

الدرس الثاني

أهلاً بكم في الدرس الثاني من سلسلة دروس تعلم 3D Xna (السلسلة الثانية)، في هذا الدرس سوف نتحدث عن الخامات.

حتى اللحظة، الطريقة الوحيدة التي إستخدمناها لتلوين المشهد، هي من خلال تعريف رؤوس منفصلة بحيث يحمل كل رأس لون مختلف. بالتأكيد، ليست هذه الطريقة التي تم صنع الألعاب الحديثة الضخمة بها. XNA تدعم طريقة فعالة جداً لإضافة الألوان و الصور إلى المشهد: بإمكانك ببساطة وضع صورة على مثلث. يطلق على هذه الصور إسم الخامات "Textures".

كمثال أولي، سوف نقوم برسم مثلث واحد بسيط، وتغطيته بخامة. بإمكانك الحصول على خامة بسيطة من الرابط [هنا](#) (بإمكانك تنزيلها من خلال الضغط بالزر الأيمن للماوس على الرابط و إختيار حفظ الصورة بإسم...). بعدها قم بإستيراد الصورة إلى مجلد ال Content في ال Solution Explorer، تماماً كما فعلنا مع ملف التأثير.

عندما تضغط على الملف في خانة ال Content في ال Solution Explorer، سيكون بإمكانك مشاهدة إسم هذا الملف بإسم 'riemersttexture'، وذلك في مربع الخصائص في الجهة السفلى اليمنى للشاشة، بإمكانك تغيير هذا الاسم كما تحب، لكن دعه كما هو الآن.

في الكود، سوف نقوم بإضافة متغير جديد لكي يحمل صورة الخامة. قم بإضافة السطر التالي في أعلى الكود:

```
Texture2D texture;
```

الآن جد الدالة LoadContent في الكود. وقم بإضافة السطر التالي بعد السطر الذي نقوم فيه بإستيراد ملف التأثير:

```
texture = Content.Load<Texture2D> ("riemersttexture");
```

يقوم هذ السطر بربط الخامة (الصورة) الذي قمنا بإضافتها إلى المشروع مع المتغير الذي قمنا بتعريفه لتمثيل الخامة.

بعد أن قمنا بتحميل الخامة إلى مشروع ال Xna، حان الوقت لتعريف 3 رؤوس، حيث سنقوم بتخزينهم في مصفوفة. بما أن الرؤوس سيتحتّم عليها تخزين الموقع الثلاثي الأبعاد و إحداثيات الخامة (مشروحة في الأسفل)، إذن سوف يكون تنسيق الرؤوس هو VertexPositionTexture، لذا قم بتعريف هذا المتغير في أعلى الكود. كما سيلزمنا أيضاً "VertexDeclaration" جديد، لأنه يتحتّم علينا إخبار بطاقة الرسومات بطبيعة البيانات التي نخزنها في كل رأس:

```
VertexPositionTexture[] vertices;  
VertexDeclaration texturedVertexDeclaration;
```

بعدها سوف نقوم بتعريف الرؤوس الثلاثة التي تمثل المثلث المطلوب، سنقوم بذلك في دالة جديدة بإسم SetupVertices سنقوم بإنشائها:

```
private void SetupVertices()  
{  
    vertices = new VertexPositionTexture[3];  
    vertices[0].Position = new Vector3(-10f, 10f, 0f);
```

```

vertices[0].TextureCoordinate.X = 0;
vertices[0].TextureCoordinate.Y = 0;

vertices[1].Position = new Vector3(10f, -10f, 0f);
vertices[1].TextureCoordinate.X = 1;
vertices[1].TextureCoordinate.Y = 1;

vertices[2].Position = new Vector3(-10f, -10f, 0f);
vertices[2].TextureCoordinate.X = 0;
vertices[2].TextureCoordinate.Y = 1;

texturedVertexDeclaration = new VertexDeclaration(device,
VertexPositionTexture.VertexElements);
}

```

كما ترى، لكل رأس قمنا بتحديد الموقع الثلاثي الأبعاد لهذا الرأس في الفضاء. لاحظ مرة أخرى أننا قمنا بتعريف الرؤوس باتجاه عقارب الساعة، لذا الـ Xna لن تقوم بغربلتهم "Culling" (راجع [الدروس الرابع](#) في السلسلة الأولى). الإحداثيين التاليين مهمان جداً، حيث يحددان أي نقطة في صورة الخامة سوف تكون مرتبطة مع الرأس الحالي. هذه الإحداثيات هي ببساطة إحداثيات X و Y للخامة، حيث الإحداثي (0,0) يشير إلى الزاوية العليا اليسرى في صورة الخامة، و الإحداثي (1,0) يمثل النقطة العليا اليمنى في الخامة، و الإحداثي (1,1) يمثل النقطة اليمنى السفلى في الخامة.

السطر الأخير يقوم بإنشاء الـ "VertexDeclaration" التي تمثل النوع VertexPositionTexture.

لا تنسى استدعاء الدالة SetUpVertices من داخل الدالة LoadContent:

```
SetUpVertices ();
```

حسناً، قمنا بتجهيز الرؤوس التي نريد، و قمنا بتحميل صورة الخامة في المتغير الازم. دعنا نقوم برسم المثلث!

إذهب إلى الدالة Draw، و قم بإضافة الكود التالي بعد عملية استدعاء الدالة Clear:

```

Matrix worldMatrix = Matrix.Identity;
effect.CurrentTechnique = effect.Techniques["Textured"];
effect.Parameters["xWorld"].SetValue(worldMatrix);
effect.Parameters["xView"].SetValue(viewMatrix);
effect.Parameters["xProjection"].SetValue(projectionMatrix);
effect.Parameters["xTexture"].SetValue(texture);
effect.Begin();
foreach (EffectPass pass in effect.CurrentTechnique.Passes)
{
    pass.Begin();
    device.VertexDeclaration = texturedVertexDeclaration;
    device.DrawUserPrimitives(PrimitiveType.TriangleList, vertices,
0, 1);
    pass.End();
}
effect.End();

```

كما العادة، حددنا ما هي التقنية التي نريد من بطاقة الرسومات إستخدامها لكي نقوم برسم/تصيير "Render" المثلث على الشاشة. حيث نريد إرشاد بطاقة الرسومات أن تأخذ لون كل بكسل من صورة الخامة. و هو ما تقوم به التقنية Textured الموجوده في ملف التأثير الذي نستخدمه، لذا نستدعيها لتكون هي التقنية الفعالة.

كما شرحنا في السلسلة الأولى، نقوم بإنشاء و إرسال مصفوفة العالم كمصفوفة محايدة، حيث سيتم رسم المثلث في المكان الذي تم تعريفه فيه. كما نقوم بإرسال مصفوفات العرض "View" و الإسقاط "Projection" بحيث يصبح بإمكان بطاقة الرسومات تحويل المواقع الثلاثية الأبعاد إلى إحداثيات الشاشة الثنائية الأبعاد.

أخيراً، قمنا بتمرير الخامة إلى التقنية المستخدمة. بعدها نقوم عملياً برسم المثلث من مصفوفة النقاط التي قمنا بتعريفها، بنفس الطريقة المستخدمة في السلسلة الأولى.

تشغيل هذا الكود سوف يعطينا فعلياً مثلث عليه خامة، ولكن المعروض هو نصف الخامة! من أجل عرض كامل الصورة، كل ما علينا عمله هو توسيع الدالة SetUpVertices من خلال إضافة المثلث الآخر:

```
private void SetUpVertices()
{
    vertices = new VertexPositionTexture[6];

    vertices[0].Position = new Vector3(-10f, 10f, 0f);
    vertices[0].TextureCoordinate.X = 0;
    vertices[0].TextureCoordinate.Y = 0;

    vertices[1].Position = new Vector3(10f, -10f, 0f);
    vertices[1].TextureCoordinate.X = 1;
    vertices[1].TextureCoordinate.Y = 1;

    vertices[2].Position = new Vector3(-10f, -10f, 0f);
    vertices[2].TextureCoordinate.X = 0;
    vertices[2].TextureCoordinate.Y = 1;

    vertices[3].Position = new Vector3(10.1f, -9.9f, 0f);
    vertices[3].TextureCoordinate.X = 1;
    vertices[3].TextureCoordinate.Y = 1;

    vertices[4].Position = new Vector3(-9.9f, 10.1f, 0f);
    vertices[4].TextureCoordinate.X = 0;
    vertices[4].TextureCoordinate.Y = 0;

    vertices[5].Position = new Vector3(10.1f, 10.1f, 0f);
    vertices[5].TextureCoordinate.X = 1;
    vertices[5].TextureCoordinate.Y = 0;

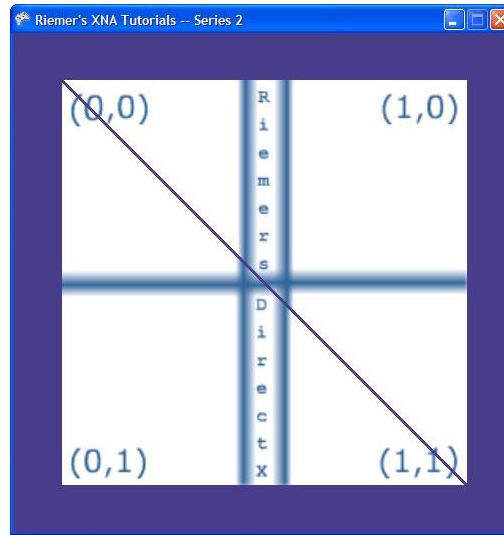
    texturedVertexDeclaration = new VertexDeclaration(device,
    VertexPositionTexture.VertexElements);
}
```

ببساطة قمنا بإضافة مجموعه أخرى مكونة من ثلاث رؤوس من أجل المثلث الثاني، لأجل إكمال الصورة.

لا تنسى أن تعدل الدالة Draw بحيث يتم رسم مثلثين بدلاً من واحد:

```
device.DrawUserPrimitives(PrimitiveType.TriangleList, vertices, 0, 2);
```

الآن قم بتشغيل الكود، سيكون بإمكانك مشاهدة صورة الخامة كاملة معروضة على مثلثين!



سوف تلاحظ الفراغ البسيط الموجود بين المثلثين... السبب في ذلك هو في الحقيقة لأننا قمنا بتعريف المواقع للرؤوس بالطريقة السابقة، حيث تتكون الصورة من مثلثين منفصلين.

حاول حل التمارين التالية:

- حاول حذف الفجوة بين المثلثين.
- قم بتغيير القيم الخاصة بإحداثيات الخامة في الدالة `SetUpVertices`، بإمكانك إختيار أي قيمة بين الصفر والواحد.

كود هذا الدرس:

```
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 XNAseries2
{
    public class Game1 : Microsoft.Xna.Framework.Game
    {
        GraphicsDeviceManager graphics;
        GraphicsDevice device;

        Effect effect;
        Matrix viewMatrix;
        Matrix projectionMatrix;
```

```
Texture2D texture;

VertexPositionTexture[] vertices;
VertexDeclaration texturedVertexDeclaration;

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 2";

    base.Initialize();
}

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

    effect = Content.Load<Effect> ("effects");
    texture = Content.Load<Texture2D> ("riemerstexture");

    SetUpVertices();
    SetUpCamera();
}

private void SetUpVertices()
{
    vertices = new VertexPositionTexture[6];

    vertices[0].Position = new Vector3(-10f, 10f, 0f);
    vertices[0].TextureCoordinate.X = 0;
    vertices[0].TextureCoordinate.Y = 0;

    vertices[1].Position = new Vector3(10f, -10f, 0f);
    vertices[1].TextureCoordinate.X = 1;
    vertices[1].TextureCoordinate.Y = 1;

    vertices[2].Position = new Vector3(-10f, -10f, 0f);
    vertices[2].TextureCoordinate.X = 0;
    vertices[2].TextureCoordinate.Y = 1;

    vertices[3].Position = new Vector3(10.1f, -9.9f, 0f);
    vertices[3].TextureCoordinate.X = 1;
    vertices[3].TextureCoordinate.Y = 1;

    vertices[4].Position = new Vector3(-9.9f, 10.1f, 0f);
    vertices[4].TextureCoordinate.X = 0;
    vertices[4].TextureCoordinate.Y = 0;

    vertices[5].Position = new Vector3(10.1f, 10.1f, 0f);
    vertices[5].TextureCoordinate.X = 1;
    vertices[5].TextureCoordinate.Y = 0;
```

```
texturedVertexDeclaration = new VertexDeclaration(device,
VertexPositionTexture.VertexElements);
}

private void SetUpCamera()
{
    viewMatrix = Matrix.CreateLookAt(new Vector3(0, 0, 30), new Vector3(0, 0, 0),
new Vector3(0, 1, 0));
    projectionMatrix = Matrix.CreatePerspectiveFieldOfView(MathHelper.PiOver4,
device.Viewport.AspectRatio, 0.2f, 500.0f);
}

protected override void UnloadContent()
{
}

protected override void Update(GameTime gameTime)
{
    base.Update(gameTime);
}

protected override void Draw(GameTime gameTime)
{
    device.Clear(ClearOptions.Target | ClearOptions.DepthBuffer,
Color.DarkSlateBlue, 1.0f, 0);

    Matrix worldMatrix = Matrix.Identity;
    effect.CurrentTechnique = effect.Techniques["Textured"];
    effect.Parameters["xWorld"].SetValue(worldMatrix);
    effect.Parameters["xView"].SetValue(viewMatrix);
    effect.Parameters["xProjection"].SetValue(projectionMatrix);
    effect.Parameters["xTexture"].SetValue(texture);
    effect.Begin();
    foreach (EffectPass pass in effect.CurrentTechnique.Passes)
    {
        pass.Begin();

        device.VertexDeclaration = texturedVertexDeclaration;
        device.DrawUserPrimitives(PrimitiveType.TriangleList, vertices, 0, 2);
        pass.End();
    }
    effect.End();

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