ';
            pituus++;
        }

        sana = "";
        for (int i = 0; i < pituus; i++) sana += nimi[i];

        return sana;
    }

    private string webPost(string _URI, string _postString)
    {
        const string REQUEST_METHOD_POST = "POST";
        const string CONTENT_TYPE = "application/x-www-form-urlencoded";
        Stream dataStream = null;
        StreamReader reader = null;
        WebResponse response = null;
        string responseString = null;
        // Create a request using a URL that can receive a post.
        WebRequest request = WebRequest.Create(_URI);
        // Set the Method property of the request to POST.
        request.Method = REQUEST_METHOD_POST;
        // Create POST data and convert it to a byte array.
        string postData = _postString;

        // Set the ContentType property of the WebRequest.
        request.ContentType = CONTENT_TYPE;
        // Set the ContentLength property of the WebRequest.

        byte[] byteArray = Encoding.UTF8.GetBytes(postData);
        request.ContentLength = byteArray.Length;

        try
        {
```

```
                // Get the request stream.
                dataStream = request.GetRequestStream();
                // Write the data to the request stream.
                dataStream.Write(byteArray, 0, byteArray.Length);
                // Close the Stream object.
                dataStream.Close();
                // Get the response.

                response = request.GetResponse();
                // Display the status.
                Console.WriteLine(((HttpWebResponse)response).StatusDescription);
                // Get the stream containing content returned by the server.
                dataStream = response.GetResponseStream();
                // Open the stream using a StreamReader for easy access.
                reader = new StreamReader(dataStream);
                // Read the content.
                responseString = reader.ReadToEnd();
                // Display the content.
                //Console.WriteLine(responseFromServer);

                // Clean up the streams.
                if (reader != null) reader.Close();
                if (dataStream != null) dataStream.Close();
                if (response != null) response.Close();
            }

            catch
            {
                yhtvirhe = true;
            }
            return responseString;
        }

        public string getScores()
        {
            try
            {
                return webPost("http://www.pelilakka.com/P2/requestscores.php", postString);
            }
            catch
            {
                return "";
            }
        }

        public string sendScore(string ennatukset)
        {
            try
            {
                string postString = "&Hash=" + "kSL8Eb4t" + "&Ennatukset=" + ennatukset;
                return webPost("http://www.pelilakka.com/P2/newscore.php", postString);

            }
            catch
            {
                return "";
            }
        }
    }
}
```