الدرس الثامن

أهلا بكم في الدرس الثامن من سلسلة دروس تعلم ال 3D Xna السلسلة الأولى، في هذا الدرس سوف نقوم بإنشاء التضاريس من ملف.

أخيرا قد حان الوقت لإنشاء منظر طبيعي جميل ☺. بدلا من إدخال البيانات بشكل يدوي داخل المصفوفة heightData، سوف نقوم بملئها من ملف. لعمل ذلك سوف نقوم بإستيراد صورة لتدرج رمادي بحجم 128X128 بكسل. سوف نقوم بإستخدام قيمة اللون الأبيض 'white value' لكل بكسل من أجل تمثيل إرتفاع الإحداثي للرأس المقابل! بإمكانك تنزيل ملف المثال الخاص بي من الرابط التالي هنا أو من الملفات المرفقة.

بعد أن تقوم بتنزيل الملف، قم بإضافته للمشروع بنفس الطريقة التي قمت بها لملف التأير fx. . بإمكانك أيضا عمل ذلك من خلال سحب الملف من المجلد الموجود فيه (من خلال متصفح الويندوز) و إسقاطه في مجلد ال Content داخل مشروع ال Xna. إذا كان كل شيئ على ما يرام، يجب أن ترى ملف ال heightmap.bmp قد تم إضافته إلى مجلد ال Content.

الملف يجب أن يتم تحميله داخل الدالة LoadContent. ملف التأثير تم تحميله في متغير من نوع Effect، و الصورة يجب أن يتم تحميلها في متغير من النوع Texture2D:

```
Texture2D heightMap = Content.Load<Texture2D> ("heightmap");
LoadHeightData(heightMap);
```

عند إستخدام الأنبوب "Content Pipeline"، لا يهم إذا كنت تستخدم ملفات من النوع bmp,jpg أو png. السطر الثاني يقوم بإستدعاء الدالة LoadHeightData، التي سوف نقوم بتعديلها بحيث تقوم بإستقبال هذه الخامة. هذا هي النسخة الجديدة من الدالة LoadHeightData:

```
private void LoadHeightData(Texture2D heightMap)
{
    terrainWidth = heightMap.Width;
    terrainHeight = heightMap.Height;

    Color[] heightMapColors = new Color[terrainWidth * terrainHeight];
    heightMap.GetData(heightMapColors);

    heightData = new float[terrainWidth, terrainHeight];
    for (int x = 0; x < terrainWidth; x++)
        for (int y = 0; y < terrainHeight; y++)
            heightData[x, y] = heightMapColors[x + y * terrainWidth].R / 5.0f;
}
```

كما ترى، هذه الدالة تستقبل الصورة كويسط. بدلا من إستخدام طول و عرض معرفين بشكل مسبق للتضاريس، تقوم الدالة بإستخدام أبعاد الصورة. أول سطرين يقرءان العرض و الطول للصورة، و يتم تخزينهم بإعتبارهم الطول و العرض للتضاريس في جميع اجزاء باقي البرنامج. هذا سوف يؤدي في باقي الكود إلى إنشاء عدد كافي من الرؤوس والفهارس بشكل تلقائي، و إلى رسم عدد كافي من المثلثات.

لأنك تريد الوصول إلى قيم الألوان للصورة، قمنا بإنشاء تحتوي على كائنات من النوع Color، حيث يتم تخزين اللون لكل بكسل في الصورة. تم عمل ذلك بسهولة في سطرين.

المهم التالية هي إعادة تشكيل مصفوفة الألوان أحادية البعد هذه إلى مصفوفة أعداد حقيقية "Float" ثنائية البعد. أولا قم بإنشاء مصفوفة أعداد حقيقة ثنائية البعد بحجم مناسب. بعدها، يتم المرور على كل الألوان و يتم إختيار قيم اللون الأحمر، التي تقع

بين 0 (لا يوجد لون) و 255 (أحمر بشكل تام). قبل أن تقوم بتخزين هذه القيمة يتم تصغيرها بشكل قليل، حتى لا تكون التضاريس عميقة جدا.

في هذه اللحظة، لديك مصفوفة ثنائية الأبعاد تحتوي على إرتفاع كل نقطة في التضاريس. الدالة SetUpVertices سوف تقوم بإنشاء رأس لكل نقطة في المصفوفة. الدالة SetUpIndices سوف تقوم بإنشاء 3 فهارس لكل مثلث سوف يتم رسمه من أجل تغطية التضاريس. في النهاية، الدالة Draw سوف تقوم برسم العديد من المثلثات بناء على ما تحتوي مصفوفة الفهارس indices. لذا عند تشغيل الكود يجب أن ترى التضاريس!

لسوء الحظ، مرة أخرى ☹. ولكن الحل بسيط، هذا الركن من التضاريس يقع فوق الكاميرا! لذا إذا قمت بزيادة إرتفاع الكاميرا إلى 40، يجب أن ترى التضاريس.

عندما تقوم بتشغيل البرنامج، سوف ترى فقط ركن واحد من التضاريس الكبيرة. إذن يجب تحريك التضاريس، لذا يجب أن يتم سحب مركزها إلى نقطة الأصل ثلاثية الأبعاد (0,0,0). يمكن عمل ذلك داخل الدالة Draw:

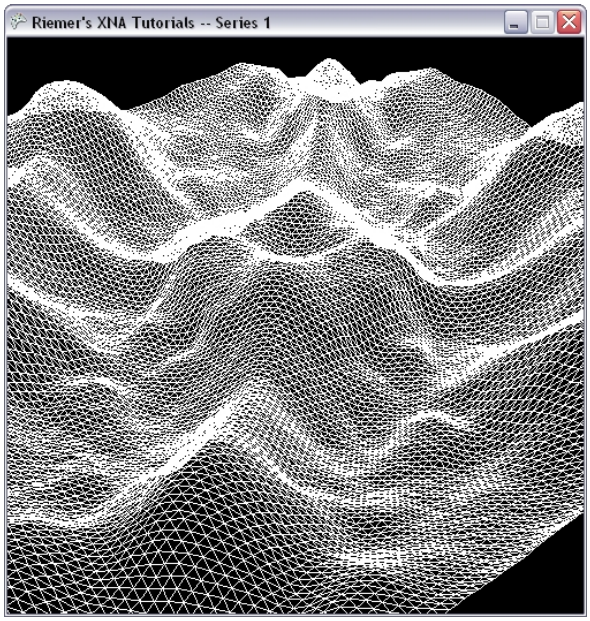
```
 Matrix worldMatrix = Matrix.CreateTranslation(-terrainWidth / 2.0f, 0,
terrainHeight / 2.0f);
 effect.CurrentTechnique = effect.Techniques["Colored"];
 effect.Parameters["xView"].SetValue(viewMatrix);
 effect.Parameters["xProjection"].SetValue(projectionMatrix);
 effect.Parameters["xWorld"].SetValue(worldMatrix);
```

ذلك سوف يقوم بنقل التضاريس إلى وسط الشاشة.

بما أن الكاميرا ما زالت تنظر بشكل مستقيم إلى التضاريس، سوف نحصل على مشهد أفضل إذا قمنا بتغير موقع الكاميرا شيئ ما:

```
 viewMatrix = Matrix.CreateLookAt(new Vector3(60, 80, -80), new Vector3(0, 0,
0), new Vector3(0, 1, 0));
```

عندما تختار اللون الأسود Color.Black بإعتباره لون المسح يجب أن تحصل على الصورة المعروضة في الأسفل.



بإمكانك محاولة حل التمارين التالية من أجل ممارسة ما قد تعلمته في هذا الدرس:

- هذا الكود يقوم بتصغير قيمة اللون الأحمر ل5 أضعاف. حاول إستخدام قيم أخرى للتحجيم.
- إفتح الصورة heihgtmap.bmp بإستخدام برنامج الرسام، وقم بتعديل قيم بعض البكسلات. إجعل بعض البكسلات بلون أحمر تام. و بعضها الآخر أزرق تام. قم بتحميل الصورة إلى البرنامج ولاحظ التغييرات.
- حاول إيجاد صورة أخرى وقم بتحميلها إلى البرنامج. إبحث عن صور لها أبعاد قليلة، لأن هذه الطريقة لن تتيح لك رسم تضاريس كبيرة.

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 int terrainWidth;
        private int terrainHeight;
        private float[,] heightData;

        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");


            Texture2D heightMap = Content.Load<Texture2D>
("heightmap");            LoadHeightData(heightMap);


            SetUpVertices();
            SetUpCamera();
```

```
            SetUpIndices();
        }

        private void SetUpVertices()
        {
            vertices = new VertexPositionColor[terrainWidth * terrainHeight];
            for (int x = 0; x < terrainWidth; x++)
            {
                for (int y = 0; y < terrainHeight; y++)
                {
                    vertices[x + y * terrainWidth].Position = new Vector3(x, heightData[x, y], -y);
                    vertices[x + y * terrainWidth].Color = Color.White;
                }
            }

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

        private void SetUpIndices()
        {
            indices = new int[(terrainWidth - 1) * (terrainHeight - 1) * 6];
            int counter = 0;
            for (int y = 0; y < terrainHeight - 1; y++)
            {
                for (int x = 0; x < terrainWidth - 1; x++)
                {
                    int lowerLeft = x + y*terrainWidth;
                    int lowerRight = (x + 1) + y*terrainWidth;
                    int topLeft = x + (y + 1) * terrainWidth;
                    int topRight = (x + 1) + (y + 1) * terrainWidth;

                    indices[counter++] = topLeft;
                    indices[counter++] = lowerRight;
                    indices[counter++] = lowerLeft;

                    indices[counter++] = topLeft;
                    indices[counter++] = topRight;
                    indices[counter++] = lowerRight;
                }
            }
        }

        private void SetUpCamera()
        {
            viewMatrix = Matrix.CreateLookAt(new Vector3(60, 80, -80), new Vector3(0, 0, 0), new
Vector3(0, 1, 0));
            projectionMatrix = Matrix.CreatePerspectiveFieldOfView(MathHelper.PiOver4,
device.Viewport.AspectRatio, 1.0f, 300.0f);
        }

        private void LoadHeightData(Texture2D heightMap)
        {
            terrainWidth = heightMap.Width;
            terrainHeight = heightMap.Height;

            Color[] heightMapColors = new Color[terrainWidth * terrainHeight];
            heightMap.GetData(heightMapColors);

            heightData = new float[terrainWidth, terrainHeight];
            for (int x = 0; x < terrainWidth; x++)
                for (int y = 0; y < terrainHeight; y++)
                    heightData[x, y] = heightMapColors[x + y * terrainWidth].R / 5.0f;
        }

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

```
            Matrix worldMatrix = Matrix.CreateTranslation(-terrainWidth / 2.0f, 0, terrainHeight /
2.0f);
            effect.CurrentTechnique = effect.Techniques["Colored"];
            effect.Parameters["xView"].SetValue(viewMatrix);
            effect.Parameters["xProjection"].SetValue(projectionMatrix);
            effect.Parameters["xWorld"].SetValue(worldMatrix);

            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);
        }
    }
 }
```