الدرس الثالث

أهلا بكم في الدرس الثالث من سلسلة دروس تعلم ال 3D Xna السلسلة الأولى، في هذا الدرس سوف نقوم برسم المثلث الأول.

هذا الدرس سوف يغطي الأساسيات في الرسم. في البداية هناك بضعة اشياء يجب عليك معرفتها.

كل عنصر يتم رسمه في العالم الثلاثي الأبعاد، يتم رسمه بإستخدام المثلثات. حتى الكرات يمكن وصفها بإستخدام المثلثات، إذا كان هناك ما يكفي منهم. مباشرة، يتم وصف المثلث بإستخدام ثلاث رؤوس. كل رأس معرفة بإستخدام متجه "Vector"، كل متجه يتم وصفه بإستخدام ثلاث إحداثيات هي X,Y و Z. مع ذلك مجرد معرفة إحداثيات الرأس قد لايكون كافيا. على سبيل المثال، ربما يلزمك أن تقوم بتحديد لون معين لرأس. في هذه الحالة يمكن وصف الرأس "Vertex" (الرؤوس "Vertices") بـ: مجموعة من الخصائص لرأس معين بما فيها الموقع "Position" ، اللون و ما إلى ذلك.

ال Xna لديها تركيب "Structure" يلائم تماما الحاجة لتخزين معلومات الرأس: التركيب VertexPositionColor. رأس من هذا النوع بإمكانه تخزين موقع و لون نقطة، و هي الأفضل لكي نبدأ بها. من أجل تعريف مثلث، سوف نحتاج إلى ثلاثة من هذه الرؤوس، الذين سوف نقوم بتخزينهم في مصفوفة. لذا دعنا نقوم بتعريف هذا المتغير في أعلى الصنف"Class" :

```
VertexPositionColor[] vertices;
```

بعدها، سوف نقوم بإضافة دالة صغيرة للكود الخاص بنا، SetUpVertices، التي سوف تقوم بتعبئة المصفوفة بثلاثة رؤوس:

```
private void SetUpVertices()
{
    vertices = new VertexPositionColor[3];

    vertices[0].Position = new Vector3(-0.5f, -0.5f, 0f);
    vertices[0].Color = Color.Red;
    vertices[1].Position = new Vector3(0f, 0.5f, 0f);
    vertices[1].Color = Color.Green;
    vertices[2].Position = new Vector3(0.5f, -0.5f, 0f);
    vertices[2].Color = Color.Yellow;
}
```

تم تجهيز المصفوفة لكي تحتوي على 3 رؤوس، بعد ذلك يتم تعبئتها. حتى اللحظة، نحن نستخدم إحداثيات نسبة إلى الشاشة: نقطة الأصل (0,0,0) سوف تكون منتصف الشاشة لدينا، النقطة (-1,-1,0) تمثل الركن الأسفل الأيسر للشاشة، و النقطة (1,1,0) هي الركن العلوي الأيمن. لذا المثال في الأعلى، النقطة الأولى هي نصف الطريق إلى النقطة السفلى اليسرى للشاشة، و النقطة الثانية هي نصف الطريق للنقطة العلوية لمنتصف الشاشة. (بما أنها حقيقة ليست إحداثيات ثلاثية أبعاد، ليس هناك حاجة لتحويلها إلى إحداثيات ثنائية أبعاد. حيث نسمي هذه التقنية بـ : ('Pretransformed').

حرف 'f' الذي يتبع بعض الأرقام تدل على أن هذه القيم عبارة عن أعداد حقيقية "Floats"، و هو التنسيق المفضل عند التعامل مع ال Xna. قمنا أيضا بإعطاء كل من الرؤوس لون مختلف.

في هذه اللحظة قمنا بتخزين معلومات الموقع و اللون لثلاث رؤوس في داخل مصفوفة. عندما نرشد ال Xna لكي ترسم المثلث بناء على هذه البيانات، سوف تقوم بوضع كل هذه البيانات في سلسلة بايتات "Byte Stream" و من ثم إرسالها إلى بطاقة الرسوميات. بطاقة الرسوميات تستقبل سلسلة البايتات، ولكنها لا تعلم ما هي محتوياتها! لذا علينا أولا أن نمرر تعريف النقطة VertexDeclaration: هذا الكائن يخبر جهاز الرسوميات عن نوع الرؤوس الذي يجب أن يتوقعه.

الكائن **VertexDeclaration** مهم جدا، لإنك سوف تحتاجه دائما قبل رسم أي مثلث على الشاشة. بما أن ال VertexDeclaration لن يتغير في المشروع، يجب عليك أن تضيفه كمتغير إلى الكود و تقوم بعمل إستهلال له لمرة واحدة.

(علاوة على ذلك إنشاء هذ المتغير في كل إطار "Frame" سوف يؤدي إلى إنهيار إطار عمل ال Net.المضغوطة على ال Xbox360) لذا قم بإضافة هذا المتغير في أعلى الصنف الخاص بنا:

```
VertexDeclaration myVertexDeclaration;
```

وقم بتجهيزه في داخل الدالة SetUpVertices. هذا هو المكان الأفضل، بما أن ال VertexDeclaration ينتمي إلى نوع الرؤس التي قمت بتعبئتها في المصفوفة.

```
myVertexDeclaration = new VertexDeclaration(device,
VertexPositionColor.VertexElements);
```

هذا التعريف يجب أن يتم تمريره إلى بطاقة الرسوميات من أجل إعلامها بأن هناك رؤوس من النوع VertexPositionColor قادمة في الطريق، و هو ما سوف نفعله في داخل الدالة Draw لاحقا.

ما زلنا بحاجة إلى إستدعاء الدالة SetUpVertices. قم بإستدعائها في نهاية الدالة LoadContent:

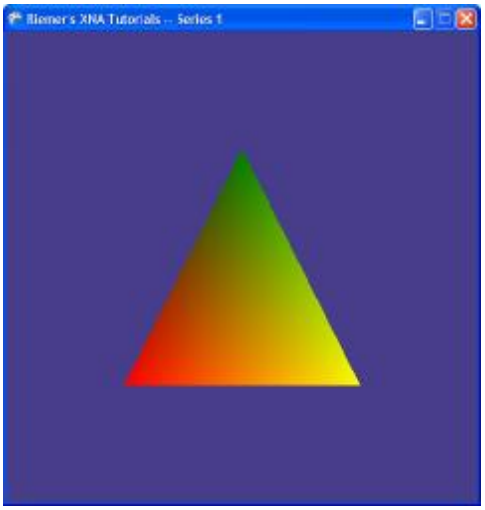
```
SetUpVertices();
```

كل ما نحتاج لعمله الآن هو أن نخبر جهاز الرسوميات أن يرسم المثلث! إذهب إلى الدالة Draw، المكان الذي يتوجب علينا فيه رسم المثلث هو بين الإستدعائين للدالتين pass.Begin و pass.End:

```
device.VertexDeclaration = myVertexDeclaration;
device.DrawUserPrimitives(PrimitiveType.TriangleList, vertices, 0, 1);
```

السطر الأول يرسل ال VertexDeclaration الذي تم تعريفه سابقا، بينما السطر الثاني يقوم برسم المثلث فعليا: نحن نريد رسم مثلث واحد من الرؤوس المخزنة في المصفوفة، بدأ بالرأس رقم 0. ال TriangleList تعني أن مصفوفة الرؤوس تحتوي على قائمة بالمثلثات (في حالتنا هنا، قائمة بمثلث واحد فقط). إذا أردت رسم 4 مثلثات، سوف تحتاج لمصفوفة تحتوي على 12 رأس. إحتمالية أخرى هي TriangleStrip، حيث بالإمكان عملها بشكل أسرع، ولكنها مفيده فقط في حال كانت المثلثات متصلة ببعضها البعض.

تشغيل هذا الكود يجب أن يعطي نتيجة مثل الموضحة في الصورة:



هذا كل شيئ في هذا الدرس! تشغيل هذا الكود سوف يعطينا مثلث ملون على خلفية زرقاء. خذ حريتك في التغيير على الإحداثيات و الألوان.

بإمكانك أن تجرب حل هذه التمارين لممارسة ما تعلمته:

- إجعل الزاوية السفلية اليسرى للمثلث يتصادم مع الزاوية السفليه اليسرى للشاشة.
- قم برسم مثلثين لتغطية كامل الشاشة.

الكود كامل حتى الآن:

```
using System;
using System.Collections.Generic;
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.Net;
using Microsoft.Xna.Framework.Storage;

namespace XNAtutorial
{
    public class Game1 : Microsoft.Xna.Framework.Game
    {
        GraphicsDeviceManager graphics;
        SpriteBatch spriteBatch;
        GraphicsDevice device;
        Effect effect;
        VertexPositionColor[] vertices;
        VertexDeclaration myVertexDeclaration;

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

        protected override void Initialize()
        {
            graphics.PreferredBackBufferWidth = 500;
            graphics.PreferredBackBufferHeight = 500;
            graphics.IsFullScreen = false;
            graphics.ApplyChanges();
            Window.Title = "Riemer's XNA Tutorials -- Series 1";

            base.Initialize();
        }

        private void SetUpVertices()
        {
            vertices = new VertexPositionColor[3];

            vertices[0].Position = new Vector3(-0.5f, -0.5f, 0f);
            vertices[0].Color = Color.Red;
            vertices[1].Position = new Vector3(0, 0.5f, 0f);
            vertices[1].Color = Color.Green;
            vertices[2].Position = new Vector3(0.5f, -0.5f, 0f);
```

```
            vertices[2].Color = Color.Yellow;

            myVertexDeclaration = new VertexDeclaration(device,
VertexPositionColor.VertexElements);
        }

        protected override void LoadContent()
        {
            device = graphics.GraphicsDevice;
            spriteBatch = new SpriteBatch(GraphicsDevice);

           effect = Content.Load<Effect> ("effects");


            SetUpVertices();
        }

        protected override void UnloadContent()
        {
        }

        protected override void Update(GameTime gameTime)
        {
            if (GamePad.GetState(PlayerIndex.One).Buttons.Back ==
ButtonState.Pressed)
                this.Exit();

            base.Update(gameTime);
        }

        protected override void Draw(GameTime gameTime)
        {
            device.Clear(Color.DarkSlateBlue);

            effect.CurrentTechnique = effect.Techniques["Pretransformed"];
            effect.Begin();
            foreach (EffectPass pass in effect.CurrentTechnique.Passes)
            {
                pass.Begin();

                device.VertexDeclaration = myVertexDeclaration;
                device.DrawUserPrimitives(PrimitiveType.TriangleList,
vertices, 0, 1);

                pass.End();
            }
            effect.End();

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