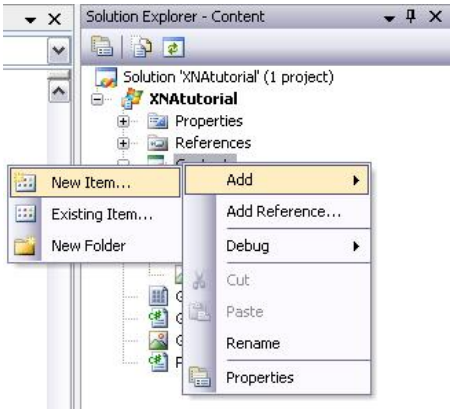
الدرس السابع

أهلا بكم في الدرس السابع من سلسلة دروس تعلم ال Xna , سوف نقوم في هذا الدرس بكتابة نص على الشاشة.

في هذا الدرس سوف نتعلم كيفية الكتابة على الشاشة, بحيث يصبح بإمكاننا كتابة قيمة الطاقة و الزاوية لمدفع اللاعب الحالي.

لقد اصبحت الكتابة على الشاشة في ال Xna عملية سهلة جدا. أولا, يجب أن نقوم بإستيراد خط "Font" إلى المشروع الخاص بنا و ربطه بمتغير داخل الكود, بنفس الطريقة التي قمنا فيها بإضافة الصورة إلى المشروع. بعدها يمكننا ببساطة أن نقوم برسم النص على الشاشة بإستخدام ال SpriteBatch. لا يلزمنا أيضا ان نقوم بإنشاء SpriteBatch جديد : لإنه يمكننا ببساطة أن نستخدم نفس ال SpriteBatch الذي استخدمناه لرسم المدافع و التضاريس.

دعنا نبدأ بإضافة الخط إلى المشروع. لعمل ذلك, نقم بالضغط بالزر اليمين على مجلد ال Content في ال Solution Explorer. بعدها إختر من القائمة Add-> New Item , كما يظهر في الصورة التالية.



في مربع الحوار, قم بإختيار ال "Sprite Font" و اعطه اسما في الأسفل : myFont.spritefont و بعدها إضغط على زر إضافة "Add". ستلاحظ أن ملف ال myFont.spritefont قد تم إضافته إلى مجلد ال Content في المشروع.

بعد ذلك سوف يظهر لديك ملف نص(ملف Xml عمليا) غريب (ما غريب إلا الشيطان ☺). لحسن الحظ, لن نغير كثيرا في هذا الملف. إبحث عن خانة إسم الخط "FontName" في الملف, و قم بتغيره من myFont إلى Arial. بهذا سوف تقوم بتعريف عائلة الخط, كما سيمكنك من إختيار اي خط موجود على جهازك. لكي ترى ما هي الخطوط المثبته على جهازك, إفتح المجلد C:\Windows\Fonts. لاحظ أيضا انه من خلال ملف النص نفسه بإمكانك أن تقوم بتغيير حجم الخط. إفتراضيا الحجم هو 14 نقطة.

بعدها, إرجع إلي ملف ال Game1.CS قم بإضافة المتغير التالي في أعلى الكود:

```
SpriteFont font;
```

كما أن الكائن Texture2D يمكن ربطه مع ملف صورة, بإمكاننا أن نربط الكائن SpriteFont مع ملف خط. قم بتحميل الملف في المتغير الذي عرفناه , داخل الدالة LoadContent :

```
font = Content.Load<SpriteFont> ("myFont");
```

العملية شبيهة بتحميل الصورة: حيث قمنا بإستخدام الدالة Content.Load , بعدها حددنا نوع المتغير الذي نريد تحميلة, إضافة إلى القسم الأول من إسم الملف.

الآن و بعد أن تم تحميل الخط, و ال SpriteBatch مفعل, نحن جاهزين لكي نقوم برسم بعض النصوص.مره أخرى سوف نقوم بوضع هذا الكود في دالة الرسم الخاصة بنا و لكن سنقوم بذلك في دالة منفصلة:

```
 private void DrawText()
 {
     spriteBatch.DrawString(font, "Cannon power: 100", new Vector2(20,
45), Color.White);
 }
```

هذا السطر من الكود يطلب من ال SpriteBatch أن يرسم بعض النص على الشاشة بإستخدام كائن الخط font. قمنا بتحديد الموقع, اللون و النص, للتعبير عن قوة المدفع. سوف يتم رسم النص على بعد 20 بكسل يمين الشاشة و 45 من الأعلى, (من الركن الأعلى الأيسر للشاشة). طبعا لاتنسى أن تستدعي هذه الدالة من داخل دالة الرسم الأصليه Draw() :

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

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

     base.Draw(gameTime);
 }
```

الآن عند تشغيل الكود, سوف تشاهد مؤشر الطاقة موجود على الركن الأيسر العلوي!

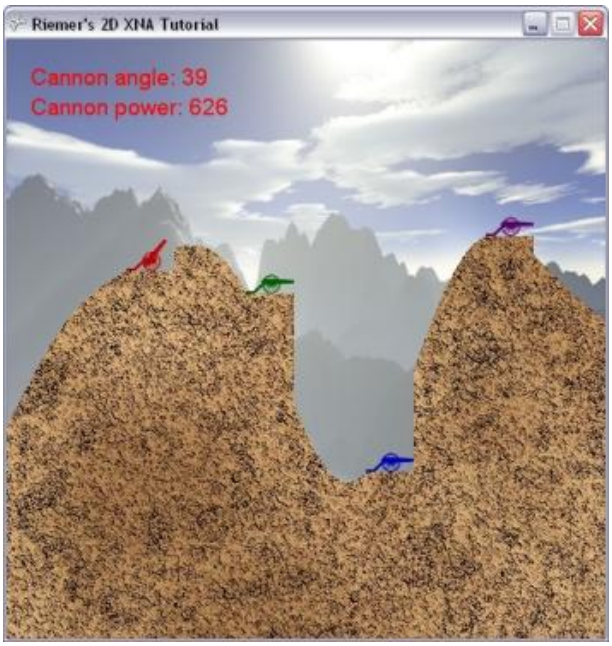
من الواضح أن الرقم الظاهر على الشاشة لا يعبر عن قيمة الطاقة الفعلية, لأننا قمنا بوضع رقم ثابت. بدلا من ذلك قم بتغير دالة DrawText كالتالي:

```
 private void DrawText()
 {
     PlayerData player = players[currentPlayer];
     int currentAngle = (int)MathHelper.ToDegrees(player.Angle);
     spriteBatch.DrawString(font, "Cannon angle: " +
currentAngle.ToString(), new Vector2(20, 20), player.Color);

     spriteBatch.DrawString(font, "Cannon power: " +
player.Power.ToString(), new Vector2(20, 45), player.Color);
 }
```

هكذا يتم قراءة معلومات اللاعب من مصفوفة ال PlayerData. من هذه البيانات إستخدمنا كل من الزاوية و الطاقة و اللون من أجل كتابة النص.

هذا كل شيئ ☺ حاول تشغيل البرنامج. سوف تظهر لك الشاشة مثل الصورة في الأسفل. حاول تغيير قيم الزاوية و الطاقة : سوف تتغير القيم الموجوده على الشاشة بناء على القيم الجديدة.



- التمرين هذا اليوم (لي و للقراء): هو محاولة الكتابة باللغة العربية, و اللغات الغير لاتينية, و اللغات من اليمين لليسار.

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;
        SpriteFont font;

        PlayerData[] players;
```

```
        int numberOfPlayers = 4;
        float playerScaling;
        int currentPlayer = 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 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");

           font = Content.Load<SpriteFont> ("myFont");

            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();

            ProcessKeyboard();
```

```
            base.Update(gameTime);
        }

        private void ProcessKeyboard()
        {
            KeyboardState keybState = Keyboard.GetState();
            if (keybState.IsKeyDown(Keys.Left))
                players[currentPlayer].Angle -= 0.01f;
            if (keybState.IsKeyDown(Keys.Right))
                players[currentPlayer].Angle += 0.01f;

            if (players[currentPlayer].Angle > MathHelper.PiOver2)
                players[currentPlayer].Angle = -MathHelper.PiOver2;
            if (players[currentPlayer].Angle < -MathHelper.PiOver2)
                players[currentPlayer].Angle = MathHelper.PiOver2;

            if (keybState.IsKeyDown(Keys.Down))
                players[currentPlayer].Power -= 1;
            if (keybState.IsKeyDown(Keys.Up))
                players[currentPlayer].Power += 1;
            if (keybState.IsKeyDown(Keys.PageDown))
                players[currentPlayer].Power -= 20;
            if (keybState.IsKeyDown(Keys.PageUp))
                players[currentPlayer].Power += 20;

            if (players[currentPlayer].Power > 1000)
                players[currentPlayer].Power = 1000;
            if (players[currentPlayer].Power < 0)
                players[currentPlayer].Power = 0;
        }

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

            spriteBatch.Begin();
            DrawScenery();
            DrawPlayers();
            DrawText();
            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)
                {
                    int xPos = (int)player.Position.X;
                    int yPos = (int)player.Position.Y;
                    Vector2 cannonOrigin = new Vector2(11, 50);

                    spriteBatch.Draw(cannonTexture, new Vector2(xPos + 20, yPos - 10), null,
player.Color, player.Angle, cannonOrigin, playerScaling, SpriteEffects.None, 1);
                    spriteBatch.Draw(carriageTexture, player.Position, null, player.Color, 0, new
Vector2(0, carriageTexture.Height), playerScaling, SpriteEffects.None, 0);
                }
            }
        }

        private void DrawText()
        {
            PlayerData player = players[currentPlayer];
            int currentAngle = (int)MathHelper.ToDegrees(player.Angle);
            spriteBatch.DrawString(font, "Cannon angle: " + currentAngle.ToString(), new Vector2(20,
20), player.Color);
            spriteBatch.DrawString(font, "Cannon power: " + player.Power.ToString(), new Vector2(20,
45), player.Color);
        }
    }
}
```