

الدرس الثامن

أهلاً بكم في الدرس الثامن من سلسلة دروس تعلم الـ Xna , سوف نقوم في هذا الدرس بإطلاق الصاروخ (إنبسطو على الأرض 😊).

في هذا الدرس, سوف نقوم بتوسيع الكود الخاص بنا بحيث نقوم بإطلاق الصاروخ من المدفع عند الضغط على زر المسافة "Spacebar". أولاً قم بتنزيل صورة الصاروخ من الرابط [هنا](#) (أو من خلال المرفقات) , و قم بإستيراده إلى داخل مشروع الـ Xna . أضف المتغير التالي في أعلى الكود أيضاً:

```
Texture2D rocketTexture;
```

وقم بتجهيز المتغير في دالة الـ LoadContent:

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

بعدها, اضع المتغيرات التالية في أعلى الكود أيضاً:

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

المتغير الأول يعبر عما إذا كان الصاروخ في الهواء حالياً أم لا, بمعنى هل سوف يتم رسمه أم لا.

من الواضح أننا بحاجة لمعرفة موقع الصاروخ في كل الأوقات. اما بالنسبة لإتجاه الصاروخ فيلزمنا من أجل حساب الموقع الجديد للصاروخ, و هو مرتبط مباشرة بالزاوية. في الحقيقة يمكننا إستقاق (حساب) الزاوية بناء على الإتجاه, ولكن لأننا سوف نحتاج الوصول إليها بشكل متكرر قمنا بوضعها في متغير منفصل. أخيراً, متغير التحجيم "Scaling" يمثل عدد ثابت, و لكن لأننا سوف نحتاجه أكثر من مره داخل الكود قمنا بوضعه في متغير.

دعنا نبدأ بمعالجة ضغطة الزر من خلال الدالة الـ ProcessKeyboard , سوف نحتاج لإطلاق الصاروخ عندما يتم الضغط على الزر في لوحة المفاتيح. إذن قم بإضافة الكود التالي إلى الدالة :

```
if (keybState.IsKeyDown(Keys.Enter) ||
keybState.IsKeyDown(Keys.Space))
{
    rocketFlying = true;
}
```

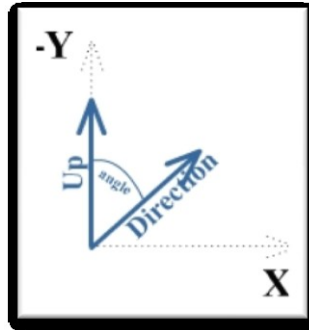
حيث يقوم هذا الكود بإعطاء قيمة true للمتغير rocketFlying حتى الآن. بعدها, يمكننا ان نعطي قيم الموقع و الزاوية للصاروخ, لكي تكون مساوية للقيم الخاصة بالمدفع الذي يطلق الصاروخ. قم بإضافة الكود التالي داخل كتلة الـ If السابقة:

```
rocketPosition = players[currentPlayer].Position;
rocketPosition.X += 20;
rocketPosition.Y -= 10;
rocketAngle = players[currentPlayer].Angle;
```

بالنسبة للموقع, بدئنا من الركن الأيسر الأسفل الخاص بحامل المدفع, و أضفنا 20 بسك إلى اليمين و 20 بسك للأعلى, ليكون مقابل مركز حامل المدفع. كقيمة ابتدائية للزاوية يجب علينا أن نأخذ الزاوية الحالية الخاصة بالمدفع.

إيجاد الإتجاه الصاروخ أعقد قليلا. الإتجاه مكون من مكونين هما ال X و ال Y, للتعبير عن كم من بسك يجب أن يتحرك الصاروخ أفقيا و عموديا في كل مره. بشكل واضح يمكن إيجاد ذلك من خلال الزاوية الخاصة بالصاروخ.

أحد الطرق تقتضي أن نأخذ جيب الزاوية "Sine" و جيب التمام لها "Cosine" و استخدامهما للتعبير عن قيم ال X و ال Y, و لكن ال Xna تتيح لنا استخدام طريقه (حل) أكثر أناقة. سوف نبدأ من الإتجاه إلى الأعلى (0,1), وبعدها سنطلب من ال Xna أن يقوم بعمل التدوير بزاوية نقوم نحن بتحديددها. كما هو موضح في الصورة التالية:



دعنا نبدأ بتعريف متجه "Vector" يعبر عن المحور Y السالب, و له الإحداثيات (0,-1), ضع السطر التالي بعد الكود السابق:

```
Vector2 up = new Vector2(0, -1);
```

بعدها, نريد من ال Xna أن يقوم بتدوير هذا المتجه. لعمل ذلك يلزمنا مصفوفة تدوير, ذلك يبدو معقدا, ولكنه ليس كذلك: المصفوفة هي "شيئ" نقوم بضربه في المتجه ليعطينا نسخة جديده و محولة "Transformed" من هذا المتجه. إذا كان بإمكان ال Xna أن يعطينا مصفوفة تحتوي على التدوير المطلوب, نستطيع بعدها أن نستخدم هذه المصفوفة من أجل تدوير المتجه. هذا السطر يتيح لنا إنشاء مصفوفة التدوير:

```
Matrix rotMatrix = Matrix.CreateRotationZ(rocketAngle);
```

لتفسير السطر السابق, فكر كيف بإمكاننا تدوير المتجه الموجود في الصورة السابقة. في يدك اليسرى دع إصبع الإبهام يشير إلى محور X و أصبع السبابة إلى محور Y السالب, و بقية الأصابع إلى داخل الشاشة, بهذا يشكل بقية الأصابع محور الدوران.

هذا يفيدنا في فهم السطر السابق: فهو يقوم بإنشاء مصفوفة للدوران حول محور Z. هذا ما نحتاجه: إذا كان محور X يشير إلى اليمين و محور Y يشير إلى الأسفل, سيكون محور Z متجه إلى خارج الشاشة!

الآن لدينا هذه المصفوفة تحتوي على التدوير الذي نريده, نستطيع ان نطلب من ال Xna أن يقوم بتحويل المتجه بهذا التدوير:

```
rocketDirection = Vector2.Transform(up, rotMatrix);
```

هذا السطر سوف يأخذ الإتجاه الأصلي المتجهه للأعلى, و يقوم بتدويره على محور Z, و يخزن النتيجة في المتغير rocketDirection! الآن كل ما نحتاجه, هو أن نقوم بزيادة أو إنقاص الإتجاه بالاعتماد على طاقة إطلاق الصاروخ:

```
rocketDirection *= players[currentPlayer].Power / 50.0f;
```

في هذه اللحظة، قمنا بتخزين كل شيء نحتاجه من أجل الصاروخ، لذا دعنا نقوم بإنشاء دالة صغيرة تقوم برسم الصاروخ على الشاشة:

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

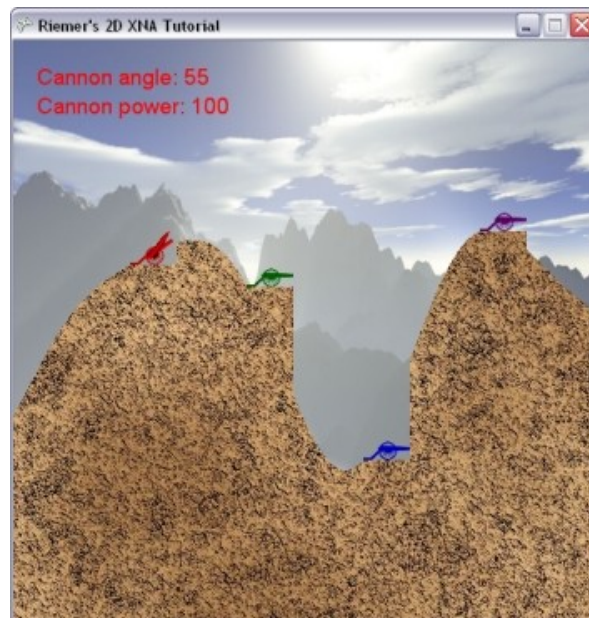
السطر الأول يفحص فيما إذا تم إطلاق الصاروخ. السطر الثاني يطلب من ال SpriteBatch أن يقوم برسم:

- صورة الصاروخ
- في موقع rocketPosition على الشاشة
- كل الصورة
- بلون اللاعب الحالي
- بعد تدويرها بزاوية rocketAngle
- باعتبار ان النقطة (42,240) هي مركز الصورة (انظر إلى هذا الموقع في صورة الصاروخ الأصلية)
- مصغرة ب 10 مرات
- بدون قلبها
- على الطبقة على بعد 1

لا تنسى إستدعاء هذه الدالة من الدالة Draw() :

```
DrawRocket();
```

الآن عند تنفيذ الكود، إذا ضغطت على زر المسافة "Space" أو زر "Enter"، ستري أنه تم عرض الصاروخ على المدفع! ولكن بما أننا لم نقوم بتحديث موقع الصاروخ، لن يتحرك الصاروخ بعد، و لكن من السهل عمل ذلك و هذا ما سنفعله في الدرس التالي بما أننا قمنا بحساب الإتجاه للصاروخ !



كود المشروع حتى 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;
        Texture2D rocketTexture;
        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;

        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");

        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;

        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;
        }
    }

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

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

```