

الدرس السادس

أهلاً بكم في الدرس السادس من سلسلة دروس تعلم الـ Xna , سوف نقوم في هذا الدرس بقراءة مدخلات لوحة المفاتيح.

في هذا الدرس سوف نرى كم هو من السهل أن نقرأ مدخلات لوحة المفاتيح في الـ Xna. كمثال : سيتم في حال قام المستخدم بالضغط على زر السهم الأيمن أو الأيسر, سوف يتم تغيير الزاوية الخاصة بمدفع اللاعب الحالي. دعنا نبدأ بإضافة المتغير التالي في أعلى الكود الخاص بنا, بحيث يمكننا من معرفة على أي من المدافع سوف نقوم بالتعديل:

```
int currentPlayer = 0;
```

قراءة مخرجات (او مدخلات) لوحة المفاتيح يجب أن تتم بشكل متواصل إلى حد ما, كما أنها تستخدم لتحديث المنطق الخاص باللعبة مثل زاوية المدفع. بناء على ذلك يجب علينا أن نقوم بهذه العملية في داخل الدالة Update. للحفاظ على الدالة Update مرتبة سوف نقوم بعمل دالة خاصة بنا ProcessKeyboard و إستدعائها من داخل دالة التحديث Update.

```
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;
}
```

ببساطة هذا الكود هو كل ما نحتاج! في السطر الأول يقرأ الحالة الحالية للوحة المفاتيح, يفحص ما هو المفتاح المضغوط حالياً. قمنا بإستعمال الكائن KeyboardState للفحص فيما إذا كان زر السهم الأيمن أو الأيسر مضغوط, في حال كان أحدهما نقوم بتغيير الزاوية الخاصة باللاعب الحالي.

الآن دعنا لا ننسى إضافة الإستدعاء لهذه الدالة من داخل دالة الـ Update, لتأكد أنه قد تم قراءة لوحة المفاتيح 60 مرة في الثانية الواحدة:

```
ProcessKeyboard();
```

قم بتشغيل الكود! عندما تضغط على أزرار السهم الأيمن أو الأيسر, سوف يتم تدوير المدفع الخاص باللاعب الأول (الأحمر). لاحظ أنه بإمكانك تدوير المدفع بشكل مستمر بحيث أنها سوف تصطدم بالأرض, هذا يعني أن علينا أن نقوم بإضافة بعض الكود في نهاية دالة الـ ProcessKeyboard :

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

في هذا الكود, يجب أن نتذكر أن الـ $\pi = 3.14$ مقابلة للزاوية 180, إذن PiOver2 مقابلة للزاوية 90 درجة. إذن ببساطة: إذا كان المدفع بزاوية 90 درجة (بمعنى أنه موجه أفقياً إلى اليمين), و تم ضغط زر السهم الأيمن, سوف تتغير الزاوية إلى -90 درجة بمعنى أن المدفع سوف يشير إلى اليسار. السطرين الأخيرين يقومان بنفس العمل و لكن للجهة الأخرى قم بتنفيذ الكود ولاحظ!

الآن سوف نقوم بإضافة مجموعه أخرى من الكود للدالة ProcessKeyboard, لكي نتيح للاعب أن يتحكم و يغير قوة المدفع الخاص به:

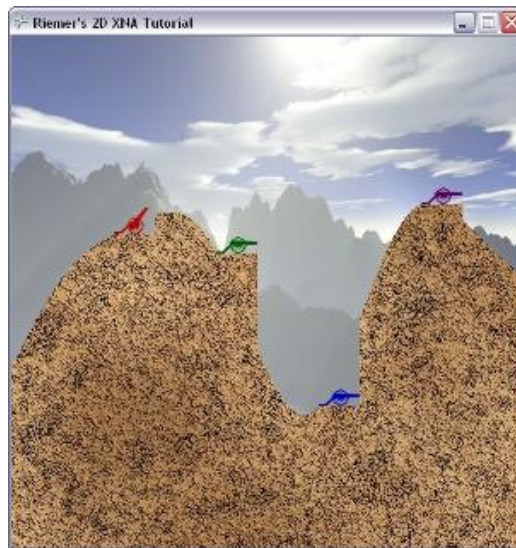
```
if (keybState.IsKeyDown(Keys.Down))
    players[currentPlayer].Power -= 1;
if (keybState.IsKeyDown(Keys.Up))
    players[currentPlayer].Power += 1;
if (keybState.IsKeyDown(Keys.PageDown))
    players[currentPlayer].Power -= 20;
if (keybState.IsKeyDown(Keys.PageUp))
    players[currentPlayer].Power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

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

عندما يقوم المستخدم بضغط زر السهم الأعلى و الأسفل, سوف يتم زيادة طاقة المدفع بواحد.

بما أنه بهذه الطريقة سوف نحتاج إلى زمن طويل نوعا ما للوصول إلى أعلى طاقة, سيتم الإستجابة لضغط أزرار ال PageUP و PageDown بحيث إن تم الضغط على أحدهما تزداد طاقة المدفع بكمية أكبر.

الأسطر الأربعة الأخيرة ببساطة تقوم بتحديد قيمة طاقة المدفع في المجال من 0 إلى 1000.



إلى هنا كل شيء في هذا الدرس ☺ بإمكانك أن ترى أنه قد تم تدوير المدفع كإستجابة للضغط على لوحة المفاتيح ولكن من الصعب عليك أن تلاحظ زيادة قوة المدفع. لذلك سوف نحاول في الدرس التالي أن نقوم بعمل كتابة على الشاشة لطباعة زاوية المدفع و طاقته للاعب الحالي. إن شاء الله.

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;
        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");
            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();
        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);
        }
    }
}
}

```