الدرس التاسع

أهلا بكم في الدرس التاسع من سلسلة دروس تعلم ال 3D Xna السلسلة الأولى، في هذا الدرس سوف نقوم بتدوير التضاريس بإستخدام لوحة المفاتيح.

بإستخدام ال Xna، من السهل جدا قراءة حالة لوحة المفاتيح الحالية. المكتبة الصحيحة، Microsoft.Xna.Framework.Input، قد تم ربطها مباشرة بالبرنامج عندما قمت ببدء مشروع Xna، لذا بإستطاعتنا مباشرة التوجه لكتابة الكود الذي يقرأ مدخلات لوحة المفاتيح.

قم بإضافة الكود الصغير التالي إلى الدالة Update، التي يتم إستدعائها 60 مره في الثانية:

```
KeyboardState keyState = Keyboard.GetState();
if (keyState.IsKeyDown(Keys.Delete))
    angle += 0.05f;
if (keyState.IsKeyDown(Keys.PageDown))
    angle -= 0.05f;
```

هنا يتم وضع الحالة الحالية للوحة المفاتيح في المتغير 'KeyState'. بإستخدام هذا المتغير، بإمكانك القراءة مباشرة ما هي المفاتيح المضغوطة! عندما يقوم المستخدم بضغط الزر Delete أو الزر PageDown، سوف يتم تغير قيمة المتغير angle.

في الدالة Draw، يجب علينا التأكد أن هذا التدوير يتم أخذه بالحسبان في مصفوفة العالم:

```
Matrix worldMatrix = Matrix.CreateTranslation(-terrainWidth / 2.0f, 0,
terrainHeight / 2.0f) * Matrix.CreateRotationY(angle);
```

التضاريس يتم تدويرها حول محور Y المتجه للأعلى بعد نقله إلى إحداثي نقطة الأصل (0,0,0).

هذا كل شيئ! عند قيامك بتشغيل الكود، بإمكانك تدوير التضاريس ببساطه عن طريق ضغط زر Delete و زر PageDown!

حاول حل التمرين التالي، من أجل ممارسة ما قد تعلمته:

- قم بإضافة بعض الكود بحيث يتم قراءة الأسهم. و قم بتعديل مصفوفة العالم بحيث يتم تحريك التضاريس بإتجاهات الأسهم المضغوطة.

الكود حتى اللحظة:

```
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;

        float angle = 0;

        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();


             KeyboardState keyState = Keyboard.GetState();
             if (keyState.IsKeyDown(Keys.Delete))
                 angle += 0.05f;
             if (keyState.IsKeyDown(Keys.PageDown))
                 angle -= 0.05f;

             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) * Matrix.CreateRotationY(angle);
             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);
         }
     }
 }
```