XNA grafiikka laajennus opas

Paavo Räisänen

www.ohjelmoimaan.net

Tämän oppaan lähdekoodit ovat ladattavissa
näiden sivujen ”Ladattavat” osiossa.

1: Johdanto

Tässä oppaassa neuvotaan, kuinka XNA:lla pystyy piirtämään viivan, ja kuinka tekstuurin pixelien väri voidaan lukea. Oppaan ymmärtäminen vaatii kohtuullista XNA:n tuntemusta. Käytetty XNA versio on XNA 4.0.

2: tekstuurin pixelin värin lukeminen

Itse lukuasian ydin on tässä. try -catch on sitä varten, että käyttäjä klikkaa pikseliä tekstuurin ulkopuolella. tekstuuri on tekstuuri, jonka pisteen värin haluat lukea.

```
Texture2D tekstuuri;
Color[] taustanvari = new Color[1];
Rectangle pistenelio = new Rectangle(Mouse.GetState().X, Mouse.GetState().Y, 1, 1);

                try
                {
                    tekstuuri.GetData<Color>(
                        0,
                        pistenelio,
                        taustanvari,
                        0,
                        1);
                }
                catch { }
```

Saatua dataa luetaan näin:

```
Color[] taustanvari = new Color[1];
Color vari = new Color(0, 65, 255);
if (taustanvari[0] == vari)
```

Koko Game1.cs koodi on seuraava. Ohjelma kokonaisuudessaan on ladattavissa paketoituna näiden www.ohjelmoimaan.net sivujen ”oppaat” sivuilta. Klikkailemalla ohjelmassa näyttöä, saat hiiren alla olevista väriarvoista tietoa.

```
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 LuePisteVari
{
    /// <summary>
    /// This is the main type for your game
    /// </summary>
    public class Game1 : Microsoft.Xna.Framework.Game
    {
        GraphicsDeviceManager graphics;
        SpriteBatch spriteBatch;
        Texture2D tekstuuri;
        SpriteFont pisteFontti;
        Color[] taustanvari = new Color[1];

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


            graphics.PreferredBackBufferWidth = 1200;
            graphics.PreferredBackBufferHeight = 800;
#if    !DEBUG
                graphics.IsFullScreen           true;
#endif
        }

        /// <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);

            tekstuuri = Content.Load<Texture2D>(@"Kuvat/Sprite-0003");
            pisteFontti = Content.Load<SpriteFont>(@"Fontit\arial");

        }
```

```
/// <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)
{
    if (Mouse.GetState().LeftButton == ButtonState.Pressed &&
        // If the left button is pressed
     Mouse.GetState().X > 0 && Mouse.GetState().Y > 0 &&
        // and we are inside the game window
     (Mouse.GetState().X <
     GraphicsDevice.PresentationParameters.BackBufferWidth) &&
    (Mouse.GetState().Y <
     GraphicsDevice.PresentationParameters.BackBufferHeight))
    {
        Rectangle pistenelio =
        new Rectangle(Mouse.GetState().X, Mouse.GetState().Y, 1, 1);

        try
        {
            tekstuuri.GetData<Color>(
                0,
                pistenelio,
                taustanvari,
                0,
                1);
        }
        catch { }


    }
    // 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.Green);

    spriteBatch.Begin();
    spriteBatch.Draw(tekstuuri, Vector2.Zero, null, Color.White, 0, Vector2.Zero, 1,
     SpriteEffects.None, 0);
    Color vari = new Color(158, 125, 255);
    Color vari2 = new Color(0, 65, 255);
    Color vari3 = new Color(0, 125, 255);
```

```
            Color vari4 = new Color(0, 190, 255);
            Color vari5 = new Color(0, 255, 0);

            spriteBatch.DrawString(pisteFontti, "Väri: " + taustanvari[0],
             new Vector2(10, 10), Color.DarkBlue, 0, Vector2.Zero,
             1, SpriteEffects.None, 1);

            if (taustanvari[0] == vari)
            {
                spriteBatch.DrawString(pisteFontti, "Väri on violetti.",
                 new Vector2(10, 50), Color.DarkBlue, 0, Vector2.Zero,
                 1, SpriteEffects.None, 1);
            }

            if (taustanvari[0] == vari2)
            {
                spriteBatch.DrawString(pisteFontti, "Väri on tumman sininen.",
                 new Vector2(10, 50), Color.DarkBlue, 0, Vector2.Zero,
                 1, SpriteEffects.None, 1);
            }

            if (taustanvari[0] == vari3)
            {
                spriteBatch.DrawString(pisteFontti, "Väri on sininen.",
                 new Vector2(10, 50), Color.DarkBlue, 0, Vector2.Zero,
                 1, SpriteEffects.None, 1);
            }

            if (taustanvari[0] == vari4)
            {
                spriteBatch.DrawString(pisteFontti, "Väri on vaalean sininen. ",
                 new Vector2(10, 50), Color.DarkBlue, 0, Vector2.Zero,
                 1, SpriteEffects.None, 1);
            }

            if (taustanvari[0] == vari5)
            {
                spriteBatch.DrawString(pisteFontti, "Väri on vihreä. ",
                 new Vector2(10, 50), Color.DarkBlue, 0, Vector2.Zero,
                 1, SpriteEffects.None, 1);
            }
            spriteBatch.End();

            // TODO: Add your drawing code here

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

3: 2D viivan piirtäminen XNA :lla

2D viivan piirto XNA :lla ei ole ihan yksinkertaista, kun XNA:sta puuttuu tarvittava metodi. Olen tehnyt siihen metodin, jolla piirto käy näppärästi, eikä sinun tarvitse välttämättä ymmärtää siitä paljoakaan. Oikeastaan sen käytön kannalta riittaa, kun osaa syöttää oikeat parametrit.

Metodin määrittely on seuraava:

```
public void ViivanPiirto(int x, int y, int x2, int y2, int paksuus, double tarkkuus, Color
vari)
```

x,y,x2,y2 ovat koordinantteja, mistä mihin viiva piiretään. Kohtaan paksuus tulee viivan paksuus, joka useinmiten on 1, mutta voi olla vaikka 100:kin.
tarkkuus muuttujaan annetaan tarkkuus millä piirretään. Vinoja viivoja piirrettäessä kannattaa käyttää suurta tarkkuutta, esim. 100, mutta vaaka- ja pystysuoriin viivoihin voi hyvin antaa arvoksi vaikka 1.

vari muuttujaan tulee viivan väri. Värin voi määritellä joko:

```
Color vari = Color.Yellow;
```
tai rgb arvoina `Color vari = new Color(255, 255, 0);`

Koko Game1.cs koodi on alla.

```
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 Viiva
{
    /// <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";

            IsMouseVisible = true;

            graphics.PreferredBackBufferWidth = 1000;
            graphics.PreferredBackBufferHeight = 800;
#if    !DEBUG
                graphics.IsFullScreen          true;
#endif
        }

        /// <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)
{
    // TODO: Add your update logic here
    // Allows the game to exit
    if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
        this.Exit();

    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);
    int x = 500;
    int y = 400;
    int x2 = 0;
    int y2 = 0;
    if (Mouse.GetState().LeftButton == ButtonState.Pressed)

    {
        x2 = Mouse.GetState().X;
        y2 = Mouse.GetState().Y;
        Color vari = Color.Yellow;
        // tai näin rgb: Color vari = new Color(255,255,0);
```

```
            ViivanPiirto(x, y, x2, y2, 1, 100, vari);

        }

        base.Draw(gameTime);
    }

    public void ViivanPiirto(int x, int y, int x2, int y2, int paksuus, double tarkkuus,
     Color vari)
    {
        //tarkkuus esim. 100
        spriteBatch.Begin();

        double i;
        double x3 = 0;
        double y3 = 0;
        tarkkuus = 1 / tarkkuus;
        Texture2D viiva = new Texture2D(GraphicsDevice, 1, 1, false,
         SurfaceFormat.Color);
        Int32[] pixel = { 0x0ff0f0f0 };
        viiva.SetData<Int32>(pixel, 0, viiva.Width * viiva.Height);
        if (x != x2)
        {
            double k = (double)(y2 - y) / (double)(x2 - x);
            i = x;
            if (x2 >= x)
            {
                while (i < x2)
                {

                    Rectangle nelio = new Rectangle(x + (int)x3, y + (int)y3, paksuus,
                     paksuus);
                    spriteBatch.Draw(viiva, nelio, vari);
                    i = i + tarkkuus;
                    x3 = x3 + tarkkuus;
                    y3 = k * x3;
                }
            }

            if (x2 < x)
            {
                i = x2;
                while (i < x)
                {

                    Rectangle nelio = new Rectangle(x - (int)x3, y - (int)y3, paksuus,
                     paksuus);
                    spriteBatch.Draw(viiva, nelio, vari);
                    i = i + tarkkuus;
                    x3 = x3 + tarkkuus;
                    y3 = k * x3;
                }
            }
        }

        else if (x == x2)
        {
          if ( y >= y2 )
          {
              Rectangle nelio = new Rectangle(x, y2, paksuus, (y + paksuus) - y2);
                  spriteBatch.Draw(viiva, nelio, vari);
          }

          if (y < y2)
          {
```

```
                Rectangle nelio = new Rectangle(x, y, paksuus, (y2 + paksuus) - y);
                spriteBatch.Draw(viiva, nelio, vari);
            }

        }

            spriteBatch.End();

    }
}
}
```

Tämä ohjelma piirtää viivan (paksuus, paksuus) kokoisina neliöinä. Kun klikkailet eri puolille näyttöä, ohjelma piirtää viivan alkaen ikkunan keskipisteestä. Sinun on helppo muokata ohjelmasta vaikka sellainen, että voit klikkaamalla antaa sekä alku, että loppupisteet. Yleensähän ne vain annetaan ohjelmassa muuttamalla koodia. Huom. tämä on vain yksi mahdollinen tapa toteuttaa viivan piirto. Esim. vectorien avulla pystyy tekemään korvaavan, mutta tekemäni ratkaisun ei pitäisi olla huono.