الدرس السادس

أهلا بكم في الدرس السادس من سلسلة دروس تعلم 3D Xna (السلسلة الثانية)، في هذا الدرس سنضيف الإضاءة إلى مدينتنا بإذن الله. سيكون هذا الدرس قصير جدا، لأننا قمنا فعليا بشرح أساسيات الإضاءة في السلسلة الأولى.

كنوع أول من الإضاءة، سوف نقوم بإستخدام ضوء موجه "directional"، كما إستخدماه في السلسلة الأولى. كل الرؤوس في العناصر في المشهد لدينا تحتوي فعليا على معلومات المتجهات العموديه: حيث قمنا بإضافتها بشكل صريح إلى الرؤوس في المدينة الثلاثية الأبعاد، كما أن ملف ال X. الذي قمنا بإستراد الطائرة منه يحتوي أيضا على معلومات المتجهات العمودية الخاصة بها. لذا كل ما نحتاج لعمله هو أن نقوم بتعريف إتجاه الضوء الذي نريد، و تنبيه التقنيات "techniques" التي نستخدمها؛ أن تأخذ الإضاءة بالحسبان!

لذا إبدأ بتعريف إتجاه الضوء من خلال إضافة السطر التالي في أعلى الكود لديك:

```
Vector3 lightDirection = new Vector3(3, -2, 5);
```

كما وضحنا في السلسلة الأولى، نحتاج لأن يكون الطول النهائي لإتجاه الضوء مساوي لواحد "1". بإمكاننا عمل ذلك من خلال عمل تطبيع "normalizing" للإتجاه، لذا قم بإضافة السطر التالي إلى الدالة Initialize:

```
lightDirection.Normalize();
```

الآن قمنا بتعريف الإتجاه الذي نريده للإضاءة، لذا إذهب إلى الدوال DrawCity و DrawModel. وقم بإضافة الأسطر التالية إلى عملية إعداد التأثير، في الدالة DrawCity أولا:

```
effect.Parameters["xEnableLighting"].SetValue(true);
effect.Parameters["xLightDirection"].SetValue(lightDirection);
```

ثم إلى الدالة DrawModel:

```
currentEffect.Parameters["xEnableLighting"].SetValue(true);
currentEffect.Parameters["xLightDirection"].SetValue(lightDirection);
```

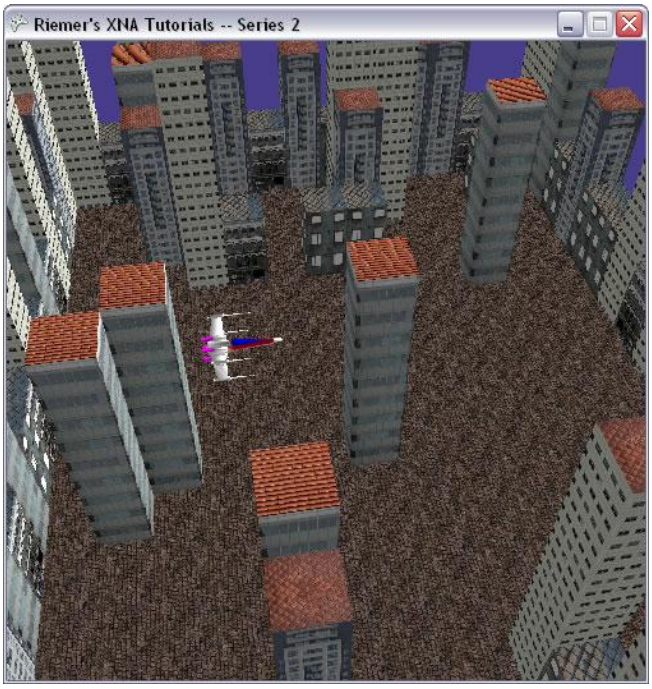
الآن قم بتشغيل الكود. سترى أن بعض أوجه المباني مضيئه أكثر من الأوجه الأخرى. بكل الأحوال، الأوجه التي تقع في الإتجاه المعاكس للضوء لاتستقبل أي إضاءة، وهي سوداء بشكل كامل! هذا ليس ما نريده، لذا قم بإضافة بعض الإضاءة المحيطة "ambient lighting" إلى كل البكسلات في المدينة، وذلك في الدالة DrawCity:

```
effect.Parameters["xAmbient"].SetValue(0.5f);
```

و في داخل الدالة DrawModel:

```
currentEffect.Parameters["xAmbient"].SetValue(0.5f);
```

عندما تقوم بتشغيل هذا الكود، سترى أنه لم يبقى هناك أي أوجه سوداء في المباني. ولكن أغلب المباني قاتمه إلى حدا ما، ولكن سبب ذلك أننا ننظر إلى الجهة التي تحتوي على الظل في المدينة. كما يجب أن تكون قد لاحظت أن هناك بعض الاوجه في الجهة اليسرى من المدينة مازالت مشرقة نتيجة ضوء الشمس (الضوء الموجه) الذي قمنا بإضافته.



من المؤكد أنه بالإمكان الوصول إلى تأثيرات إضاءة بنتيجة أفضل من خلال تعريف أنواع أخرى من الإضاءة، لكنني لم أقم بإضافة هذه التأثيرات إلى ملف التأثيرات الخاص بي، لأنك في السلسلة الثالثة سوف تتعلم كيف تفعل ذلك بنفسك، بإذن الله.

حاول حل التمارين التالية، لممارسة ما قد تعلمته:

- قم بتغيير قوة الإضاءة المحيطة.
- غير إتجاه ضوء الشمس.
- قم بتعديل موقع الكاميرا لكي تحصل على شعور بتأثير الضوء المتجه على المدينة.

الكود :

```
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
    {
```

```
        int[] buildingHeights = new int[] { 0, 2, 2, 6, 5, 4 };

        GraphicsDeviceManager graphics;
        GraphicsDevice device;

        Effect effect;
        Vector3 lightDirection = new Vector3(3, -2, 5);
        Matrix viewMatrix;
        Matrix projectionMatrix;

        Texture2D sceneryTexture;
        Model xwingModel;

        int[,] floorPlan;

        VertexBuffer cityVertexBuffer;
        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";

            LoadFloorPlan();
            lightDirection.Normalize();

            base.Initialize();
        }

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


           effect = Content.Load<Effect> ("effects");
           sceneryTexture = Content.Load<Texture2D> ("texturemap");
           xwingModel = LoadModel("xwing");

           SetUpVertices();
           SetUpCamera();
       }

       private Model LoadModel(string assetName)
       {

           Model newModel = Content.Load<Model> (assetName);            foreach (ModelMesh mesh in
newModel.Meshes)
                foreach (ModelMeshPart meshPart in mesh.MeshParts)
                    meshPart.Effect = effect.Clone(device);
           return newModel;
       }

       private void LoadFloorPlan()
       {
           floorPlan = new int[,]
            {
                {1,1,1,1,1,1,1,1,1,1,1,1,1,1},
                {1,0,0,0,0,0,0,0,0,0,0,0,0,1},
                {1,0,0,1,1,0,0,0,1,1,0,0,1,1},
                {1,0,0,1,1,0,0,0,1,0,0,0,1,1},
                {1,0,0,0,1,1,0,1,1,0,0,0,0,1},
                {1,0,0,0,0,0,0,0,0,0,0,1,0,1},
                {1,0,0,0,0,0,0,0,0,0,0,0,0,1},
                {1,0,0,0,0,0,0,0,0,0,0,0,0,1},
                {1,0,0,0,0,0,0,0,0,0,0,0,0,1},
                {1,0,0,0,0,0,0,0,0,0,0,0,0,1},
                {1,0,1,1,0,0,0,1,0,0,0,0,0,1},
                {1,0,1,0,0,0,0,0,0,0,0,0,0,1},
                {1,0,0,0,0,0,0,0,0,0,0,0,0,1},
                {1,0,0,0,0,0,0,0,0,0,0,0,0,1},
                {1,0,0,0,0,1,0,0,0,0,0,0,0,1},
                {1,0,0,0,0,1,0,0,0,1,0,0,0,1},
```

```
                {1,0,1,0,0,0,0,0,1,0,0,0,0,1},
                {1,0,1,1,0,0,0,0,1,1,0,0,0,1,1},
                {1,0,0,0,0,0,0,0,1,1,0,0,0,1,1},
                {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},
             };

            Random random = new Random();
            int differentBuildings = buildingHeights.Length - 1;
            for (int x = 0; x < floorPlan.GetLength(0); x++)
                for (int y = 0; y < floorPlan.GetLength(1); y++)
                    if (floorPlan[x, y] == 1)
                        floorPlan[x, y] = random.Next(differentBuildings) + 1;
        }

        private void SetUpCamera()
        {
            viewMatrix = Matrix.CreateLookAt(new Vector3(20, 13, -5), new Vector3(8, 0, -7), new
Vector3(0, 1, 0));
            projectionMatrix = Matrix.CreatePerspectiveFieldOfView(MathHelper.PiOver4,
device.Viewport.AspectRatio, 0.2f, 500.0f);
        }

        private void SetUpVertices()
        {
            int differentBuildings = buildingHeights.Length - 1;
            float imagesInTexture = 1 + differentBuildings * 2;

            int cityWidth = floorPlan.GetLength(0);
            int cityLength = floorPlan.GetLength(1);


            List<VertexPositionNormalTexture> verticesList = new List<VertexPositionNormalTexture> ();
            for (int x = 0; x < cityWidth; x++)
            {
                for (int z = 0; z < cityLength; z++)
                {
                    int currentbuilding = floorPlan[x, z];

                    //floor or ceiling
                    verticesList.Add(new VertexPositionNormalTexture(new Vector3(x,
buildingHeights[currentbuilding], -z), new Vector3(0, 1, 0), new Vector2(currentbuilding * 2 /
imagesInTexture, 1)));
                    verticesList.Add(new VertexPositionNormalTexture(new Vector3(x,
buildingHeights[currentbuilding], -z - 1), new Vector3(0, 1, 0), new Vector2((currentbuilding * 2) /
imagesInTexture, 0)));
                    verticesList.Add(new VertexPositionNormalTexture(new Vector3(x + 1,
buildingHeights[currentbuilding], -z), new Vector3(0, 1, 0), new Vector2((currentbuilding * 2 + 1) /
imagesInTexture, 1)));

                    verticesList.Add(new VertexPositionNormalTexture(new Vector3(x,
buildingHeights[currentbuilding], -z - 1), new Vector3(0, 1, 0), new Vector2((currentbuilding * 2) /
imagesInTexture, 0)));
                    verticesList.Add(new VertexPositionNormalTexture(new Vector3(x + 1,
buildingHeights[currentbuilding], -z - 1), new Vector3(0, 1, 0), new Vector2((currentbuilding * 2 + 1)
/ imagesInTexture, 0)));
                    verticesList.Add(new VertexPositionNormalTexture(new Vector3(x + 1,
buildingHeights[currentbuilding], -z), new Vector3(0, 1, 0), new Vector2((currentbuilding * 2 + 1) /
imagesInTexture, 1)));

                    if (currentbuilding != 0)
                    {
                        //front wall
                        verticesList.Add(new VertexPositionNormalTexture(new Vector3(x + 1, 0, -z - 1),
new Vector3(0, 0, -1), new Vector2((currentbuilding * 2) / imagesInTexture, 1)));
                        verticesList.Add(new VertexPositionNormalTexture(new Vector3(x,
buildingHeights[currentbuilding], -z - 1), new Vector3(0, 0, -1), new Vector2((currentbuilding * 2 - 1)
/ imagesInTexture, 0)));
                        verticesList.Add(new VertexPositionNormalTexture(new Vector3(x, 0, -z - 1), new
Vector3(0, 0, -1), new Vector2((currentbuilding * 2 - 1) / imagesInTexture, 1)));

                        verticesList.Add(new VertexPositionNormalTexture(new Vector3(x,
buildingHeights[currentbuilding], -z - 1), new Vector3(0, 0, -1), new Vector2((currentbuilding * 2 - 1)
/ imagesInTexture, 0)));
                        verticesList.Add(new VertexPositionNormalTexture(new Vector3(x + 1, 0, -z - 1),
new Vector3(0, 0, -1), new Vector2((currentbuilding * 2) / imagesInTexture, 1)));
                        verticesList.Add(new VertexPositionNormalTexture(new Vector3(x + 1,
buildingHeights[currentbuilding], -z - 1), new Vector3(0, 0, -1), new Vector2((currentbuilding * 2) /
imagesInTexture, 0)));

                        //back wall
                        verticesList.Add(new VertexPositionNormalTexture(new Vector3(x + 1, 0, -z), new
```

```
Vector3(0, 0, 1), new Vector2((currentbuilding * 2) / imagesInTexture, 1)));
                        verticesList.Add(new VertexPositionNormalTexture(new Vector3(x, 0, -z), new
Vector3(0, 0, 1), new Vector2((currentbuilding * 2 - 1) / imagesInTexture, 1)));
                        verticesList.Add(new VertexPositionNormalTexture(new Vector3(x,
buildingHeights[currentbuilding], -z), new Vector3(0, 0, 1), new Vector2((currentbuilding * 2 - 1) /
imagesInTexture, 0)));

                        verticesList.Add(new VertexPositionNormalTexture(new Vector3(x,
buildingHeights[currentbuilding], -z), new Vector3(0, 0, 1), new Vector2((currentbuilding * 2 - 1) /
imagesInTexture, 0)));
                        verticesList.Add(new VertexPositionNormalTexture(new Vector3(x + 1,
buildingHeights[currentbuilding], -z), new Vector3(0, 0, 1), new Vector2((currentbuilding * 2) /
imagesInTexture, 0)));
                        verticesList.Add(new VertexPositionNormalTexture(new Vector3(x + 1, 0, -z), new
Vector3(0, 0, 1), new Vector2((currentbuilding * 2) / imagesInTexture, 1)));

                        //left wall
                        verticesList.Add(new VertexPositionNormalTexture(new Vector3(x, 0, -z), new
Vector3(-1, 0, 0), new Vector2((currentbuilding * 2) / imagesInTexture, 1)));
                        verticesList.Add(new VertexPositionNormalTexture(new Vector3(x, 0, -z - 1), new
Vector3(-1, 0, 0), new Vector2((currentbuilding * 2 - 1) / imagesInTexture, 1)));
                        verticesList.Add(new VertexPositionNormalTexture(new Vector3(x,
buildingHeights[currentbuilding], -z - 1), new Vector3(-1, 0, 0), new Vector2((currentbuilding * 2 - 1)
/ imagesInTexture, 0)));

                        verticesList.Add(new VertexPositionNormalTexture(new Vector3(x,
buildingHeights[currentbuilding], -z - 1), new Vector3(-1, 0, 0), new Vector2((currentbuilding * 2 - 1)
/ imagesInTexture, 0)));
                        verticesList.Add(new VertexPositionNormalTexture(new Vector3(x,
buildingHeights[currentbuilding], -z), new Vector3(-1, 0, 0), new Vector2((currentbuilding * 2) /
imagesInTexture, 0)));
                        verticesList.Add(new VertexPositionNormalTexture(new Vector3(x, 0, -z), new
Vector3(-1, 0, 0), new Vector2((currentbuilding * 2) / imagesInTexture, 1)));

                        //right wall
                        verticesList.Add(new VertexPositionNormalTexture(new Vector3(x + 1, 0, -z), new
Vector3(1, 0, 0), new Vector2((currentbuilding * 2) / imagesInTexture, 1)));
                        verticesList.Add(new VertexPositionNormalTexture(new Vector3(x + 1,
buildingHeights[currentbuilding], -z - 1), new Vector3(1, 0, 0), new Vector2((currentbuilding * 2 - 1)
/ imagesInTexture, 0)));
                        verticesList.Add(new VertexPositionNormalTexture(new Vector3(x + 1, 0, -z - 1),
new Vector3(1, 0, 0), new Vector2((currentbuilding * 2 - 1) / imagesInTexture, 1)));

                        verticesList.Add(new VertexPositionNormalTexture(new Vector3(x + 1,
buildingHeights[currentbuilding], -z - 1), new Vector3(1, 0, 0), new Vector2((currentbuilding * 2 - 1)
/ imagesInTexture, 0)));
                        verticesList.Add(new VertexPositionNormalTexture(new Vector3(x + 1, 0, -z), new
Vector3(1, 0, 0), new Vector2((currentbuilding * 2) / imagesInTexture, 1)));
                        verticesList.Add(new VertexPositionNormalTexture(new Vector3(x + 1,
buildingHeights[currentbuilding], -z), new Vector3(1, 0, 0), new Vector2((currentbuilding * 2) /
imagesInTexture, 0)));
                    }
                }
            }

            cityVertexBuffer = new VertexBuffer(device, verticesList.Count *
VertexPositionNormalTexture.SizeInBytes, BufferUsage.WriteOnly);

            cityVertexBuffer.SetData<VertexPositionNormalTexture> (verticesList.ToArray());
            texturedVertexDeclaration = new VertexDeclaration(device,
VertexPositionNormalTexture.VertexElements);
        }

        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);

            DrawCity();
            DrawModel();

            base.Draw(gameTime);
        }
```

```
        private void DrawCity()
        {
            effect.CurrentTechnique = effect.Techniques["Textured"];
            effect.Parameters["xWorld"].SetValue(Matrix.Identity);
            effect.Parameters["xView"].SetValue(viewMatrix);
            effect.Parameters["xProjection"].SetValue(projectionMatrix);
            effect.Parameters["xTexture"].SetValue(sceneryTexture);

             effect.Parameters["xEnableLighting"].SetValue(true);
             effect.Parameters["xLightDirection"].SetValue(lightDirection);
             effect.Parameters["xAmbient"].SetValue(0.5f);
             effect.Begin();
             foreach (EffectPass pass in effect.CurrentTechnique.Passes)
             {
                 pass.Begin();
                 device.VertexDeclaration = texturedVertexDeclaration;
                 device.Vertices[0].SetSource(cityVertexBuffer, 0,
VertexPositionNormalTexture.SizeInBytes);
                 device.DrawPrimitives(PrimitiveType.TriangleList, 0, cityVertexBuffer.SizeInBytes /
VertexPositionNormalTexture.SizeInBytes / 3);
                 pass.End();
             }
             effect.End();
         }

         private void DrawModel()
         {
             Matrix worldMatrix = Matrix.CreateScale(0.0005f, 0.0005f, 0.0005f) *
Matrix.CreateRotationY(MathHelper.Pi) * Matrix.CreateTranslation(new Vector3(19, 12, -5));

             Matrix[] xwingTransforms = new Matrix[xwingModel.Bones.Count];
             xwingModel.CopyAbsoluteBoneTransformsTo(xwingTransforms);
             foreach (ModelMesh mesh in xwingModel.Meshes)
             {
                 foreach (Effect currentEffect in mesh.Effects)
                 {
                     currentEffect.CurrentTechnique = currentEffect.Techniques["Colored"];
                     currentEffect.Parameters["xWorld"].SetValue(xwingTransforms[mesh.ParentBone.Index]
* worldMatrix);
                     currentEffect.Parameters["xView"].SetValue(viewMatrix);
                     currentEffect.Parameters["xProjection"].SetValue(projectionMatrix);
                     currentEffect.Parameters["xEnableLighting"].SetValue(true);
                     currentEffect.Parameters["xLightDirection"].SetValue(lightDirection);
                     currentEffect.Parameters["xAmbient"].SetValue(0.5f);
                 }
                 mesh.Draw();
             }
         }
     }
 }
```