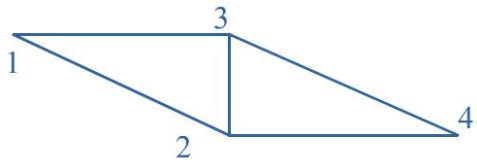
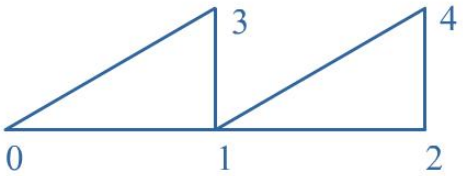
الدرس السادس

أهلا بكم في الدرس السادس من سلسلة دروس تعلم ال 3D Xna السلسلة الأولى، في هذا الدرس سوف نقوم بإنشاء الرؤوس من خلال الفهارس.

كان المثلث جميلا، ولكن ماذا عن مجموعة من المثلثات؟ سوف نحتاج لأن نعرف 3 رؤوس لكل مثلث. لاحظ المثال التالي:



هناك 4 رؤوس فريدة من أصل 6. لذا الرأسين الآخرين هم ببساطة مجرد إضاعة لطاقة بطاقة الرسوميات! لذلك من الأفضل تعريف 4 رؤوس في مصفوفة من 0 إلى 3، و بعدها تعريف المثلث الأول من الرؤوس 2،1 و 3 و المثلث الثاني بإستخدام الرؤوس 2,3 و 4. بهذه الطريقة، لن تتكرر بيانات الرؤوس المركبة. هذه هي الفكرة بالضبط من وراء الفهارس. إفترض أننا نريد أن نرسم هذين المثلثين:



في الوضع الطبيعي يجب علينا تعريف 6 رؤوس، ولكن الآن يجب علينا تعريف 5 رؤوس فقط. لذا قم بتغيير الدالة SetUpVertices كالتالي:

```
 private void SetUpVertices()
 {
     vertices = new VertexPositionColor[5];

     vertices[0].Position = new Vector3(0f, 0f, 0f);
     vertices[0].Color = Color.White;
     vertices[1].Position = new Vector3(5f, 0f, 0f);
     vertices[1].Color = Color.White;
     vertices[2].Position = new Vector3(10f, 0f, 0f);
     vertices[2].Color = Color.White;
     vertices[3].Position = new Vector3(5f, 0f, -5f);
     vertices[3].Color = Color.White;
     vertices[4].Position = new Vector3(10f, 0f, -5f);
     vertices[4].Color = Color.White;

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

الرؤوس من 0 إلى 2 تقع على المحور X الموجب، الرؤوس 3 و 4 لها قيمة سالبة للإحداثي Z، بما أن ال Xna تعتبر محور Z السالب متجهه للأمام (للداخل). بما أن الرؤوس معرفة بإتجاه الأمام، فإن المثلثات الناتجه لن تكون مسطحة.

بعدها، سوف نقوم بإنشاء مجموعة من الفهارس. كما تحدثنا سابقا، الفهارس تشير إلى الرؤوس الموجودة في مصفوفة الرؤوس. الفهارس تحدد المثلثات، لذا سوف نحتاج من أجل مثلثين أن نقوم بتعريف 6 فهارس. إبدأ بتعريف مصفوفة في أعلى كود الصنف. بما أن الفهارس عبارة عن أرقام صحيحة "Integer"، يجب علينا تعريف مصفوفة بإمكانها إحتواء الأرقام الصحيحة:

```
int[] indices;
```

دعنا نقوم بإنشاء دالة صغيرة جديدة تقوم بتعبئة مصفوفة الفهارس.

```
private void SetUpIndices()
{
    indices = new int[6];

    indices[0] = 3;
    indices[1] = 1;
    indices[2] = 0;
    indices[3] = 4;
    indices[4] = 2;
    indices[5] = 1;
}
```

كما تلاحظ، هذه الدالة تعرف 6 فهارس، يحددون مثلثين. الرأس رقم 1 يتم إستعماله مرتين، و هو هدفنا الأول كما ترى في الصورة السابقة. في هذه الحالة، الفائدة من العملية كانت قليلة إلى حد ما، ولكن في التطبيقات الأكبر (كما سوف ترى قريبا) ستكون هذه الطريقة الأفضل التي يجب إتباعها. لاحظ أيضا أن المثلثات تم تعريفهم بالترتيب بإتجاه عقارب الساعه مرة أخرى، لذا سوف ترى ال Xna هذه المثلثات بإعتبارها مواجهة للكاميرات ولن يتم غربلتهم.

تأكد من إستدعاء هذه الدالة من داخل الدالة LoadContent:

```
SetUpIndices();
```

كل ما تبقى لدينا في هذا الدرس هو أن نرسم المثلثات من الذاكرة "Buffer"! قم بتغيير السطر التالي في داخل الدالة Draw:

```
device.DrawUserIndexedPrimitives(PrimitiveType.TriangleList, vertices, 0,
vertices.Length, indices, 0, indices.Length / 3);
```

بدلا من الدالة DrawUserPrimitives، هذه المرة قمنا بإستدعاء الدالة DrawUserIndexedPrimitives. هذا سوف يتيح لنا أن نحدد مصفوفة للرؤوس و مصفوفة للفهارس. الوسيط الأخير يحدد كم عدد المثلثات المعرفة من خلال هذه الفهارس. بما أن المثلث يتم تحديدة بواسطة 3 فهارس، قمنا بتحديد عدد الفهارس مقسوما على 3.

قبل أن تجرب هذا الكود، قم بإيقاف المثلثات عن الدوران من خلال إعادة ضبط مصفوفة العالم الخاصة بهم إلى مصفوفة وحدة "Unity". المصفوفة المحايدة هي مصفوفة الوحدة، لذا سوف يتم إستخدام إحداثيات العالم الأصلية.

```
Matrix worldMatrix = Matrix.Identity;
```

هذا كل شيئ! عندما تقوم بتشغيل البرنامج، بكل الأحوال لن يكون هناك الكثير لتشاهده. هذا لأن كل من المثلثين و الكاميرا واقعين على الأرضية! يجب علينا تغيير موقع الكاميرا بحيث تصبح المثلثات في مجال رؤية الكاميرا. لذا إتجه إلى الدالة SetUpCamera، و قم بتغير موقع الكاميرا بحيث بحيث تقع فوق إحداثي نقطة الأصل الخاصة بنا (0,0,0):

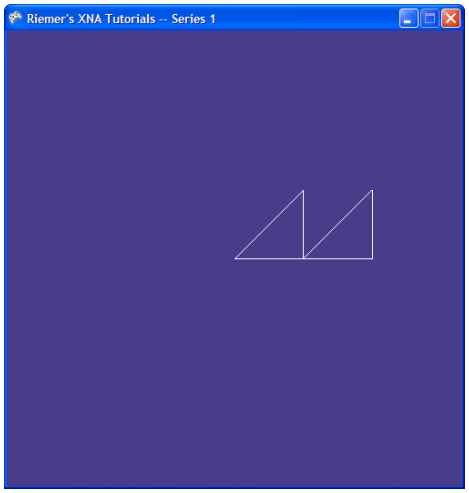
```
 viewMatrix = Matrix.CreateLookAt(new Vector3(0, 50, 0), new Vector3(0, 0, 0),
new Vector3(0, 0, -1));
```

قمنا بوضع الكاميرا فوق نقطة الأصل ب 50 وحدة، بما أن الXna تعتبر محور Y هو المتجهه للأعلى. بكل الأحوال، لأن الكاميرا تنظر إلى الأسفل، لن يكون بإمكانك إعتبار المتجه (0,1,0) بإعتباره المتجهه الخارج للأعلى بالنسبة للكاميرا! لذلك قمنا بتحديد المتجهه (0,0,-1) المتجهه للأمام بإعتباره المتجه للأعلى بالنسبة للكاميرا.

الآن عندما تقوم بتشغيل الكود، يجب عليك أن ترى كل من المثلثين، ولكنهما ما زالا معبئين. حاول وضع السطر التالي في السطر الأول في الدالة Draw:

```
device.RenderState.FillMode = FillMode.WireFrame;
```

سوف يؤدي ذلك إلى رسم الحواف الخاصة بالمثلثات، بدلا من تعبئتهما.



إذا لم يقم هذا الدرس أو الدرس التالي برسم أي مثلثات على الشاشة لديك، فإن الإحتمالات قد تكون أن بطاقة الرسوميات لديك غير متوافقه مع الرسم من عدة فهارس. لحل هذه المشكلة، قم بتخزين الفهارس كمتغيرات من نوع "Shorts" بدلا من ال "Integers". قم بتعريف المصفوفة كالتالي:

```
short[] indices;
```

و في الدالة SetUpIndices، قم بتجهيزه كالتالي:

```
indices = new short[6];
```

ذلك يجب أن يعمل، حتى مع الأجهزة التي لديها بطاقات رسوميات منخفضة الكفائة.

الفائدة من إستخدام الفهارس في هذا المثال ليست كبيرة جدا، حيث قمنا بتمرير 5 رؤوس بدلا من 6. في الدرس القادم سوف تكون الفائدة أكبر بكثير.

حاول تطبيق التمارين التالية للممارسة ما تعلمته:

- حاول رسم مثلث يتكون من الرؤوس 1,3 و 4. التغيير الوحيد الذي سوف تحتاجه هو في داخل الدالة SetUpIndices.
- قم بإستخدام متجه آخر مثل (1,0,0) بإعتباره المتجه الخارج للأعلى بالنسبة لمصفوفة عرض الكاميرا.

الكود 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;
        int[] indices;

        Matrix viewMatrix;
        Matrix projectionMatrix;

        private float angle = 0f;

        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();
        }

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

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

            SetUpIndices();
        }

        private void SetUpVertices()
        {
            vertices = new VertexPositionColor[5];

            vertices[0].Position = new Vector3(0f, 0f, 0f);
            vertices[0].Color = Color.White;
            vertices[1].Position = new Vector3(5f, 0f, 0f);
            vertices[1].Color = Color.White;
```

```
            vertices[2].Position = new Vector3(10f, 0f, 0f);
            vertices[2].Color = Color.White;
            vertices[3].Position = new Vector3(5f, 0f, -5f);
            vertices[3].Color = Color.White;
            vertices[4].Position = new Vector3(10f, 0f, -5f);
            vertices[4].Color = Color.White;

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

        private void SetUpIndices()
        {
            indices = new int[6];

            indices[0] = 3;
            indices[1] = 1;
            indices[2] = 0;
            indices[3] = 4;
            indices[4] = 2;
            indices[5] = 1;
        }

        private void SetUpCamera()
        {
            viewMatrix = Matrix.CreateLookAt(new Vector3(0, 50, 0), new Vector3(0, 0, 0), new
Vector3(0, 0, -1));
            projectionMatrix = Matrix.CreatePerspectiveFieldOfView(MathHelper.PiOver4,
device.Viewport.AspectRatio, 1.0f, 300.0f);
        }

        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);
            device.RenderState.CullMode = CullMode.None;
            device.RenderState.FillMode = FillMode.WireFrame;

            effect.CurrentTechnique = effect.Techniques["Colored"];
            effect.Parameters["xView"].SetValue(viewMatrix);
            effect.Parameters["xProjection"].SetValue(projectionMatrix);
            effect.Parameters["xWorld"].SetValue(Matrix.Identity);

            effect.Begin();
            foreach (EffectPass pass in effect.CurrentTechnique.Passes)
            {
                pass.Begin();

                device.VertexDeclaration = myVertexDeclaration;
                device.DrawUserIndexedPrimitives(PrimitiveType.TriangleList, vertices, 0,
vertices.Length, indices, 0, indices.Length / 3);

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

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