

الدرس العاشر

أهلاً بكم في الدرس العاشر من سلسلة دروس تعلم الـ Xna , سوف نقوم في هذا الدرس بعمل ذيل من الدخان للصاروخ.

هذا الدرس سوف يكون لطيفاً إذا قمت بإتباعه بشكل صحيح, سوف نقوم بعمل تدريب جميل. الفكرة ببساطة: في كل موقع للصاروخ, سوف نقوم بإضافة بعض من الدخان.

إبدأ بتنزيل صورة الدخان من الرابط [هنا](#) (أو من الملفات المرفقة) و إستيراد الصورة إلى داخل مشروع الـ Xna. قم بإضافة المتغير المرتبط بالصورة في أعلى الكود:

```
Texture2D smokeTexture;
```

وقم بتجهيز المتغير في دالة الـ LoadContent :

```
smokeTexture = Content.Load<Texture2D> ("smoke");
```

هناك شيء واحد قد يجعل المهمة صعبة بعض الشيء: أننا لا نعرف بالتحديد كم من جزيئات الدخان سوف نرسم, إذن نحن لا نعرف كم موقع يلزمنا أن نخزن ولكن ذلك سهل الحل بواسطة الميزة في الـ NET. المسامة : List . القائمة "List" هي مجموعة ولكن لا يلزمك معرفة أو تحديد حجمها: فقط قم بإضافة أو حذف عناصر إلى القائمة متى شئت. إذن قم بإضافة الكود التالي في أعلى الكود الخاص بنا:

```
List<Vector2> smokeList = new List<Vector2> ();  
Random randomizer = new Random();
```

لقد قمنا بتحديد هذه القائمة لكي تستخدم لتخزين كائنات من النوع Vector2. أيضاً قمنا بتجهيزها لتكون قائمة فارغة. المتغير الثاني مفيد جداً لنا وفي برمجة الألعاب بشكل عام: لأنه يقوم بإنشاء أرقام عشوائية كلما طلبنا ذلك منه. لإننا سوف نقوم فعلياً بإستخدام أرقام عشوائية في هذا الدرس, كما سوف نقوم بإستخدام إضافة بعض الانحراف العشوائي في موقع جزيئات الدخان.

الآن سوف نذهب إلى الدالة UpdateRocket. هذه الدالة مكان مناسب جداً لكي نقوم بتخزين مواقع في القائمة, بما أن هذه القائمة يتم إستدعائها 60 مره في الثانية. قم بإضافة الكود التالي في نهاية كتلة الـ If في الدالة:

```
Vector2 smokePos = rocketPosition;  
smokePos.X += randomizer.Next(10) - 5;  
smokePos.Y += randomizer.Next(10) - 5;  
smokeList.Add(smokePos);
```

هذا الكود يقوم بقراءة الموقع الحالي للصاروخ, و إضافة رقم عشوائي لكل من الـ X و الـ Y. تم الطلب من الـ randomizer ان يعطينا رقم عشوائي بين الـ 0 و الـ 10, و قمنا بطرح 5 من الناتج, ليرتج لدينا رقم بين الـ -5 و الـ 5. الموقع الناتج سوف يتم تخزينه في القائمة smokeList !

لئلا نحتاجه الآن, هو رسم صورة الدخان في كل موقع مخزن في القائمة smokeList. لعمل ذلك سوف نقوم بإضافة دالة جديدة و بسيطة بإسم DrawSmoke :

```
private void DrawSmoke()
{
    foreach (Vector2 smokePos in smokeList)
        spriteBatch.Draw(smokeTexture, smokePos, null, Color.White, 0, new
        Vector2(40, 35), 0.2f, SpriteEffects.None, 1);
}
```

لكل موقع مخزن في القائمة smokeList, سوف نقوم برسم ال smokeTexture. بما أننا لن نقوم بتعديل أي من المواقع في القائمة, بإمكاننا استخدام الصيغة foreach للتكرار بدلاً من ال for-loop العادية.

لا تنسى استدعاء هذه الدالة من داخل دالة Draw الأصلية:

```
DrawSmoke();
```

الآن حاول تشغيل الكود! عندما تطلق الصاروخ, يجب أن ترى أن ذيل الدخان خلف الصاروخ. سوف نحتاج إلى المزيد من الدخان, إذن بدل من إضافة جزيئ واحد في المرة الواحدة, سوف نقوم بإضافة 5:

```
for (int i = 0; i < 5; i++)
{
    Vector2 smokePos = rocketPosition;
    smokePos.X += randomizer.Next(10) - 5;
    smokePos.Y += randomizer.Next(10) - 5;
    smokeList.Add(smokePos);
}
```

إلى هنا كل شيء في هذا الدرس!



كودنا 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;
        Texture2D rocketTexture;
        Texture2D smokeTexture;
        SpriteFont font;

        PlayerData[] players;
        int numberOfPlayers = 4;
        float playerScaling;
        int currentPlayer = 0;

        bool rocketFlying = false;
        Vector2 rocketPosition;
        Vector2 rocketDirection;
        float rocketAngle;
        float rocketScaling = 0.1f;

        List<Vector2> smokeList = new List<Vector2> ();
        Random randomizer = new Random();

        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");
        rocketTexture = Content.Load<Texture2D> ("rocket");

        smokeTexture = Content.Load<Texture2D> ("smoke");

        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();
        UpdateRocket();

        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;

        if (keybState.IsKeyDown(Keys.Enter) || keybState.IsKeyDown(Keys.Space))
        {
            rocketFlying = true;
            rocketPosition = players[currentPlayer].Position;
            rocketPosition.X += 20;
            rocketPosition.Y -= 10;
            rocketAngle = players[currentPlayer].Angle;
            Vector2 up = new Vector2(0, -1);
            Matrix rotMatrix = Matrix.CreateRotationZ(rocketAngle);
            rocketDirection = Vector2.Transform(up, rotMatrix);
            rocketDirection *= players[currentPlayer].Power / 50.0f;
        }
    }

    private void UpdateRocket()
    {
        if (rocketFlying)
        {
            Vector2 gravity = new Vector2(0, 1);
            rocketDirection += gravity / 10.0f;
            rocketPosition += rocketDirection;
            rocketAngle = (float)Math.Atan2(rocketDirection.X, -rocketDirection.Y);

            for (int i = 0; i < 5; i++)
            {
                Vector2 smokePos = rocketPosition;
                smokePos.X += randomizer.Next(10) - 5;
                smokePos.Y += randomizer.Next(10) - 5;
                smokeList.Add(smokePos);
            }
        }
    }

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

        spriteBatch.Begin();
        DrawScenery();
        DrawPlayers();
        DrawText();
        DrawRocket();
        DrawSmoke();
        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);
}

private void DrawRocket()
{
    if (rocketFlying)
        spriteBatch.Draw(rocketTexture, rocketPosition, null, players[currentPlayer].Color,
rocketAngle, new Vector2(42, 240), 0.1f, SpriteEffects.None, 1);
}

private void DrawSmoke()
{
    foreach (Vector2 smokePos in smokeList)
        spriteBatch.Draw(smokeTexture, smokePos, null, Color.White, 0, new Vector2(40, 35),
0.2f, SpriteEffects.None, 1);
}
}
```