الدرس الثاني عشر

أهلا بكم في الدرس الثاني عشر من سلسلة دروس تعلم ال Xna , سوف نقوم في هذا الدرس بإنشاء التضاريس بشكل عشوائي, "بلشنا الجد".

النتيجة من الدرس السابق غير مرضية أبدا. كتطوير أولي,سوف نقوم بتغيير إنحدار التضاريس. لن نشير إلى ميزات جديده في ال Xna, لكن سوف نعرض كيف بإمكاننا أن نستغل بعض الأرقام العشوائية في اللعبة.

بدلا من التضاريس المستوية, دعنا نقوم بتغيير الكود لكي يقوم بإنشاء إنحدارات التضاريس. قم بإستبدال الكود في الدالة GenerateTerrainContour:

```
 private void GenerateTerrainContour()
 {
     terrainContour = new int[screenWidth];

     float offset = screenHeight / 2;
     float peakheight = 100;
     float flatness = 50;
     for (int x = 0; x < screenWidth; x++)
     {
         double height = peakheight * Math.Sin((float)x /
flatness)+offset;
         terrainContour[x] = (int)height;
     }
 }
```

أولا قم بتشغيل الكود, يجب أن ترى أن التضاريس تشبه الصورة التالية. الإحداثي Y يتم حسابه بإستخدام إقتران الجيب "Sine", الذي يعتبر رياضيا مقابل للأمواج. هناك بعض المتغيرات التي يمكننا تغييرها:

- الإزاحة "Offset" هي اسهلها, لأنها ببساطه تحدد المسافة بين موقع منتصف الموجة على الشاشة.
- قيمة إرتفاع القمة "peakheight" يتم ضربها بناتج إقتران الجيب, مما يحدد إرتفاع الموجة.
- أخيرا, قيمة الإنسطاح "flatness" لها تأثيرا قوة (فاعلية) القيمة X , بحيث تسرع أو تبطئ من الموجة. هذا يزيد أو يقلل طول الموجة"wavelength" .

موجة بسيطة مثل هذه تبدو مزيفة بشكل كبير. يمكن حل ذلك عن طريق مزجها مع موجة أخرى. على سبيل المثال, دعنا نقول أننا نريد أن نجمع هذه الموجة مع موجة لها إرتفاع أصغر, و طول موجه أقل. سوف تكون النتيجة كالتالي:

في الكود الخاص بنا, سوف نقوم بإضافة موجة ثالثة. لإنتاج بشكل مختلف في كل مره, سوف نقوم بالتأثير على 3 عوامل لكل موجة بقيم عشوائية. هذا ما يجب أن تبدو عليه دالة GenerateTerrainContour :

```
 private void GenerateTerrainContour()
 {
     terrainContour = new int[screenWidth];

     double rand1 = randomizer.NextDouble() + 1;
     double rand2 = randomizer.NextDouble() + 2;
     double rand3 = randomizer.NextDouble() + 3;

     float offset = screenHeight / 2;
     float peakheight = 100;
     float flatness = 70;

     for (int x = 0; x < screenWidth; x++)
     {
         double height = peakheight / rand1 * Math.Sin((float)x /
flatness * rand1 + rand1);
         height += peakheight / rand2 * Math.Sin((float)x / flatness *
rand2 + rand2);
         height += peakheight / rand3 * Math.Sin((float)x / flatness *
rand3 + rand3);
         height += offset;
         terrainContour[x] = (int)height;
     }
 }
```

لكل موجة, قمنا أولا بإختيار قيمة عشوائية بين ال 0 و 1. بعدها قمنا بإزاحتها قليلا, بحيث أصبحت القيمة الأولى في المجال [1-0], و القيمة الثانية بين ال [2-1], و الأخيره بين ال [3-2]. للموجات الثلاثة , هذه القيم سوف يتم تقسيمها على ال "peakheight" وعلى ال "flatness", إذن الموجة الثالثة سوف تكون أقصر و أخفض من الموجة الأولى و الثانية. علاوة على ذلك, يجب أن تلاحظ أننا قمنا بإضافة القيم العشوائية داخل دالة الجيب. من ناحية أخرى كل الموجات الثلاثة يجب أن يبدءون بالإحداثي Y المحدد بال "offset", مما يجعل هذه النقطة ثابته في كل مره نبدأ فيها اللعبة.

الآن عندما تحاول تشغيل الكود, سوف تحصل على تضاريس مع إنحدارات أفضل, كما يظهر في الصورة في آخر الدرس.

هناك شيئ يمكنك أن تلاحظة مباشرة عند التنفيذ, هو أن اللاعبين غير موضوعين على التضاريس. هذا بالتأكيد لأننا قمنا بتحديد أماكن اللاعبين, بناء على الصورة الثابتة التي إستخدمناها في الدروس السابقة.

لحل هذه المشكلة , إبحث عن الأسطر الأربعة المسؤولين عن تحديد أماكن اللاعبين في الدالة SetUpPlayers , وقم **بحذفهم**. بدلا من ذلك , سوف نقوم بحساب المواقع في داخل التكرار في تلك الدالة , يجب أن يكون الكود على النحو التالي:

```
        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[i].Position = new Vector2();
            players[i].Position.X = screenWidth / (numberOfPlayers + 1) *
       (i + 1);
            players[i].Position.Y =
       terrainContour[(int)players[i].Position.X];
        }
```

الإحداثي X للاعبين يمكن إيجاده بسهولة عن طريق تقسيم عرض الشاشة على عدد اللاعبين. ال "1+" يلزم لكي نتأكد أنه لا يوجد أي لاعب على يمين أو يسار حدود الشاشة. الإحداثي Y يمكن إيجاده بسهولة أيضا من خلال معرفة إرتفاع التضاريس في الموقع الحالي (الإحداثي X الحالي).

على كل حال, لكي يعمل هذا الكود يجب أن تستطيع الدالة SetUpPlayers الوصول إلى مصفوفة ال terrainContour النهائية. هذا يعني أنه في داخل دالة LoadContent ,يجب علينا ان نستدعي الدالة SetUpPlayers بعد إستدعائنا للدالة GenerateTerrainContour :

```
GenerateTerrainContour();
SetUpPlayers();
CreateForeground();
```

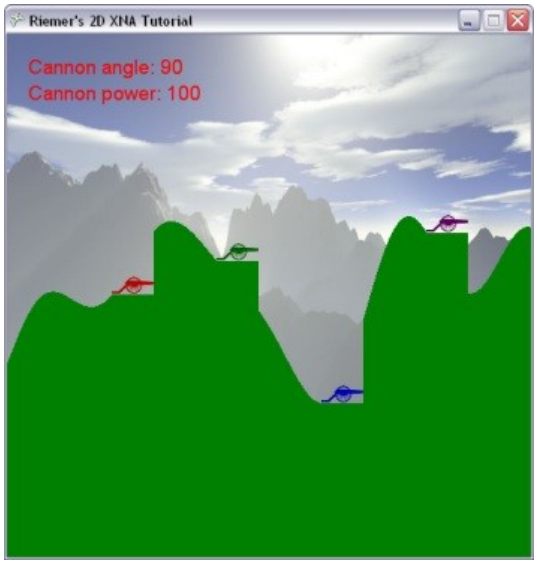
عندما تقوم بتشغيل الكود, سوف تلاحظ أن الركن الأسفل الأيسر للمدافع موجوده فعليا على الأرض, ولكن التضاريس تحتها غير متناسبة. لحل ذلك, يجب أن نقوم ببساطة بتوسيع التضاريس تحت اللاعبين, بإستخدام الدالة التالية:

```
private void FlattenTerrainBelowPlayers()
{
    foreach (PlayerData player in players)
        if (player.IsAlive)
            for (int x = 0; x < 40; x++)
                terrainContour[(int)player.Position.X + x] =
terrainContour[(int)player.Position.X];
}
```

لكل لاعب, الأرضية تحت المدفع تم تسويتها إلى نفس الإرتفاع الخاص بالركن السفلي الأيسر للمدفع. سوف نحتاج لإستدعاء هذه الدالة من داخل دالة الLoadContent , و لكن يجب أن نبقي شيئا في بالنا هو أن إستدعاء هذه الدالة يجب أن يتم بعد إستدعاء دالة ال GenerateTerrainContour و دالة ال SetUpPlayers , بما أنها تستخدم البيانات التي يتم تعريفها في تلك الدوال. من ناحية أخرى يجب أن يتم إستدعائها قبل إستدعاء الدالة CreateForeground , بما أنه هذه الدالة (الاخيره) تستخدم مخرجات الدالة TerrainContour , التي تم تعديلها بإستخدام دالة ال FlattenTerrainBelowPlayers :

```
GenerateTerrainContour();
SetUpPlayers();
FlattenTerrainBelowPlayers();
CreateForeground();
```

عند تشغيل الكود هذه المرة, يجب أن يكون إنحدار التضاريس ممتازا ☺



التمرين في هذا الدرس: هو محاولة التدرب على إستخدام دالة الجيب في رسم التضاريس.

في الدرس القادم سوف نجد حلا للون الأخضر, إن شاء الله.

كود المشروع للآن:

```
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();
```

```
        int[] terrainContour;

        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");
            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");
            playerScaling = 40.0f / (float)carriageTexture.Width;

             GenerateTerrainContour();
             SetUpPlayers();
             FlattenTerrainBelowPlayers();
             CreateForeground();
         }

         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[i].Position = new Vector2();
                 players[i].Position.X = screenWidth / (numberOfPlayers + 1) * (i + 1);
                 players[i].Position.Y = terrainContour[(int)players[i].Position.X];
             }
         }

         private void GenerateTerrainContour()
         {
             terrainContour = new int[screenWidth];

             double rand1 = randomizer.NextDouble() + 1;
             double rand2 = randomizer.NextDouble() + 2;
             double rand3 = randomizer.NextDouble() + 3;

             float offset = screenHeight / 2;
             float peakheight = 100;
             float flatness = 70;

             for (int x = 0; x < screenWidth; x++)
             {
                 double height = peakheight / rand1 * Math.Sin((float)x / flatness * rand1 + rand1);
                 height += peakheight / rand2 * Math.Sin((float)x / flatness * rand2 + rand2);
                 height += peakheight / rand3 * Math.Sin((float)x / flatness * rand3 + rand3);
                 height += offset;
                 terrainContour[x] = (int)height;
             }
         }

         private void FlattenTerrainBelowPlayers()
         {
             foreach (PlayerData player in players)
                 if (player.IsAlive)
```

```
                    for (int x = 0; x < 40; x++)
                        terrainContour[(int)player.Position.X + x] = terrainContour[(int)player.Position.X];
        }

        private void CreateForeground()
        {
            Color[] foregroundColors = new Color[screenWidth * screenHeight];

            for (int x = 0; x < screenWidth; x++)
            {
                for (int y = 0; y < screenHeight; y++)
                {
                    if (y > terrainContour[x])
                        foregroundColors[x + y * screenWidth] = Color.Green;
                    else
                        foregroundColors[x + y * screenWidth] = Color.TransparentBlack;
                }
            }

            foregroundTexture = new Texture2D(device, screenWidth, screenHeight, 1, TextureUsage.None,
SurfaceFormat.Color);
            foregroundTexture.SetData(foregroundColors);
        }

        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);
        }
    }
}
```