الدرس الحادي و العشرون

أهلا بكم في الدرس الحادي و العشرين من سلسلة دروس تعلم ال Xna, في هذا الدرس سوف نتحدث عن تشغيل المؤثرات الصوتية في ال Xna.

في هذه اللحظة، لدينا لعبة كاملة جاهزة للعب، تحتوي على كل الميزات و الإمكانيات التي مررنا عليها في هذه السلسلة من الدروس. ولكن بالإمكان تحسين الشعور العام في اللعبة بشكل **كبير** في حال تم إضافة مؤثرات صوتية للعبة، وهو ما سوف نفعله في هذا الدرس.

سوف نقوم بإضافة ثلاث مؤثرات صوتية بسيطة: واحد لإطلاق الصاروخ, واحد للإصطدام و الإنفجار بالأرضية و آخر للإنفجار المدفع. قم بإختيار أي ثلاث أصوات لديك حاليا. تأكد أنك قمت بوضعهم في مجلد ال Content الخاص بالمشروع، ولكن لا تقم بإستيرادهم إلى لعبة ال Xna بعد.

يتم التعامل مع الأصوات في ال Xna بطريقة مختلفة قليلا عن باقي المكونات، مثل الصور. يتم ذلك بإستخدام ال XAct، و هو برنامج صغير منفصل يأتي مع ال Xna. تم فصل هذا البرنامج عن باقي أجزاء تطوير اللعبة من أجل إتاحة الفرصه للفنانين الموسيقيين القيام بإضافة الكثير من المؤثرات الصوتية إلى اللعبة، بشكل منفصل عن فريق تطوير الكود في اللعبة.

من أجل فتح ال XAct، من قائمة إبدأ:

```
Start Menu=> Programs =>Microsoft Xna Game Studio 2.0 =>Tools=> Microsoft Cross-Platform Audio Creation Tool (XAct)
```

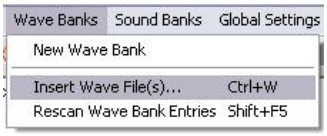
بعد أن يتم تشغيل ال XAct، قم ببدء مشروع جديد من خلال قائمة File و إختيار New Project. إنتقل إلى مجلد ال Content الخاص بمشروع ال Xna الخاص بك، أعطي المشروع إسم xactProject و إضغط على زر الحفظ.

بشكل عام، هناك ثلاث أشياء يلزمك معرفتها عن ال XAct :

- عندما تقوم بإستيراد ملف صوت إلى ال XAct، يسمى ذلك بالموجة "Wave".
- من كل موجة بإمكانك إنشاء عدة أصوات. بإمكانك إضافة مؤثرات خاصة إلى الأصوات، مثل مؤثر ال "Doppler". إذن بإمكانك إضافة مؤثرين مختلفين إلى موجة عن طريق إنشاء صوتين إثنين منها، و تطبيق كل مؤثر إلى كل صوت.
- بإمكانك إنشاء قائمة تشغيل "Play List" من الأصوات، و التي تدعى طابور "Cues" . عمليا سوف نبدأ في كود ال Xna من الطابور.

أولا سوف نحتاج لإنشاء بنك أمواج "Wave Bank" من أجل تخزين الملفات الصوتية الخاصة بنا. يمكن عمل ذلك من خلال الضغط بالزر الأيمن على خانة ال "Wave Banks" على يسار الشاشة، و بعدها إختيار "New Wave Bank":

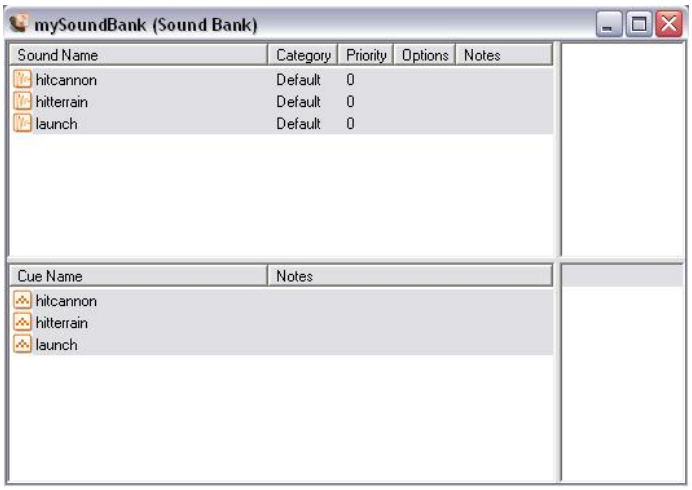
قم بتسمية البنك الجديد ب "myWaveBank" و إضغط عليها مرتين. الآن حان الوقت لإضافة الملفات الثلاث إلى بنك الأمواج، يمكن عمل ذلك من خلال إختيار قائمة ال "Wave Banks" و إختيار "Insert Wave Files":



تصفح إلى المكان الذي قمت بتنزيل الملفات الصوتية الثلاث فيه، قم بإختيار الملفات (إضغط مع الإستمرار على زر Ctrl من أجل إختيار عدة ملفات)، و بعدها إضغط على زر الفتح "Open". يجب أن ترى أنه قد تم إضافة الملفات الثلاثة في داخل بنك الأمواج باللون الأحمر، ما يعني أنه لم يتم ربطهم مع صوت بعد.

من أجل إنشاء صوت من الأمواج التي قمنا بإضافتها، سوف نحتاج أولا لإنشاء بنك أصوات "Sound Bank" عن طريق الضغط بالزر الايمن على ال "Sound Banks" و إختيار "New Sound Bank". قم بتسميته 'mySoundBank'. قم بتحريك الشاشات بحيث يكون بإستطاعتك مشاهدة بنك الأمواج و بنك الأصوات. الآن قم بإختيار الأمواج الثلاثة، وقم بسحبهم إلى داخل المنطقة العليا في بنك الأصوات. يجب أن ترى أنه تم إضافة الأصوات الثالثة إلى داخل بنك الأصوات، و أن الأمواج لم يعد لونهم أحمر.

في النهاية، قم بإختيار الأصوات الثلاثة، وقم بسحبهم إلى داخل الجزء الأسفل من بنك الأصوات، حيث تم تعريف الطوابير "Cues". يجب أن يكون لديك شاشة بنك الأصوات مماثلة للصورة التالية:



الآن كل شيئ جاهز تأكد أنك قمت بالحفظ من قائمة File-> Save. بإمكانك الآن إغلاق ال XAct إذا أردت.

في داخل مشروع ال Xna، يجب أن تقوم بإضافة ملف مشروع ال XAct مثل أي مكونات أخرى نقوم بإضافتها. بالزر الأيمن للماوس على مجلد ال Content, بعدها Add->Existing Item و بعدها إختر ملف ال xap. الذي قمنا بإنشاءه مؤخرا.

من أجل المؤثرات الصوتية الخاصة بنا في داخل كود ال Xna، سوف نحتاج لإضافة ثلاثة متغيرات في أعلى الكود:

```
AudioEngine audioEngine;
WaveBank waveBank;
SoundBank soundBank;
```

ال "AudioEngine" يجب أن يتم تحديثه في كل لحظة، و المتغيرين الآخرين يربطاننا مباشرة بالبنوك التي قمنا بإنشائها في مشروع ال XAct. قم بتعبئة هذه المتغيرات الثلاث في نهاية دالة "LoadContent":

```
 audioEngine = new AudioEngine("Content/xactProject.xgs");
 waveBank = new WaveBank(audioEngine, "Content/myWaveBank.xwb");
 soundBank = new SoundBank(audioEngine,
"Content/mySoundBank.xsb");
```

هذه الملفات الثلاثة تم إستخلاصها من ملف ال xap. بواسطة ال "Xna's Content Pipeline" بشكل تلقائي في كل مرة يتم فيها ترجمة "Compiling" المشروع.

دعنا نتأكد أننا نقوم بتحديث محرك الأصوات "AudioEngine" في نهاية الدالة Update:

```
 audioEngine.Update();
```

بعد الكثير من الإعداد، تشغيل الصوت بحد ذاته سهل جدا. الصوت الأول يجب أن يتم تشغيله عندما يتم إطلاق الصاروخ، و الذي يتم إكتشافه داخل الدالة ProcessKeyboard. قم بإضافة السطر التالي إلى نهاية تلك الدالة، في داخل الكتلة الشرطية التي تكتشف فيما إذا كان يجب إطلاق الصاروخ:

```
 soundBank.PlayCue("launch");
```

ببساطة يقوم هذا السطر بالطلب من بنك الأصوات بتشغيل الطابور المسمى "launch".

الصوتين الآخرين يجب أن يتم تشغيلهم في حالة تم إكتشاف تصادم بين الصاروخ و اللاعبين أو بين الصاروخ و الأرض. إذن إذهب إلى الدالة CheckCollisions، و قم بإضافة السطر التالي في داخل كتلة الشرط التي تفحص التصادم مع المدفع:

```
 soundBank.PlayCue("hitcannon");
```

وقم بإضافة السطر التالي في داخل كتلة الشرط التي تكتشف إصطدام الصاروخ بالارض:

```
 soundBank.PlayCue("hitterrain");
```

هذا كل شيئ! عندما تقوم بتشغيل الكود، يجب أن تسمع المؤثرات الصوتية في كل مرة تقوم فيها بإطلاق صاروخ، و عندما يصطدم الصاروخ بأي شيئ.

الكود 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 ParticleData
```

```
{
    public float BirthTime;
    public float MaxAge;
    public Vector2 OrginalPosition;
    public Vector2 Accelaration;
    public Vector2 Direction;
    public Vector2 Position;
    public float Scaling;
    public Color ModColor;
}

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;
    Texture2D groundTexture;
    Texture2D explosionTexture;
    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<ParticleData> particleList = new List<ParticleData> ();
   List<Vector2> smokeList = new List<Vector2> ();        Random randomizer = new Random();
   int[] terrainContour;

   Color[,] rocketColorArray;
   Color[,] foregroundColorArray;
   Color[,] carriageColorArray;
   Color[,] cannonColorArray;
   Color[,] explosionColorArray;


    AudioEngine audioEngine;
    WaveBank waveBank;
    SoundBank soundBank;

    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");
            groundTexture = Content.Load<Texture2D> ("ground");
            font = Content.Load<SpriteFont> ("myFont");
            explosionTexture = Content.Load<Texture2D> ("explosion");
            playerScaling = 40.0f / (float)carriageTexture.Width;
            GenerateTerrainContour();
            SetUpPlayers();
            FlattenTerrainBelowPlayers();
            CreateForeground();

            rocketColorArray = TextureTo2DArray(rocketTexture);
            carriageColorArray = TextureTo2DArray(carriageTexture);
            cannonColorArray = TextureTo2DArray(cannonTexture);
            explosionColorArray = TextureTo2DArray(explosionTexture);


             audioEngine = new AudioEngine("Content/xactProject.xgs");
             waveBank = new WaveBank(audioEngine, "Content/myWaveBank.xwb");
             soundBank = new SoundBank(audioEngine, "Content/mySoundBank.xsb");
        }

        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[,] groundColors = TextureTo2DArray(groundTexture);
            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] = groundColors[x % groundTexture.Width, y %
groundTexture.Height];
                    else
                        foregroundColors[x + y * screenWidth] = Color.TransparentBlack;
                }
            }
```

```
            foregroundTexture = new Texture2D(device, screenWidth, screenHeight, 1, TextureUsage.None,
SurfaceFormat.Color);
            foregroundTexture.SetData(foregroundColors);
            foregroundColorArray = TextureTo2DArray(foregroundTexture);
        }

        private Color[,] TextureTo2DArray(Texture2D texture)
        {
            Color[] colors1D = new Color[texture.Width * texture.Height];
            texture.GetData(colors1D);

            Color[,] colors2D = new Color[texture.Width, texture.Height];
            for (int x = 0; x < texture.Width; x++)
                for (int y = 0; y < texture.Height; y++)
                    colors2D[x, y] = colors1D[x + y * texture.Width];

            return colors2D;
        }

        protected override void UnloadContent()
        {
        }

        protected override void Update(GameTime gameTime)
        {
            if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
                this.Exit();

            if ((!rocketFlying) && (particleList.Count == 0))
                ProcessKeyboard();

            if (rocketFlying)
            {
                UpdateRocket();
                CheckCollisions(gameTime);
            }

            if (particleList.Count > 0)
                UpdateParticles(gameTime);

            audioEngine.Update();

            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;
    soundBank.PlayCue("hitcannon");
            }
        }

        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);
                }
            }
        }

        private Vector2 TexturesCollide(Color[,] tex1, Matrix mat1, Color[,] tex2, Matrix mat2)
        {
            Matrix mat1to2 = mat1 * Matrix.Invert(mat2);

            int width1 = tex1.GetLength(0);
            int height1 = tex1.GetLength(1);
            int width2 = tex2.GetLength(0);
            int height2 = tex2.GetLength(1);

            for (int x1 = 0; x1 < width1; x1++)
            {
                for (int y1 = 0; y1 < height1; y1++)
                {
                    Vector2 pos1 = new Vector2(x1, y1);
                    Vector2 pos2 = Vector2.Transform(pos1, mat1to2);

                    int x2 = (int)pos2.X;
                    int y2 = (int)pos2.Y;
                    if ((x2 >= 0) && (x2 < width2))
                    {
                        if ((y2 >= 0) && (y2 < height2))
                        {
                            if (tex1[x1, y1].A > 0)
                            {
                                if (tex2[x2, y2].A > 0)
                                {
                                    Vector2 screenPos = Vector2.Transform(pos1, mat1);
                                    return screenPos;
                                }
                            }
                        }
                    }
                }
            }

            return new Vector2(-1, -1);
        }

        private Vector2 CheckTerrainCollision()
        {
            Matrix rocketMat = Matrix.CreateTranslation(-42, -240, 0) * Matrix.CreateRotationZ(rocketAngle) *
Matrix.CreateScale(rocketScaling) * Matrix.CreateTranslation(rocketPosition.X, rocketPosition.Y, 0);
            Matrix terrainMat = Matrix.Identity;
            Vector2 terrainCollisionPoint = TexturesCollide(rocketColorArray, rocketMat, foregroundColorArray,
terrainMat);
            return terrainCollisionPoint;
        }

        private Vector2 CheckPlayersCollision()
        {
            Matrix rocketMat = Matrix.CreateTranslation(-42, -240, 0) * Matrix.CreateRotationZ(rocketAngle) *
Matrix.CreateScale(rocketScaling) * Matrix.CreateTranslation(rocketPosition.X, rocketPosition.Y, 0);
            for (int i = 0; i < numberOfPlayers; i++)
            {
                PlayerData player = players[i];
                if (player.IsAlive)
                {
                    if (i != currentPlayer)
                    {
                        int xPos = (int)player.Position.X;
                        int yPos = (int)player.Position.Y;

                        Matrix carriageMat = Matrix.CreateTranslation(0, -carriageTexture.Height, 0) *
Matrix.CreateScale(playerScaling) * Matrix.CreateTranslation(xPos, yPos, 0);
                        Vector2 carriageCollisionPoint = TexturesCollide(carriageColorArray, carriageMat,
rocketColorArray, rocketMat);
                        if (carriageCollisionPoint.X > -1)
                        {
                            players[i].IsAlive = false;
                            return carriageCollisionPoint;
                        }

                        Matrix cannonMat = Matrix.CreateTranslation(-11, -50, 0) *
Matrix.CreateRotationZ(player.Angle) * Matrix.CreateScale(playerScaling) * Matrix.CreateTranslation(xPos + 20, yPos -
10, 0);
                        Vector2 cannonCollisionPoint = TexturesCollide(cannonColorArray, cannonMat, rocketColorArray,
rocketMat);
                        if (cannonCollisionPoint.X > -1)
                        {
                            players[i].IsAlive = false;
                            return cannonCollisionPoint;
                        }
                    }
```

```
                }
            }
            return new Vector2(-1, -1);
        }

        private bool CheckOutOfScreen()
        {
            bool rocketOutOfScreen = rocketPosition.Y > screenHeight;
            rocketOutOfScreen |= rocketPosition.X < 0;
            rocketOutOfScreen |= rocketPosition.X > screenWidth;

            return rocketOutOfScreen;
        }

        private void CheckCollisions(GameTime gameTime)
        {
            Vector2 terrainCollisionPoint = CheckTerrainCollision();
            Vector2 playerCollisionPoint = CheckPlayersCollision();
            bool rocketOutOfScreen = CheckOutOfScreen();

            if (playerCollisionPoint.X > -1)
            {
                rocketFlying = false;

               smokeList = new List<Vector2> ();               NextPlayer();
               AddExplosion(playerCollisionPoint, 10, 80.0f, 2000.0f, gameTime);

                soundBank.PlayCue("hitcannon");
            }

            if (terrainCollisionPoint.X > -1)
            {
                rocketFlying = false;

               smokeList = new List<Vector2> ();               NextPlayer();
               AddExplosion(terrainCollisionPoint, 4, 30.0f, 1000.0f, gameTime);

                soundBank.PlayCue("hitterrain");
            }

            if (rocketOutOfScreen)
            {
                rocketFlying = false;

               smokeList = new List<Vector2> ();               NextPlayer();
           }
        }

        private void NextPlayer()
        {
            currentPlayer = currentPlayer + 1;
            currentPlayer = currentPlayer % numberOfPlayers;
            while (!players[currentPlayer].IsAlive)
            {
                currentPlayer = ++currentPlayer % numberOfPlayers;
            }
        }

        private void AddExplosion(Vector2 explosionPos, int numberOfParticles, float size, float maxAge, GameTime
gameTime)
        {
            for (int i = 0; i < numberOfParticles; i++)
                AddExplosionParticle(explosionPos, size, maxAge, gameTime);

            float rotation = (float)randomizer.Next(10);
            Matrix mat = Matrix.CreateTranslation(-explosionTexture.Width / 2, -explosionTexture.Height / 2, 0) *
Matrix.CreateRotationZ(rotation) * Matrix.CreateScale(size / (float)explosionTexture.Width * 2.0f) *
Matrix.CreateTranslation(explosionPos.X, explosionPos.Y, 0);
            AddCrater(explosionColorArray, mat);

            for (int i = 0; i < players.Length; i++)
                players[i].Position.Y = terrainContour[(int)players[i].Position.X];
            FlattenTerrainBelowPlayers();
            CreateForeground();
        }

        private void AddExplosionParticle(Vector2 explosionPos, float explosionSize, float maxAge, GameTime gameTime)
        {
            ParticleData particle = new ParticleData();

            particle.OrginalPosition = explosionPos;
            particle.Position = particle.OrginalPosition;

            particle.BirthTime = (float)gameTime.TotalGameTime.TotalMilliseconds;
            particle.MaxAge = maxAge;
            particle.Scaling = 0.25f;
            particle.ModColor = Color.White;

            float particleDistance = (float)randomizer.NextDouble() * explosionSize;
            Vector2 displacement = new Vector2(particleDistance, 0);
            float angle = MathHelper.ToRadians(randomizer.Next(360));
            displacement = Vector2.Transform(displacement, Matrix.CreateRotationZ(angle));

            particle.Direction = displacement * 2.0f;
            particle.Accelaration = -particle.Direction;
```

```
            particleList.Add(particle);
        }

        private void UpdateParticles(GameTime gameTime)
        {
            float now = (float)gameTime.TotalGameTime.TotalMilliseconds;
            for (int i = particleList.Count - 1; i >= 0; i--)
            {
                ParticleData particle = particleList[i];
                float timeAlive = now - particle.BirthTime;

                if (timeAlive > particle.MaxAge)
                {
                    particleList.RemoveAt(i);
                }
                else
                {
                    float relAge = timeAlive / particle.MaxAge;
                    particle.Position = 0.5f * particle.Accelaration * relAge * relAge + particle.Direction * relAge +
particle.OrginalPosition;

                    float invAge = 1.0f - relAge;
                    particle.ModColor = new Color(new Vector4(invAge, invAge, invAge, invAge));

                    Vector2 positionFromCenter = particle.Position - particle.OrginalPosition;
                    float distance = positionFromCenter.Length();
                    particle.Scaling = (50.0f + distance) / 200.0f;

                    particleList[i] = particle;
                }
            }
        }

        private void AddCrater(Color[,] tex, Matrix mat)
        {
            int width = tex.GetLength(0);
            int height = tex.GetLength(1);

            for (int x = 0; x < width; x++)
            {
                for (int y = 0; y < height; y++)
                {
                    if (tex[x, y].R > 10)
                    {
                        Vector2 imagePos = new Vector2(x, y);
                        Vector2 screenPos = Vector2.Transform(imagePos, mat);

                        int screenX = (int)screenPos.X;
                        int screenY = (int)screenPos.Y;

                        if ((screenX) > 0 && (screenX < screenWidth))
                            if (terrainContour[screenX] < screenY)
                                terrainContour[screenX] = screenY;
                    }
                }
            }
        }

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

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

            spriteBatch.Begin(SpriteBlendMode.Additive, SpriteSortMode.Deferred, SaveStateMode.None);
            DrawExplosion();
            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), rocketScaling, 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);
        }

        private void DrawExplosion()
        {
            for (int i = 0; i < particleList.Count; i++)
            {
                ParticleData particle = particleList[i];
                spriteBatch.Draw(explosionTexture, particle.Position, null, particle.ModColor, i, new Vector2(256,
256), particle.Scaling, SpriteEffects.None, 1);
            }
        }
    }
}
```