

الدرس الرابع

أهلاً بكم في الدرس الرابع من سلسلة دروس تعلم الـ Xna , في هذا الدرس سوف نتحدث عن ميزات أكثر في الـ SpriteBatch.

الصورة النهائية في الدرس السابق نتج فيها 3 عيوب أساسية, و التي سوف نقوم بحلها في هذا الدرس إن شاء الله:

- تم رسم المدافع تحت التضاريس و ليس عليها
- المدافع كبيره جدا
- كلها ملونة باللون الرمادي

دعنا نبدأ بالمشكلة الأولى. عندما تستخدم دالة الـ `SpriteBatch.Draw()` , يقوم الـ Xna برسم الركن العلوي الأيسر للصورة في المكان الذي قمت بتحديدده. عندما تقوم بتشغيل البرنامج مرة أخرى, تأكد أن ذلك صحيح.

يمكن حل هذه المشكلة بطريقتين:

- نستطيع ان نخصص أن ترسم الصورة على إحداثي أعلى في المحور Y.
- نستطيع تحديد أن يتم رسم الصورة بناء على الركن السفلي الأيسر في الموقع الذي نريده , بدلا من الركن العلوي الأيسر.

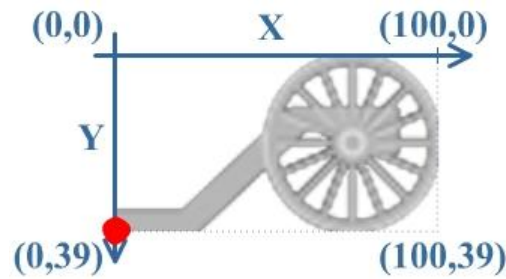
في هذه الحالة سوف نستخدم الحل الثاني. الدالة `SpriteBatch.Draw()` لها عدة اشكال مختلفة "OverLoads" تدعم وسائل أخرى كثيرة. قم باستبدال السطر الذي وضعناه في الدالة `DrawPlayers` بهذا السطر:

```
spriteBatch.Draw(carriageTexture, player.Position, null, Color.White, 0, new Vector2(0, carriageTexture.Height), 1, SpriteEffects.None, 0);
```

دعنا نناقش الوسائط بشكل ملخص:

- (1) الصورة المراد رسمها
- (2) الموقع الذي نريد أن نرسم عليه (على الشاشة)
- (3) هذا الوسيط يسمح لنا بتحديد أي جزء من الصورة يجب رسمها. هذا الخيار يفيدك في حال كنت قد خزنت عدة صور في صورة واحدة كبيرة. بمجرد وضع القيمة `null` هذا يعني أننا نريد ان نرسم كامل الصورة.
- (4) اللون التعديلي "Modulation Color" سوف نقوم بمناقشته لاحقا عند الحديث عن التعامل مع ألوان اللاعبين.
- (5) الوسيط الخامس يتيح لنا تدوير الصورة قبل أن يتم تصييرها. سوف نقوم باستخدامها في الدرس القادم لكي نرسم المدفع.
- (6) سوف نقوم باستخدام الوسيط السادس لكي نحل مشكلة الموقع (التي نحن بصدد حلها). الوسيط السادس يتيح لنا تحديد ما يسمى في الـ Xna نقطة الأصل "origin" الخاصة بالصورة. إفتراضيا و تعتبر نقطة الأصل هي نقطة الركن العلوي الأيسر للصورة. تقوم الـ Xna برسم الصورة بحيث تكون نقطة الأصل لهذه الصورة على الموقع على الشاشة الذي تحدده لرسم الصورة (الذي قمنا بتحديدده في الوسيط الثاني).
- (7) الوسيط السابع يتيح لنا القيام بعمل تحجيم "Scale" (تكبير/تصغير) للصورة. قمنا بوضع القيمة 1 ما يعني أننا نريد ان يتم رسم الصورة بحجمها الأصلي.
- (8) يمكن استخدام الوسيط الثامن لتحديد إذا ما كنا نرغب بقلب الصورة أفقيا أو عموديا قبل رسمها على الشاشة.
- (9) الوسيط الأخير يمكن استخدامه لتحديد الطبقة "Layer" التي سوف يتم الرسم عليها. وهي تقنية مفيدة في برمجة الألعاب المقدمة تتيح لنا عدة صور شفافة مرتبة فوق بعضها البعض.

في الوسيط السادس, قمنا بتحديد الركن الأسفل الأيسر للصورة عن طريق `Vector2(0,carriageTexture.height)`. هذا الموقع ممثل بالنقطة الحمراء في الصورة في الأسفل. سوف يقوم الـ Xna برسم هذا الموقع في الموقع `player.Position`.



الآن عندما نقوم بتشغيل الكود يجب أن ترى أن الركن الأسفل الأيسر مرسوم في موقع أفضل!

دعنا الآن ننقل لنحاول تصحيح المشكلة الثانية , و هي مشكلة حجم المدفع. و هو سهل إلى حد ما, حيث نقوم ببساطة بتحديد معامل التحجيم في الوسيط السابع في سطر الكود السابق. تستطيع أن تجرب على سبيل المثال القيمة `0.5f`, بحيث يتم تصغير حجم الصورة إلى النصف. بكل الاحوال و لأننا سوف نحتاج معامل التصغير هذا لاحقا في الكود سوف نقوم بتخزينه في متغير, قم بتعريف المتغير في أعلى الكود :

```
float playerScaling;
```

و قم بإعطائه قيمته الأولية في نهاية الدالة `:LoadContent`:

```
playerScaling = 40.0f / (float)carriageTexture.Width;
```

بما أن عرض كل منطقة مسطحة في التضاريس يساوي 40 بكسل , إذن يجب على معامل التحجيم هذا ان يقوم بتحجيم صورة حامل المدفع بحيث يتلائم مع المساحة المسطحة.

الآن قم باستخدام هذا المعامل للتحجيم في جملة الدالة `: SpriteBatch.Draw` :

```
spriteBatch.Draw(carriageTexture, player.Position, null,
Color.White, 0, new Vector2(0, carriageTexture.Height),
playerScaling, SpriteEffects.None, 0);
```

أما الآن عند تنفيذ الكود, سوف تلاحظ أن حاملات المدافع قد تم تصغيرها بحيث أنها تتلائم مع مساحة المنطقة المسطحة. بهذا يتبقى لنا أن نلون هذه المدافع !

أو في الحقيقة: حذف بعض الألوان, كما نعرف في برمجة الجرافيكس أن اللون الأبيض يمثل المزج بين جميع الألوان معا. أي لون يتم الوصول إليه من خلال مزج 3 ألوان أساسية: وهي الأحمر و الأخضر و الأزرق (RGB). إذا قمنا بإضافتهم جميعا نصل إلى اللون الأبيض, إذا قمنا بمزج الأحمر مع الأزرق فقط فسوف نحصل على اللون الأرجواني إذا لم تقم باستخدام أي منها سوف نحصل على اللون الأسود.

هذه المعلومات مفيدة لنا لكي نقوم باستخدام الوسيط الخاص باللون في الدالة `SpriteBatch.Draw()`. قبل أن يتم رسم الصورة على الشاشة, تقوم الـ Xna بالمرور على كل بكسل (لون) في الصورة, و اللون في الذي أرسلتها إلى

الدالة `SpriteBatch.Draw()` بعدها, المكون الأحمر لكل من اللونين يتم ضربهما معا. و كذلك الأمر بالنسبة للونين الأخضر و الأزرق. و بعدها يتم رسم اللون الناتج على الشاشة !

دعنا نفترض أنك تريد أن ترسم بكسل بلون أبيض على الشاشة, بمعنى أن مكونات الألوان الأحمر و الأخضر و الأزرق بحيث $RGB=1,1,1$. إضافة إلى ذلك, إفتراض أنك قمت بإرسال اللون الأزرق إلى الدالة `SpriteBatch.Draw()` بمعنى $RGB = 0,0,1$. اللون الناتج سوف يكون $0,0,1 = Blue$ (أزرق).

مثال آخر: إفتراض أننا نريد أن نعرض اللون $0.8, 0.6, 1$ على الشاشة و قد قمت بتحديد اللون $0.5, 0.2, 0.4$ كوسيط للدالة. اللون الناتج سوف يكون $0.8 * 0.5, 0.6 * 0.2, 0.4 * 1 = 0.4, 0.12, 0.4$

واضح انه معقده , ولكن مثالنا سوف يبين أنها ليست كذلك. الصورة الخاصة بحامل المدفع تحتوي على ألوان بيضاء أو رمادية فقط. حاول الآن ان ترسل اللون الأزرق كوسيط اي `Color.Blue` بالشكل التالي :

```
spriteBatch.Draw(carriageTexture, player.Position, null,
Color.Blue, 0, new Vector2(0, carriageTexture.Height),
playerScaling, SpriteEffects.None, 0);
```

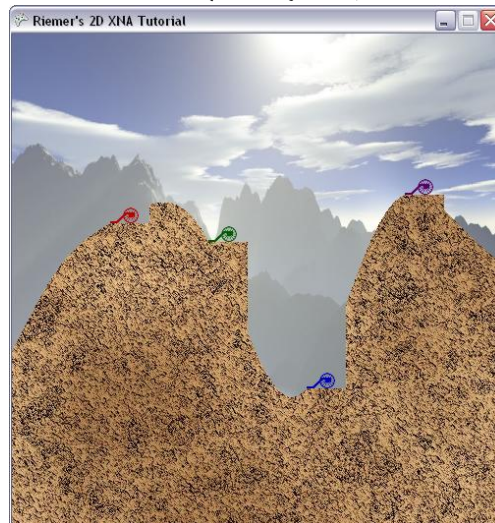
الآن إذا قمت بتشغيل هذا الكود, سوف تلاحظ أن جميع المدافع قد تم رسمها باللون الأزرق. كما تم التفسير سابقا, هذا ببساطة أن المكونات الحمراء و الخضراء للون قد تم حذفها من اللون الأصلي. بحيث تم إستبدال البكسل الملون بالأبيض ببكسل ملون بالأزرق, بينما تم إستبدال اللون الرمادي باللون الأزرق القاتم:

$0.5, 0.5, 0.5 = Grey$ يصبح $0.5, 0.5, 0.5 = Greydarkblue$

الآن بما أننا لا نريد أن يتم رسم جميع المدافع باللون الأزرق, سوف نقوم بأخذ اللون الخاص باللاعب من مصفوفة ال `PlayerData` :

```
spriteBatch.Draw(carriageTexture, player.Position, null,
player.Color, 0, new Vector2(0, carriageTexture.Height),
playerScaling, SpriteEffects.None, 0);
```

إذا قمت بتشغيل الكود الآن سوف تلاحظ أنه تم رسم جميع حاملات المدافع في أماكنها الصحيحة, بألوانها الصحيحة بأحجامها الصحيحة ☺



- بإمكانك تجربة التمرين التالي لكي تمارس ما تعلمته في الدرس:
- حاول قلب حاملات المدافع بشكل أفقي عن طريق تغيير ال SpriteBatch.None في دالة ال SpriteBatch.Draw

كود المشروع 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 struct PlayerData
    {
        public Vector2 Position;
        public bool IsAlive;
        public Color Color;
        public float Angle;
        public float Power;
    }

    public class Game1 : Microsoft.Xna.Framework.Game
    {
        GraphicsDeviceManager graphics;
        SpriteBatch spriteBatch;
        GraphicsDevice device;

        int screenWidth;
        int screenHeight;

        Texture2D backgroundTexture;
        Texture2D foregroundTexture;
        Texture2D carriageTexture;
        Texture2D cannonTexture;

        PlayerData[] players;
        int numberOfPlayers = 4;
        float playerScaling;

        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);
        }
    }
}
```

```

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

        backgroundTexture = Content.Load<Texture2D> ("background");
        foregroundTexture = Content.Load<Texture2D> ("foreground");
        carriageTexture = Content.Load<Texture2D> ("carriage");
        cannonTexture = Content.Load<Texture2D> ("cannon");
        SetUpPlayers();

        playerScaling = 40.0f / (float)carriageTexture.Width;
    }

    private void SetUpPlayers()
    {
        Color[] playerColors = new Color[10];
        playerColors[0] = Color.Red;
        playerColors[1] = Color.Green;
        playerColors[2] = Color.Blue;
        playerColors[3] = Color.Purple;
        playerColors[4] = Color.Orange;
        playerColors[5] = Color.Indigo;
        playerColors[6] = Color.Yellow;
        playerColors[7] = Color.SaddleBrown;
        playerColors[8] = Color.Tomato;
        playerColors[9] = Color.Turquoise;

        players = new PlayerData[numberOfPlayers];
        for (int i = 0; i < numberOfPlayers; i++)
        {
            players[i].IsAlive = true;
            players[i].Color = playerColors[i];
            players[i].Angle = MathHelper.ToRadians(90);
            players[i].Power = 100;
        }

        players[0].Position = new Vector2(100, 193);
        players[1].Position = new Vector2(200, 212);
        players[2].Position = new Vector2(300, 361);
        players[3].Position = new Vector2(400, 164);
    }

    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();
        DrawPlayers();
        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);
    }

    private void DrawPlayers()
    {
        foreach (PlayerData player in players)
        {
            if (player.IsAlive)
            {
                spriteBatch.Draw(carriageTexture, player.Position, null,
player.Color, 0, new Vector2(0, carriageTexture.Height), playerScaling,
SpriteEffects.None, 0);
            }
        }
    }
}
```