الدرس الثاني

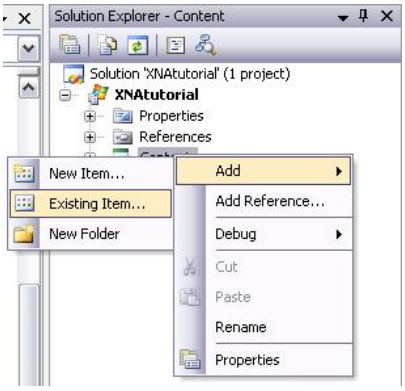
أهلا بكم في الدرس الثاني من سلسلة دروس تعلم ال Xna , في هذا الدرس سوف نقوم برسم خلفية اللعبة.

الخطوة الأولى في عملية إنشاء لعبة ثنائية الأبعاد هي عملية تصيير "Render" صورة ثنائية الأبعاد. في ال Xna, من السهل جدا عمل ذلك. كمثال أول سوف نقوم بتصيير صورتين على كامل الشاشة.

الأولى سوف تكون صورة الأرضية الخلفية و التي تحتوي على مشهد السماء و الجبال وفوق هذه الصورة سوف نقوم بتصيير صورة الأرضية الأمامية التي تحتوي على التضاريس.

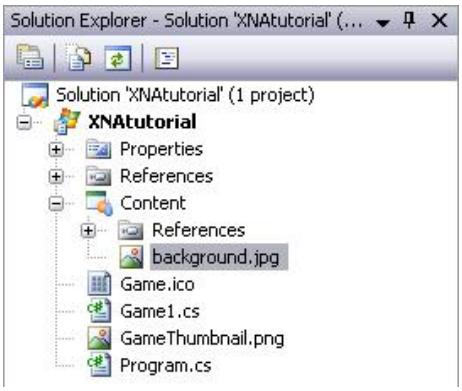
بداية يجب علينا تنزيل الصور من خلال الروابط التالية هنا و الرابط هنا (أو من خلال المرفقات).

الآن إذهب إلى المشروع الذي قمنا بإنشائه في الدرس السابق, و إذهب إلى "Solution Explorer" مربع صغير في الركن العلوي الأيسر من الشاشة, يحتوي على كل ملفات المشروع. بعدها ابحث عن مجلد ال "Content" و بعدها إضغط بالزر اليمين على المجلد و إختر Add->Existing Item , كما هو موضح في الصورة التالية:



من خلال الشاشة المفتوحة قم بالتصفح حتى تصل إلى موقع الصورتان اللتان قمت بتنزيلهما سابقا. إختر ملفات الصوره و بعدها إضغط إضافة "Add" .

هكذا سوف يتم نسخ الملف إلى مجلد ال "Content" في مشروعك. يجب أن يظهر الملف اسفل مجلد ال "Content" كما في الصورة:



بعد تحميل الصوره إلى مشروع ال Xna الخاص بك, يجب أن نقوم بإنشاء متغير في الكود, بحيث يتم ربط المتغير مع الصورة. وجود المتغير هو شرط أساسي, مما يمكننا من الوصول إلى الصورة من خلال الكود.

قم بإضافة المتغير في أعلى الكود , فوق الدالة "()Game1" .

```
Texture2D backgroundTexture;
```

قمنا بتعريف المتغير بحيث يكون النوع "Texture2D" . في برمجة الألعاب تدعى الصور بالخامات "Textures" او اشباح "Sprite" , لهذا سميت بهذا الأسم.

المتغير من نوع "Textuere2D" يستخدم للربط مع الصورة , كما أنه يستخدم من اجل عملية التصيير للصورة , او الوصول إلى معلومات الألوان في الصورة (كما سنفعل لاحقا في أحد الدروس القادمة).

بشكل واضح, نحتاج الآن لكي نربط المتغير مع ملف الصورة التي قمنا بإستيرادها لمشروع ال Xna. يتم ذلك من خلال وضع الكود التالي في الدالة "LoadContent" .

```
backgroundTexture = Content.Load<Texture2D> ("background");
```

في هذا السطر يتم ربط المتغير مع الصورة. الوسيط "background" يمثل اسم الملف الذي تم إستيراده, وهو "background.jpg". طبعا لا يهم إذا كانت صورتك تحمل الإمتداد JPG او BMP أو PNG أو حتى اي تنسيق للصور. فقط قم بإستيراد ملف الصورة إلى المشروع و قم بربط الملف مع المتغير.

بعدما قمنا بربط المتغير مع الصوره نحن الآن جاهزين لرسم الصورة على الشاشة. يزودنا ال Xna بأداة سهلة جدا و فعالة جدا من أجل ذلك: SpriteBatch . يمثل أيضا متغير موجود فعليا بشكل مسبق في الكود , حيث يتم إنشائه عند إنشاء مشروع Xnaجديد. بإمكانك البحث عنه في الكود : فهو معرف في اعلى الكود و تم إستهلاله في دالة "LoadContent" .

ال SpriteBatch هي ما يعنيه إسمها فعليا : فهي تتيح لنا القيام بعملية تصيير لحزم من الأشباح "sprites" (الصور ثنائية البعد) . حتى الآن سوف نقوم بعمل تصيير لمره واحدة , لكن بالإمكان إستخدام SpriteBatch واحد لعمل تصيير لكمية كبيرة من الصور. الجيد ان كل ذلك يتم بإستخدام اجزاء التسريع في كرت الشاشة.

بما أن ال SpriteBatch قد تم تجهزيه مسبقا, نستطيع الآن رسم المتغيرات. يجب ان تتم هذه العملية في دالة ال "Draw", لكننا سوف نقوم بعمل دالة صغيرة للقيام بذلك. السبب الوحيد لعمل ذلك هو أن الدالة "Draw" سوف يصبح شكلها فوضوي في نهاية سلسلة الدروس , لأننا سوف نقوم برسم عدد كبير من الصور.

إذن قم بإضافة دالة جديدة في نهاية الكود, بعد دالة ال "()Draw" مباشرة:

```
private void DrawScenery()
{
}
```

عندما نريد أن نرسم صورة ثنائية البعد على الشاشة, بشكل أساسي لدينا خيارين:

- نقوم بتحديد الموقع العلوي الأيسر لزاوية الصورة.
- أو بإمكاننا تحديد مستطيل على الشاشة يمثل المكان الذي نريد أن نرسم الصورة فيه.

بما أننا نريد ان تغطي صورة الخلفية لدينا كامل الشاشة, سوف نختار الخيار الثاني.

سوف نستخدم الخيار الأول في الدرس التالي.

عندما نريد تحديد المستطيل المرتبط بكامل الشاشة سوف نحتاج أن نعرف عرض وطول الشاشة. بشكل واضح بما أننا قمنا بتحديد حجم الشاشة ب 500 * 500 بكسل, نحن نعرف حجم الشاشه فعليا. لكن بالتأكيد نحن نريد ان تقوم لعبتنا بضبط نفسها بشكل تلقائي مهما تغير حجم الشاشة لاحقا.

إذن سوف نقوم بإستخدام العرض و الطول للشاشة الخاصه بنا, حيث سوف نقوم بإستخدام متغيريين لتمثيل الطول و العرض. في أعلى الكود:

```
int screenWidth;
int screenHeight;
```

إضافة إلى ذلك ضع الكود التالي في نهاية الدالة "LoadContent" , حيث سوف يتم تخزين قيم هذه المتغيرات:

```
screenWidth = device.PresentationParameters.BackBufferWidth;
screenHeight = device.PresentationParameters.BackBufferHeight;
```

بما أننا حصلنا الآن على حجم الشاشة, نستطيع العودة الآن إلى الدالة DrawScenery و تعريف المستطيل الذي يغطي كامل الشاشة:

```
Rectangle screenRectangle = new Rectangle(0, 0, screenWidth, screenHeight);
spriteBatch.Draw(backgroundTexture, screenRectangle, Color.White);
```

السطر الأول يقوم بإنشاء المستطيل, زاويته العليا اليسرى في الإحداثيات (0,0) و له العرض و الطول المساوي للشاشة.

ملاحظة هامة:

في برمجة الألعاب , الإحداثي (0,0) يمثل الزاوية العلوية اليسرى للشاشة. المحور X الموجب يمثل الإتجاه **لليمين**, إضافة إلى أن المحور Y الموجب يمثل الإتجاه **للأسفل**.

السطر الأخير يطلب من ال SpriteBatch أن يقوم برسم الصورة ! السطر يطلب من ال SpriteBatch أن يرسم ال backgroundTexture , ممتدة على كامل الشاشة. سوف نقوم بمناقشة الوسيط الأخير في الدرس القادم بإذن الله, لكن بشكل عام يمكنك ان تفترض أن Color.White تعني ان الصورة يجب ان يتم تصييرها بإستخدام ألوانها الأصلية. هذا السطر من الكود يقوم بإضافة صورة الخلفية إلى قائمة التي يجب على ال SpriteBatch ان يقوم برسمها (حتى الآن صورة واحدة).

هذا يقوم (بشكل غير مباشر) برسم الصورة على الشاشة. و لكن يجب أولا أن نقوم بتشغيل "Start" لل SpriteBatch , بعد ذلك عليك عليك إضافة الصور إلى قائمة الصور الخاصة بها, بعدها عليك أن تصدر أمر التصيير للSpriteBatch .

إذن إذهب إلى دالة الرسم "Draw", و تأكد أن الكود لديك مشابه للتالي:

```
protected override void Draw(GameTime gameTime)
{
    graphics.GraphicsDevice.Clear(Color.CornflowerBlue);

    spriteBatch.Begin();
```

```
        DrawScenery();
        spriteBatch.End();

        base.Draw(gameTime);
    }
```

السطر الأول يقوم بمسح الشاشة بلون من إختيارنا. بعدها , قمنا ببدء ال SpriteBatch, إذن نستطيع أن نقوم بإضافة الصور إلى القائمة الخاصه بها من أجل الرسم. بعدها نقوم بإستدعاء الدالة DrawScenery, التي تضيف صورة واحدة إلى القائمة الخاصة بال SpriteBatch . في النهاية , نقوم بإنهاء "End" ال SpriteBatch , التي تقوم بشكل فعلي بالطلب من ال SpriteBatch أن يقوم بعملية الرسم لقائمة الصور على الشاشة. السطر الأخير يقوم بإستدعاء دالة الرسم لأي من عناصر اللعبة "GameComponents" الأخرى المرفقه للعبة.

عندما تقوم بتشغيل هذا الكود, صورة الخلفية يجب أن تظهر على الشاشة, بشكل جميل ☺ , نلاحظ أن الصورة مضغوطة بحيث تلائم حجم الشاشة! الآن عندما تقوم بتغيير حجم الشاشة , سوف تقوم الصورة بضبط حجمها لكي تتلائم مع حجم الشاشة بشكل تلقائي.

الآن بما أن صورة الأرضية الخلفية في مكانها الصحيح , دعنا نقم برسم صورة الأرضية الأمامية. في الدروس التاليه سوف نقوم برسم التضاريس بشكل ديناميكي بحيث أننا سوف نحصل على شكل جديد للتضاريس في كل مرة نشغل فيها البرنامج. الآن سوف نقوم ببساطة بتحميل الصورة التي قمت بحفظها في بداية الدرس, إبدأ بتنفيذ الخطوات الثلاثة السابقة نفسها:

1) قم بإستيراد الصورة "foreground.png" إلي مجلد ال "Content" في ال "Solution Explorer"
2) أضف المتغير في أعلى الكود

```
Texture2D foregroundTexture;
```

3) قم بتجهيز المتغير في الدالة "LoadContent" :

```
foregroundTexture = Content.Load<Texture2D> ("foreground");
```

لتصيير هذه الصورة فوق صورة الأرضية الخلفية, ببساطة قم بإضافة السطر التالي إلى نهاية كود الدالة DrawScencery:

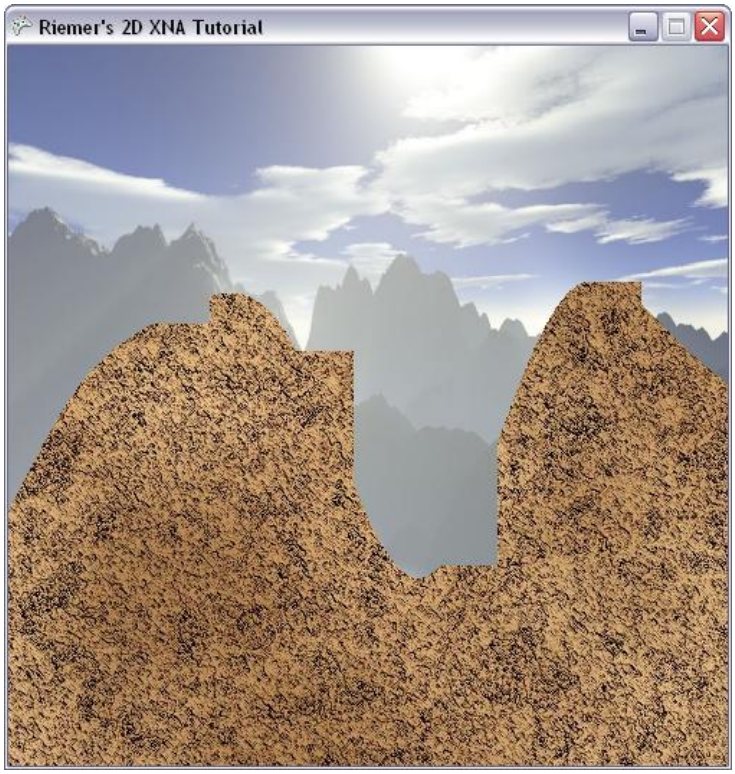
```
spriteBatch.Draw(foregroundTexture, screenRectangle, Color.White);
```

الذي يطلب من ال SpriteBatch ان يرسم ال foregroundTexture , بحيث أنها تغطي كامل الشاشة, بألوانها الأصلية.

الآن عندما تقوم بتشغيل البرنامج , سوف ترى ان صورة الأرضية الأمامية قد تم رسمها بشكل جميل فوق الأرضية الخلفية!

عندما تقوم بفتح الملف foreground.png في الويندوز , تستطيع ان تلاحظ ان الجزء فوق التضاريس شفاف. لهذا قمت بإختيار ملفات ال PNG بدلا من تنسيق ال JPG لأنه لا يستطيع تخزين معلومات المناطق الشفافة. بما أننا قمنا بإضافة صورة الأرضية الخلفية إلى قائمة ال SpriteBatch, و بعدها قمنا بإضافة صورة الأرضية الأمامية, تم رسم صورة الأرضية الخلفية اولا.

عند التنفيذ يجب أن يكون لديك النتيجة التالية:



الكود النهائي الذي وصل إليه المشروع هو

```
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;

        Texture2D backgroundTexture;
        Texture2D foregroundTexture;

        int screenWidth;
        int screenHeight;
```

```
        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 2D XNA Tutorial";

            base.Initialize();
        }

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


           backgroundTexture = Content.Load<Texture2D> ("background");
           foregroundTexture = Content.Load<Texture2D> ("foreground");
           screenWidth =
device.PresentationParameters.BackBufferWidth;
           screenHeight =
device.PresentationParameters.BackBufferHeight;

        }

        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)
        {
            graphics.GraphicsDevice.Clear(Color.CornflowerBlue);

            spriteBatch.Begin();
            DrawScenery();
            spriteBatch.End();

            base.Draw(gameTime);
        }
```

```
        private void DrawScenery()
        {
            Rectangle screenRectangle = new Rectangle(0, 0,
screenWidth, screenHeight);
            spriteBatch.Draw(backgroundTexture, screenRectangle,
Color.White);
            spriteBatch.Draw(foregroundTexture, screenRectangle,
Color.White);
        }
    }
}
```