

الدرس الثالث

أهلاً بكم في الدرس الثالث من سلسلة دروس تعلم الـ Xna , في هذا الدرس سوف نقوم بتحديد موقع صورة في الشاشة.

في الدرس السابق قمنا بعرض صورتين على كامل الشاشة. من الجلي لنا أنه من أجل عمل لعبة نحن بحاجة إلى عرض صور صغيرة على الشاشة, في الموقع الذي نحدده. و هذا هو هدفنا في هذا الدرس.

في هذا الدرس نحن بصدد عرض حاملات المدافع و المدافع الخاصة باللاعبين. قبل إنتقالنا إلى الكود الخاص بالرسم , يجب علينا أولاً تعريف بعض البيانات عن اللاعبين, مثل موقع اللاعب , لون اللاعب, هل اللاعب حي ام لا. لتخزين كل هذه البيانات معا , يلزمنا تعريف تركيبه "Struct" , الذي يعتبر طريقة نتيج لنا تخزين البيانات التي عرفناها. أضف هذا التركيب في أعلى الكود, مباشرة بعد تعريف فضاء الأسماء "namespace", (إذا لم تكون واثق من الموقع ألق نظرة على الكود في نهاية الصفحة) :

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

كما ترى, أن كائن من النوع PlayerData بإمكانه تخزين الموقع , الذي يمثل بكائن من النوع Vector2. النوع Vector2 يخزن قيمتين , سوف نستخدمهما لتخزين قيمة المحاور X و Y لتحديد مكان عرض مدافع اللاعبين. بعدها , سوف نخزن متغير من نوع منطقي "Boolean" (True/false) للتعبير عن حالة اللاعب إذا كان على قيد الحياة أم لا (بمعنى اخر هل سوف يتم رسمه على الشاشة أم لا), كما يتم الإحتفاظ باللون الذي سوف نلون المدفع الخاص باللاعب به, و أخيراً لكل لاعب يجب أن نحتفظ بالزاوية للمدفع , و الطاقة التي سوف يطلق المدفع بها.

الآن يجب علينا أن نحتفظ بمصفوفة كائنات من النوع PlayerData. المصفوفة سوف تقوم بتخزين قائمة من العناصر سوف تساعدنا على الوصول إلى كائن محدد فيها, أو عمل تغييرات على كائن فيها. إذن قم بإضافة تعريف المتغيرين التاليين في أعلى الكود, فوق الدالة "Game1()" :

```
PlayerData[] players;
int numberOfPlayers = 4;
```

عندما ننتهي من الكود , سيكون من السهل علينا تعديل عدد اللاعبين , المتمثل بالمتغير الأخير.

الآن دعنا نقوم بعمل دالة بإسم , "SetUpPlayers" بحيث تقوم بإعطاء القيم الإستهلاكية للمصفوفة.

```
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;
}
}

```

لقد بدأنا بتعريف مصفوفة من الألوان من النوع "Color"، حيث حددنا 10 ألوان قمنا بإختيارها. بعدها السطر قبل للتكرار يقوم بشكل فعلي بإنشاء المصفوفة من النوع PlayerData. بما أننا قمنا بتعريف عدد اللاعبين ب 4
 numberOfPlayers=4 , يقوم هذا السطر بإنشاء مصفوفة تحتوي على 4 كائنات فارغة من النوع PlayerData.

يتم التكرار بالمرور على كل عنصر في المصفوفة. بحيث نقوم بوضع قيمة ال IsAlive ل true, كما نقوم بإسناد قيمة اللون بناء على الألوان التي اخترناها سابقا. في برمجة الألعاب, كما هو الحال في ال Xna , كل الزوايا يتم تمثيلها بالتقدير الدائري "Radians". ربما يكون هذا صعبا بعض الشيء , ولكن إذا كنت تعلم أن ال (Pi=3.14) يقابل الزاوية 180, تستطيع بعده إيجاد قيمة التقدير الدائري لأي زاوية تريدها. ال Xna قامت بتسهيل الأمور بشكل أكبر بحيث أنها هي من تقوم بالحسابات عنا: كل ما يلزمنا هو تمرير قيمة الزاوية إلى الدالة "MathHelper.ToRadians()" فيقوم ال بعملية التحويل.

بما أن صورة المدفع موجهة للأعلى أصلا , و نحن نريد أن تبدأ اللعبة بحيث تكون كل المدافع موجهة لليمين , سوف نحتاج إلى تدوير المدافع 90 درجة (لمعلوماتك, ال 90 درجة تساوي $\frac{\pi}{2}$, أي أن $\text{MathHelper.ToRadians}(90)$ هي نفسها $\text{MathHelper.PiOver2}$). أخير نضع القيمة الإبتدائية للقوة الخاصة بالمدفع إلى 100.

هناك شيء مهم يجب أن نقوم بتحضيره : هو موقع اللاعبين على اليابسة. كما ترى أن صورة الأرضية الأمامية (التضاريس) تحتوي على 4 مناطق مسطحة, مناسبة تماما لكي يتم وضع المدفع عليها. عندما سوف نقوم بإنشاء التضاريس بأنفسنا في دروس قادمة , سوف يكون موقع هذه المناطق المسطحة معروفة لدينا. حتى الآن, سوف نقوم بكتابتهم يدويا. هذه المناطق المسطحة الأربعة على اليابسة (ضع الكود التالي في نهاية الدالة `SetUpPlayers`):

```

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);

```

الآن دعنا لا ننسى ان نستدعي هذه الدالة في نهاية دالة `LoadContent` :

```

SetUpPlayers();

```

أما الآن كل البيانات عن اللاعبين جاهزة و معروفة, بهذا يكون قد حان الوقت لرسم اللاعبين على الشاشة. تستطيع أن تقوم بتنزيل صورة حامل المدفع من الرابط التالي [هنا](#) وصورة المدفع من الرابط [هنا](#) , كما سنقوم بعمل الخطوات الثلاث نفسها في الدرس السابق :

- (1) قم بإستيراد كلا من الصورتين إلى مجلد ال Content في ال "Solution Explorer"
- (2) اصف المتغيرات التالية في أعلى الكود:

```
Texture2D carriageTexture;  
Texture2D cannonTexture;
```

- (3) قم بتجهيز المتغيرات في دالة ال LoadContent :

```
carriageTexture = Content.Load<Texture2D> ("carriage");  
cannonTexture = Content.Load<Texture2D> ("cannon");
```

في النهاية, يجب علينا إضافة هذه الصور إلى قائمة الصور التي على ال spriteBatch رسمها. لكي نحافظ على دالة ال Draw مرتبة, سوف نقوم بعمل دالة جديدة بإسم DrawPlayers, و التي هي ببساطة :

```
private void DrawPlayers()  
{  
    foreach (PlayerData player in players)  
    {  
        if (player.IsAlive)  
        {  
            spriteBatch.Draw(carriageTexture, player.Position,  
Color.White);  
        }  
    }  
}
```

يمكن المرور على المصفوفة بطريقتين. إذا كنا نريد التعديل على البيانات في داخل المصفوفة , نستخدم ال for loop العادية, كما فعلنا في دالة ال SetUpPlayers. و لكن هذه المرة نحن نحتاج إلى قراءة البيانات من داخل مصفوفة اللاعبين (مثل موقع اللاعب) , إذن بإستطاعتنا إستخدام الصيغة foreach loop لأنها مقرونة بشكل أكثر من التكرار العادي. الكود في هذه الدالة ببساطة يقول : لكل لاعب في اللعبة , إحص إذا ما كان على قيد الحياة؛ قم برسم المدفع الخاص به في موقع هذا اللاعب.

الآن دعنا لا ننسى إضافة الإستدعاء لهذه الدالة من دالة الرسم الخاصة بنا Draw . بما أننا قمنا بإضافة الصور إلى ال spriteBatch , إذن نحتج إلى إستدعاء الدالة بين جمليتي spriteBatch.Begin() و spriteBatch.End() .

بما أن صورة الخلفية ممتلئة تماما و ليست شفافة. نحتاج إلي وضع الإستدعاء بعد إستدعاء دالة ال DrawScenery , بهذا سوف يتم رسم المدافع فوق المشهد و ليس في موقع آخر:

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

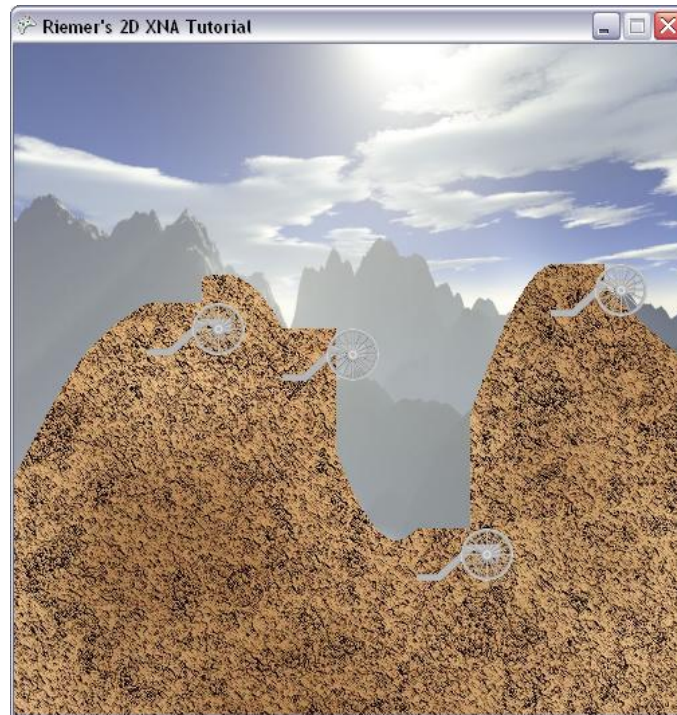
```

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

base.Draw(gameTime);
}

```

الآن قم بتنفيذ ما عملناه حتى الآن , يجب ان يكون لديك نتيجة مشابهة للصورة التالية:



- كما ترى أن المحور X للمدافع الثلاثة يبدو صحيحا, لا بد أنك لاحظت ثلاثة أخطاء:
 - تم رسم المدافع تحت التضاريس و ليس عليها.
 - المدافع كبيره جدا
 - كلها ملونة باللون الرمادي!
- سوف نقوم بحل هذه المشاكل في الدرس القادم بإذن الله.

كود المشروع حتى هذه 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 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;

        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();
        }

        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, Color.White);
            }
        }
    }
}

```