EBOOK

Michael Andre Franiatte

Game Creation and Coding with Monogame Game Engine

Methods for Learning to Create a Game with Game Engines

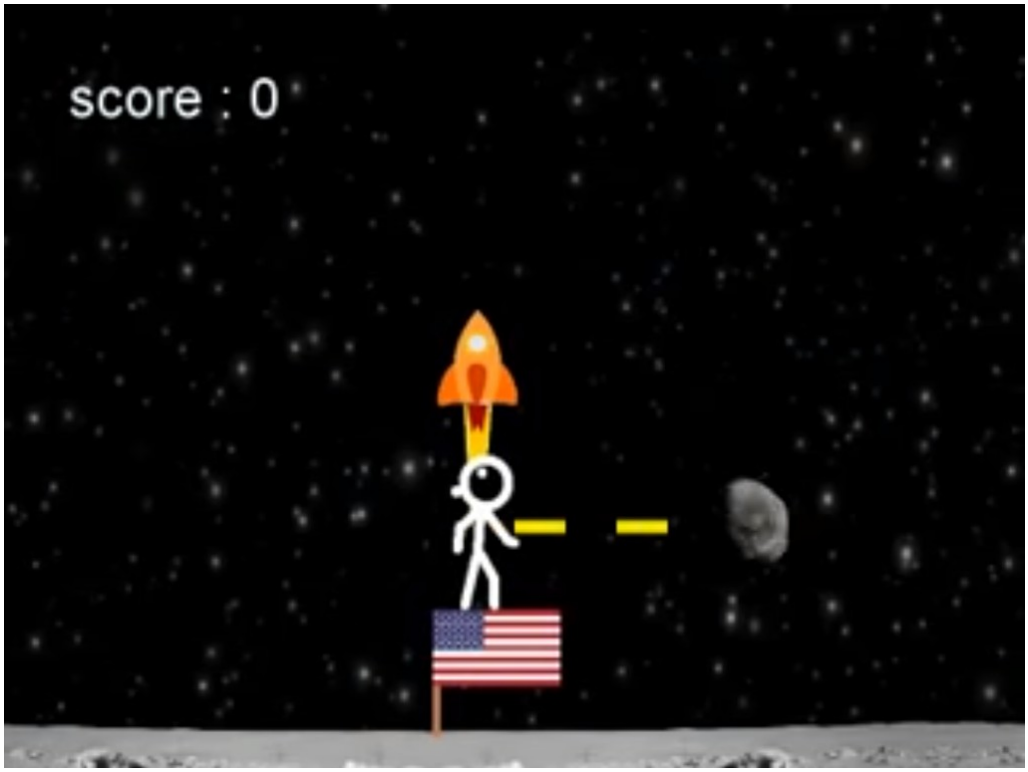
EBOOK

Game Creation and Coding with Monogame Game Engine

Methods for Learning to Create a Game with Game Engines

Michael Franiatte

06/10/2018



Game Creation and Coding with Monogame Game Engine: Methods for Learning to Create a Game with Game Engines

Michael Franiatte*

Abstract

With Monogame game engine, it's possible to create a simple game adapted to available functions and variables of XNA frameworks, but limited, despite access to all frameworks of Visual Studio. Coding with C# is a good way to understand basis of programming, and it's more enjoyable than a native language like with C++. This book concern basis for coding keyboard hook and remapping, mouse position hook, adding and changing resources to your project, adding and using classes for levels, finding and changing sprites, spritefont, audio files, finding tools to change it. A little game has been creating to illustrate this book and the codes are available in. The codes explain to create a 2D shoot'em up game, a 2D side scrolling game, and a 2D endless runner game. Training to create a game with Monogame is a good first step for understand how to create a game with other game engines, more complete and easy to use, like GameMaker or Construct. These both game engines incorporate sprite editor and in hand turnkey functions and variables. This book contains all information to know for understand a first game creation, without the need of searching by yourself among the internet forest. It's a good way for starting by burning the steps. It gives you methods to better understand the game creation.

Keywords: *Monogame, game engine, XNA, C#, coding, tools, functions, variables, 2D.*

* Author correspondence: michael.franiatte@gmail.com

1. Adding and Changing Resources

For creating an endless background, you can use powerpoint and paint. In powerpoint, add twice the picture for the background and fill with both pictures the powerpoint sheet. Reverse one of your pictures from one side of it, and associate the two pictures together for paste their sides together. Toggle to full screen the sheet and take a screenshot with the key on your keyboard, for copy it in paint. The pictures can be edited with paint.net or on http://online-image-editor.com for create transparency. Check https://www141.lunapic.com for nice online picture effects and arts. The Gimp picture editor is another good program for editing picture like a professional found at https://www.gimp.org/downloads/.

You can create an icon for your game on http://convertir-une-image.com from one of your own picture which much be sized to square. Overwrite the default Monogame Icon with your icon or picture from paint editor or by simply replacing the Monogame icon file having the same name as your game. For add icon to your game, you can use this code also as following:

in program.cs,

```
static void Main(string[] args)
{
    using (Game1 game = new Game1())
    {
((System.Windows.Forms.Form)System.Windows.Forms.Form.FromHandle(game.Window.Handle)).Icon
= new System.Drawing.Icon("file.ico");
                game.Run();
    }
}
```

The file.ico should be copied to the same place as your executable. Add references under the solution like System.Windows.Forms, System.Drawing or library or DLL you need for using available functions into your project. You have to reference these assemblies in your solution:

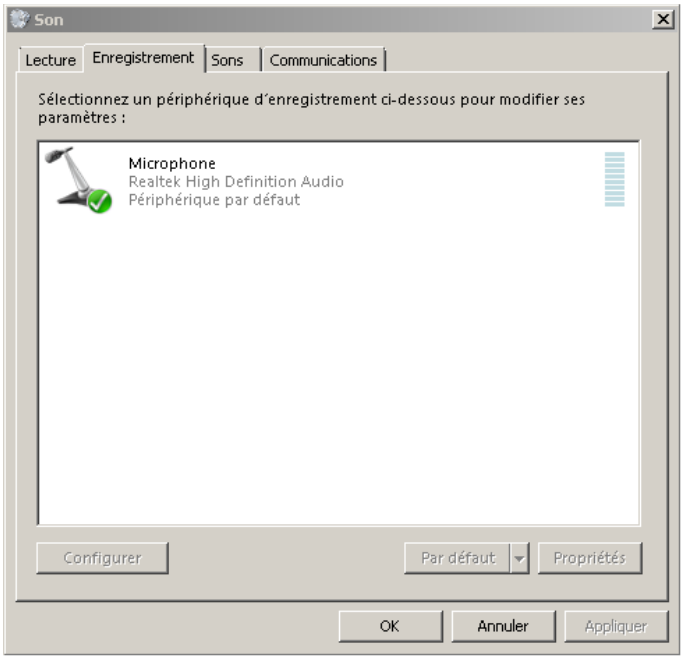
System.Windows.Forms

System.Drawing

You can also replace the Game.ico in your solution folder, which has same effect.

For add assets, copy files under content folder, and then use the tab ressources, under properties of the project, it will be added to the content folder of your project. Change properties of all assets under the content folder in your project for "Copy to Output Directory" to "Copy always". After adding audio files to the content folder, build solution, under the output folder there is the sound xnb file to add in the Content folder. For audio files (mp3) and spritefont (for draw string) use Monogame pipeline manager. To change size and police of string open the spritefont under the content folder in Visual Studio. For import audio files, audio Windows services must run.

Use free program Audacity available on https://www.audacityteam.org/download/windows/ to cut sounds of audio files. Sounds can be found on youtube by recording it with Microsoft tool named SoundRecoder. For using it, you must set properties of your sound card as following:



There are free music files free of author rights to add in your games here:

http://freemusicarchive.org/genre/Classical/

You can find good pictures for inspiration and use in your games here:

http://getwallpapers.com/collection/space-stars-background

There are nice Mongame tutorials all over the internet like:

https://www.youtube.com/watch?v=aVrZEG-GGTQ

http://www.gamefromscratch.com/post/2015/06/19/MonoGame-Tutorial-Textures-and-SpriteBatch.aspx

http://rbwhitaker.wikidot.com/monogame-managing-content

You can download Visual Studio, Monogame, Paint.Net here:

https://www.visualstudio.com/fr/vs/older-downloads/

http://www.monogame.net/

https://www.getpaint.net/download.html#download

Microsoft and Steam publisher access can be found here:

https://developer.microsoft.com/en-us/store/publish-apps

https://partner.steamgames.com/

2. Adding and Using Classes for Levels

The threads of load contents, update and draw can be organized in classes for each of your levels you will have to write the codes. Here you can find two classes code examples for organize your codes in Monogame as following:

```
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;

namespace MonoGameClassTest
{
    /// <summary>
    /// This is the main type for your game.
    /// </summary>
    public class Game1 : Microsoft.Xna.Framework.Game
    {
        Level level;
        GraphicsDeviceManager graphics;
        public Game1()
```

```
        {
            graphics = new GraphicsDeviceManager(this);
            this.TargetElapsedTime = System.TimeSpan.FromSeconds(1.0f / 1000.0f);
        }
        protected override void Initialize()
        {
            // TODO: Add your initialization logic here
            level = new Level();
            base.Initialize();
        }
        protected override void Update(GameTime gameTime)
        {
            // TODO: Add your update logic here

            level.Update(gameTime);
            base.Update(gameTime);
        }
    }
}
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;

namespace MonoGameClassTest
{
    /// <summary>
    /// This is the main type for your game.
    /// </summary>
    public class Level
    {
        int i = 0;
        float elapsed;
        public void Update(GameTime gameTime)
        {
            i = i + 1;
            // TODO: Add your update logic here
            elapsed = (float)gameTime.ElapsedGameTime.TotalMilliseconds;
        }
    }
}
```

3. Hook of Keyboard and Mouse

There are different possibilities for hook keyboard and mouse. With XNA frame-works, there are gamepad, keyboard and mouse methods to retrieve user inputs. It runs under a Monogame project but not under a Form or Console project. There is a lot of information on how to hook HID (Human Interface Device) by the author of this book in his books untitled "… Gamepad Libraries to Play PC Games" and all over internet. Here you can find a code example for hook mouse position when the user moves his mouse as following:

```
using System;
using System.IO;
using System.IO.IsolatedStorage;
```

```
using System.Xml.Serialization;
using System.Collections.Generic;
using System.Diagnostics;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Media;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
namespace CursorTest
{
    /// <summary>
    /// This is the main type for your game.
    /// </summary>
    public class Game1 : Game
    {
        GraphicsDeviceManager graphics;
        SpriteBatch spriteBatch;
        public const int WindowsWidth = 800;
        public const int WindowsHeight = 600;
        Texture2D LayoutBackGroundLvl2;
        Texture2D Spaceship;
        public double LayoutBackGroundLvl1PositionCount;
        public Vector2 SpaceshipPosition;
        public Game1()
        {
            graphics = new GraphicsDeviceManager(this);
            Content.RootDirectory = "Content";
            graphics.PreferredBackBufferWidth = 800;
            graphics.PreferredBackBufferHeight = 600;
            graphics.ToggleFullScreen();
        }

        /// <summary>
        /// Allows the game to perform any initialization it needs to before starting to
run.
        /// This is where it can query for any required services and load any non-graphic
        /// related content.  Calling base.Initialize will enumerate through any components
        /// and initialize them as well.
        /// </summary>
        protected override void Initialize()
        {
            // TODO: Add your initialization logic here

            base.Initialize();
        }

        /// <summary>
        /// LoadContent will be called once per game and is the place to load
        /// all of your content.
        /// </summary>
        protected override void LoadContent()
        {
            // Create a new SpriteBatch, which can be used to draw textures.
            spriteBatch = new SpriteBatch(GraphicsDevice);

            FileStream filestreamLayoutBackGroundLvl2 = new
FileStream("Content/LayoutBackGroundLvl2.png", FileMode.Open);
            LayoutBackGroundLvl2 = Texture2D.FromStream(GraphicsDevice,
filestreamLayoutBackGroundLvl2);
            filestreamLayoutBackGroundLvl2.Dispose();
            FileStream filestreamSpaceship = new FileStream("Content/Spaceship.png",
FileMode.Open);
            Spaceship = Texture2D.FromStream(GraphicsDevice, filestreamSpaceship);
            filestreamSpaceship.Dispose();
            // TODO: use this.Content to load your game content here
        }
```

```
        /// <summary>
        /// UnloadContent will be called once per game and is the place to unload
        /// game-specific content.
        /// </summary>
        protected override void UnloadContent()
        {
            // TODO: Unload any non ContentManager content here
        }

        /// <summary>
        /// Allows the game to run logic such as updating the world,
        /// checking for collisions, gathering input, and playing audio.
        /// </summary>
        /// <param name="gameTime">Provides a snapshot of timing values.</param>
        protected override void Update(GameTime gameTime)
        {
            if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed ||
Keyboard.GetState().IsKeyDown(Keys.Escape))
                Exit();
            SpaceshipPosition.X = System.Windows.Forms.Cursor.Position.X;
            SpaceshipPosition.Y = System.Windows.Forms.Cursor.Position.Y;
            // TODO: Add your update logic here

            base.Update(gameTime);
        }

        /// <summary>
        /// This is called when the game should draw itself.
        /// </summary>
        /// <param name="gameTime">Provides a snapshot of timing values.</param>
        protected override void Draw(GameTime gameTime)
        {
            GraphicsDevice.Clear(Color.CornflowerBlue);
            // TODO: Add your drawing code here
            LayoutBackGroundLvl1PositionCount = LayoutBackGroundLvl1PositionCount + 1;
            spriteBatch.Begin();
            spriteBatch.Draw(LayoutBackGroundLvl2, new Rectangle(0,
(int)LayoutBackGroundLvl1PositionCount, WindowsWidth, WindowsHeight), Color.White);
            spriteBatch.Draw(LayoutBackGroundLvl2, new Rectangle(0,
(int)LayoutBackGroundLvl1PositionCount - WindowsHeight, WindowsWidth, WindowsHeight), Col-
or.White);
            spriteBatch.Draw(Spaceship, new Rectangle((int)SpaceshipPosition.X,
(int)SpaceshipPosition.Y, 70, 180), Color.White);
            spriteBatch.End();
            if (LayoutBackGroundLvl1PositionCount >= 600)
                LayoutBackGroundLvl1PositionCount = 0;
            base.Draw(gameTime);
        }
    }
}
```

4. Using Linear Flow of Codes and Values of Variables

It's important to understand the state of variables and how to use it to reproduce what you imagine for your game. Here an example showing how to trigger an event for make jumping natural your character with variables and step of state in code lines as following:

```
gravity -= 0.2;
jump = buttonstate;
if (jump & acceleration > 0)
{
  sauter += acceleration + gravity;
  jump = true;
}
else
{
  acceleration = 1;
  jump = false;
}
acceleration -= 0.1;
if (acceleration <= 0)
  acceleration = 0;
```

5. Code Examples from CrazySpaceship

I created a first game for training myself with Monogame. Game I've created is illustrated by this video here:

https://www.youtube.com/watch?v=G5pCV9KDCgU

There is a level for forward player to remapping, 2D shoot'em up and 2D side scrolling runner. There are an enemy which follows the player spaceship, and an asteroid to shoot, but if the user doesn't shoot the asteroid it's game over. The end is reached when the player go to his spaceship. Here you can find the code example of this game as following:

```
using System;
using System.IO;
using System.IO.IsolatedStorage;
using System.Xml.Serialization;
using System.Collections.Generic;
using System.Diagnostics;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Media;
```

```
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Storage;
using Microsoft.Xna.Framework.GamerServices;
namespace CrazySpaceship
{
    /// <summary>
    /// This is the main type for your game.
    /// </summary>
    public class Game1 : Game
    {
        private bool[] WD = new bool[65];
        private int[] WU = new int[65];
        private bool[] _Valuechanged = new bool[65];
        private bool[] _valuechanged = new bool[65];
        private bool[] mWScbool = new bool[65];
        private bool this[int i]
        {
            get { return _valuechanged[i]; }
            set
            {
                if (_valuechanged[i] != value)
                    _Valuechanged[i] = true;
                else
                    _Valuechanged[i] = false;
                _valuechanged[i] = value;
            }
        }
        private void cevent(int cint, bool cbool)
        {
            mWScbool[cint] = cbool;
            this[cint] = mWScbool[cint];
            if (_Valuechanged[cint] & mWScbool[cint])
                WD[cint] = true;
            else
                WD[cint] = false;
            if (mWScbool[cint] & !(WU[cint] >= 1))
                WU[cint] = 1;
            if (WU[cint] > 0)
                WU[cint] = WU[cint] + 1;
        }
        public static string type = "ARROWS";
        private static void assign()
        {
            if (Keyboard.GetState().IsKeyDown(Keys.Up) | Key-
board.GetState().IsKeyDown(Keys.Down) | Keyboard.GetState().IsKeyDown(Keys.Left) | Key-
board.GetState().IsKeyDown(Keys.Right))
                type = "ARROWS";
            if (Keyboard.GetState().IsKeyDown(Keys.W) | Key-
board.GetState().IsKeyDown(Keys.A) | Keyboard.GetState().IsKeyDown(Keys.S) | Key-
board.GetState().IsKeyDown(Keys.D))
                type = "WASD";
            if (Keyboard.GetState().IsKeyDown(Keys.Z) | Key-
board.GetState().IsKeyDown(Keys.Q) | Keyboard.GetState().IsKeyDown(Keys.S) | Key-
board.GetState().IsKeyDown(Keys.D))
                type = "ZQSD";
            switch (type)
            {
                case "ARROWS":
                    KeyUp = Keys.Up;
                    KeyDown = Keys.Down;
                    KeyLeft = Keys.Left;
                    KeyRight = Keys.Right;
                    break;
                case "WASD":
                    KeyUp = Keys.W;
```

```
                KeyLeft = Keys.A;
                KeyDown = Keys.S;
                KeyRight = Keys.D;
                break;
            case "ZQSD":
                KeyUp = Keys.Z;
                KeyLeft = Keys.Q;
                KeyDown = Keys.S;
                KeyRight = Keys.D;
                break;
        }
        Typechange = type;
        if (_Typechange)
            WaitForOptions = false;
        if (type == "")
            type = "ARROWS";
    }
    public static string _typechange;
    public static bool _Typechange;
    public static string Typechange
    {
        get { return _typechange; }
        set
        {
            if (_typechange != value)
                _Typechange = true;
            else
                _Typechange = false;
            _typechange = value;
        }
    }
    public static Keys KeyUp;
    public static Keys KeyDown;
    public static Keys KeyLeft;
    public static Keys KeyRight;
    GraphicsDeviceManager graphics;
    SpriteBatch spriteBatch;
    Texture2D LayoutMenu;
    Texture2D PlayButtonMenu;
    Texture2D ContinueButtonMenu;
    Texture2D OptionsButtonMenu;
    Texture2D QuitButtonMenu;
    Texture2D LayoutBackGroundIntro;
    Texture2D LayoutBackGroundLvl1;
    Texture2D LayoutBackGroundLvl2;
    Texture2D Spaceship;
    Texture2D astronaute1;
    Texture2D BulletSpaceship;
    Texture2D End;
    Texture2D lunarsoil;
    Texture2D stickofflag;
    Texture2D americanflag;
    Texture2D Asteroid;
    Texture2D Gameover;
    Texture2D BacgroundOptions;
    Texture2D TextesOptions;
    Texture2D ARROWSOptions;
    Texture2D WASDOptions;
    Texture2D ZQSDOptions;
    Texture2D enemy1;
    public const int WindowsWidth = 800;
    public const int WindowsHeight = 600;
    public int ChainCount;
    public int ChainCountSaved;
    public int ChainCountStock;
    public int LvlEndCount;
```

```
        public int BrightMenuCount;
        public int GameoverCount;
        public int AsteroidDestroyCount;
        public int WaitForOptionsCount;
        public int rndmoveenemy1count;
        public double BrightMenuCoef;
        public double LayoutBackGroundLvl1PositionCount;
        public Vector2 SpaceshipPosition;
        public Vector2 astronaute1Position;
        public Vector2 astronaute1TruePosition;
        public Vector2 BulletsOriginPosition;
        public Vector2 BulletsInitialPosition;
        public Vector2 BulletsPosition;
        public Vector2 Bullets2ndOriginPosition;
        public Vector2 Bullets2ndInitialPosition;
        public Vector2 Bullets2ndPosition;
        public Vector2 textesMenuPosition;
        public Vector2 SpaceshipCenterPosition;
        public Vector2 AsteroidPosition;
        public Vector2 enemy1RandomPosition;
        public Vector2 enemy1Position;
        public Vector2 enemy1OriginPosition;
        public Vector2 enemy1SearchSpaceshipPosition;
        public Vector2 rndmoveenemy1;
        Random rnd = new Random();
        public float AsteroidRotation;
        public bool GameOverTrue;
        public bool AsteroidNotAlive;
        public bool enemy1NotAlive;
        public static bool WaitForOptions;
        Rectangle AsteroidHitBox;
        Rectangle BulletsHitBox;
        Rectangle Bullets2ndHitBox;
        Rectangle astronaute1HitBox;
        Rectangle SpaceshipHitBox;
        Rectangle americanflagHitBox;
        Rectangle astronaute1OnFlagHitBox;
        Rectangle enemy1HitBox;
        SpriteFont AsteroidFont;
        SoundEffectInstance sound;
        public Game1()
        {
            graphics = new GraphicsDeviceManager(this);
            Content.RootDirectory = "Content";
            graphics.PreferredBackBufferWidth = 800;
            graphics.PreferredBackBufferHeight = 600;
            graphics.ToggleFullScreen();
        }

        /// <summary>
        /// Allows the game to perform any initialization it needs to before starting to
run.
        /// This is where it can query for any required services and load any non-graphic
        /// related content.  Calling base.Initialize will enumerate through any components
        /// and initialize them as well.
        /// </summary>
        protected override void Initialize()
        {
            // TODO: Add your initialization logic here
            SpaceshipPosition = new Vector2(400 - 160 / 2, 450);
            BulletsOriginPosition = new Vector2(400 - 160 / 2 + 70 / 2, 450);
            astronaute1Position = new Vector2(400 - 160 / 2, 450);
            BrightMenuCoef = 1.05;
            AsteroidRotation = (float)0.2;
            //rndmoveenemy1.X = rnd.Next(-1, 2);
            //rndmoveenemy1.Y = rnd.Next(-1, 2);
```

```
            try
            {
                using (var data = new System.IO.StreamReader("test.dat"))
                {
                    data.ReadLine();
                    type = data.ReadLine();
                    data.Close();
                }
            }
            catch { }
            assign();
            base.Initialize();
        }

        /// <summary>
        /// LoadContent will be called once per game and is the place to load
        /// all of your content.
        /// </summary>
        protected override void LoadContent()
        {
            // Create a new SpriteBatch, which can be used to draw textures.
            spriteBatch = new SpriteBatch(GraphicsDevice);
            FileStream filestreamLayoutMenu = new FileStream("Content/LayoutMenu.png",
FileMode.Open);
            LayoutMenu = Texture2D.FromStream(GraphicsDevice, filestreamLayoutMenu);
            filestreamLayoutMenu.Dispose();
            FileStream filestreamContinueButtonMenu = new
FileStream("Content/ContinueButtonMenu.png", FileMode.Open);
            ContinueButtonMenu = Texture2D.FromStream(GraphicsDevice,
filestreamContinueButtonMenu);
            filestreamContinueButtonMenu.Dispose();
            FileStream filestreamPlayButtonMenu = new
FileStream("Content/PlayButtonMenu.png", FileMode.Open);
            PlayButtonMenu = Texture2D.FromStream(GraphicsDevice,
filestreamPlayButtonMenu);
            filestreamPlayButtonMenu.Dispose();
            FileStream filestreamOptionsButtonMenu = new
FileStream("Content/OptionsButtonMenu.png", FileMode.Open);
            OptionsButtonMenu = Texture2D.FromStream(GraphicsDevice,
filestreamOptionsButtonMenu);
            filestreamOptionsButtonMenu.Dispose();
            FileStream filestreamQuitButtonMenu = new
FileStream("Content/QuitButtonMenu.png", FileMode.Open);
            QuitButtonMenu = Texture2D.FromStream(GraphicsDevice,
filestreamQuitButtonMenu);
            filestreamQuitButtonMenu.Dispose();
            FileStream filestreamLayoutBackGroundIntro = new
FileStream("Content/LayoutBackGroundIntro.png", FileMode.Open);
            LayoutBackGroundIntro = Texture2D.FromStream(GraphicsDevice,
filestreamLayoutBackGroundIntro);
            filestreamLayoutBackGroundIntro.Dispose();
            FileStream filestreamLayoutBackGroundLvl1 = new
FileStream("Content/LayoutBackGroundLvl1.png", FileMode.Open);
            LayoutBackGroundLvl1 = Texture2D.FromStream(GraphicsDevice,
filestreamLayoutBackGroundLvl1);
            filestreamLayoutBackGroundLvl1.Dispose();
            FileStream filestreamLayoutBackGroundLvl2 = new
FileStream("Content/LayoutBackGroundLvl2.png", FileMode.Open);
            LayoutBackGroundLvl2 = Texture2D.FromStream(GraphicsDevice,
filestreamLayoutBackGroundLvl2);
            filestreamLayoutBackGroundLvl2.Dispose();
            FileStream filestreamSpaceship = new FileStream("Content/Spaceship.png",
FileMode.Open);
            Spaceship = Texture2D.FromStream(GraphicsDevice, filestreamSpaceship);
            filestreamSpaceship.Dispose();
            FileStream filestreamastronaute1 = new FileStream("Content/astronaute1.png",
```

```
FileMode.Open);
            astronaute1 = Texture2D.FromStream(GraphicsDevice, filestreamastronaute1);
            filestreamastronaute1.Dispose();
            FileStream filestreamBulletSpaceship = new
FileStream("Content/BulletSpaceship.png", FileMode.Open);
            BulletSpaceship = Texture2D.FromStream(GraphicsDevice,
filestreamBulletSpaceship);
            filestreamBulletSpaceship.Dispose();
            FileStream filestreamEnd = new FileStream("Content/End.png", FileMode.Open);
            End = Texture2D.FromStream(GraphicsDevice, filestreamEnd);
            filestreamEnd.Dispose();
            FileStream filestreamlunarsoil = new FileStream("Content/lunarsoil.png",
FileMode.Open);
            lunarsoil = Texture2D.FromStream(GraphicsDevice, filestreamlunarsoil);
            filestreamlunarsoil.Dispose();
            FileStream filestreamstickofflag = new FileStream("Content/stickofflag.png",
FileMode.Open);
            stickofflag = Texture2D.FromStream(GraphicsDevice, filestreamstickofflag);
            filestreamstickofflag.Dispose();
            FileStream filestreamamericanflag = new FileStream("Content/americanflag.png",
FileMode.Open);
            americanflag = Texture2D.FromStream(GraphicsDevice, filestreamamericanflag);
            filestreamamericanflag.Dispose();
            FileStream filestreamAsteroid = new FileStream("Content/Asteroid.png",
FileMode.Open);
            Asteroid = Texture2D.FromStream(GraphicsDevice, filestreamAsteroid);
            filestreamAsteroid.Dispose();
            FileStream filestreamGameover = new FileStream("Content/Gameover.png",
FileMode.Open);
            Gameover = Texture2D.FromStream(GraphicsDevice, filestreamGameover);
            filestreamGameover.Dispose();
            Content.RootDirectory = "Content";
            AsteroidFont = Content.Load<SpriteFont>("AsteroidFont");
            FileStream filestreamBacgroundOptions = new
FileStream("Content/BacgroundOptions.png", FileMode.Open);
            BacgroundOptions = Texture2D.FromStream(GraphicsDevice,
filestreamBacgroundOptions);
            filestreamBacgroundOptions.Dispose();
            FileStream filestreamTextesOptions = new
FileStream("Content/TextesOptions.png", FileMode.Open);
            TextesOptions = Texture2D.FromStream(GraphicsDevice, filestreamTextesOptions);
            filestreamTextesOptions.Dispose();
            FileStream filestreamARROWSOptions = new
FileStream("Content/ARROWSOptions.png", FileMode.Open);
            ARROWSOptions = Texture2D.FromStream(GraphicsDevice, filestreamARROWSOptions);
            filestreamARROWSOptions.Dispose();
            FileStream filestreamWASDOptions = new FileStream("Content/WASDOptions.png",
FileMode.Open);
            WASDOptions = Texture2D.FromStream(GraphicsDevice, filestreamWASDOptions);
            filestreamWASDOptions.Dispose();
            FileStream filestreamZQSDOptions = new FileStream("Content/ZQSDOptions.png",
FileMode.Open);
            ZQSDOptions = Texture2D.FromStream(GraphicsDevice, filestreamZQSDOptions);
            filestreamZQSDOptions.Dispose();
            FileStream filestreamenemy1 = new FileStream("Content/enemy1.png",
FileMode.Open);
            enemy1 = Texture2D.FromStream(GraphicsDevice, filestreamenemy1);
            filestreamenemy1.Dispose();
            /*using (var s = File.OpenRead("Content/sound.wav"))
            {
                var soundEffect = SoundEffect.FromStream(s);
                sound = soundEffect.CreateInstance();
            }
            FileStream filestreamsound = new FileStream("Content/sound.wav",
FileMode.Open);
            sound = SoundEffect.FromStream(filestreamsound);
```

```
    filestreamsound.Dispose();
    sound.Play();
    Content.RootDirectory = "Content";
    this.sound = Content.Load<Song>("sound");
    MediaPlayer.Play(sound);
    MediaPlayer.MediaStateChanged += MediaPlayer_MediaStateChanged;*/
    // TODO: use this.Content to load your game content here
}
/*void MediaPlayer_MediaStateChanged(object sender, System.EventArgs e)
{
    MediaPlayer.Volume -= 0.1f;
    MediaPlayer.Play(sound);
}   */
/// <summary>
/// UnloadContent will be called once per game and is the place to unload
/// game-specific content.
/// </summary>
protected override void UnloadContent()
{
    // TODO: Unload any non ContentManager content here
}

public void moveAsteroid()
{
    AsteroidPosition.X = AsteroidPosition.X - 2;
    AsteroidPosition.Y = AsteroidPosition.Y + 1;
    if (AsteroidPosition.X < -100 | AsteroidNotAlive)
    {
        if (AsteroidPosition.X < -100)
            GameOverTrue = true;
        AsteroidPosition = new Vector2(1000, WindowsHeight - 400);
        AsteroidNotAlive = false;
    }
}
public void moveenemy1()
{
    rndmoveenemy1count = rndmoveenemy1count + 1;
    if (rndmoveenemy1count > 200)
    {
        if (enemy1Position.X > 200)
            rndmoveenemy1.X = rnd.Next(-1, 1);
        else
            rndmoveenemy1.X = 1;
        if (enemy1Position.X < WindowsWidth - 200)
            rndmoveenemy1.X = rnd.Next(-1, 1);
        else
            rndmoveenemy1.X = -1;
        if (enemy1Position.Y > 100)
            rndmoveenemy1.Y = rnd.Next(-1, 1);
        else
            rndmoveenemy1.Y = 1;
        if (enemy1Position.Y < WindowsHeight - 100)
            rndmoveenemy1.Y = rnd.Next(-1, 1);
        else
            rndmoveenemy1.Y = -1;
        rndmoveenemy1count = 0;
    }
    if (rndmoveenemy1count < 100)
    {
        enemy1RandomPosition.X = enemy1RandomPosition.X + rndmoveenemy1.X;
        enemy1RandomPosition.Y = enemy1RandomPosition.Y + rndmoveenemy1.Y;
    }
    else
    {
        if (enemy1RandomPosition.X > 0)
            enemy1RandomPosition.X = enemy1RandomPosition.X - 1;
```

```
                else
                    enemy1RandomPosition.X = enemy1RandomPosition.X + 1;
                if (enemy1RandomPosition.Y > 0)
                    enemy1RandomPosition.Y = enemy1RandomPosition.Y - 1;
                else
                    enemy1RandomPosition.Y = enemy1RandomPosition.Y + 1;
            }
            if (enemy1OriginPosition.X + enemy1SearchSpaceshipPosition.X >
SpaceshipPosition.X)
                enemy1SearchSpaceshipPosition.X = enemy1SearchSpaceshipPosition.X - 1;
            else
                enemy1SearchSpaceshipPosition.X = enemy1SearchSpaceshipPosition.X + 1;
            if (enemy1OriginPosition.Y + enemy1SearchSpaceshipPosition.Y >
SpaceshipPosition.Y)
                enemy1SearchSpaceshipPosition.Y = enemy1SearchSpaceshipPosition.Y - 1;
            else
                enemy1SearchSpaceshipPosition.Y = enemy1SearchSpaceshipPosition.Y + 1;
            enemy1Position.X = enemy1OriginPosition.X + enemy1SearchSpaceshipPosition.X +
enemy1RandomPosition.X;
            enemy1Position.Y = enemy1OriginPosition.Y + enemy1SearchSpaceshipPosition.Y +
enemy1RandomPosition.Y;
            if (enemy1NotAlive)
            {
                enemy1RandomPosition = new Vector2(0, 0);
                enemy1SearchSpaceshipPosition = new Vector2(0, 0);
                enemy1OriginPosition = new Vector2(800, WindowsHeight - 650);
                enemy1NotAlive = false;
            }
        }
        /// <summary>
        /// Allows the game to run logic such as updating the world,
        /// checking for collisions, gathering input, and playing audio.
        /// </summary>
        /// <param name="gameTime">Provides a snapshot of timing values.</param>
        protected override void Update(GameTime gameTime)
        {
            if ((((GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed ||
Keyboard.GetState().IsKeyDown(Keys.Escape))) | (textesMenuPosition.Y == 3 &
(GamePad.GetState(PlayerIndex.One).Buttons.Start == ButtonState.Pressed || Key-
board.GetState().IsKeyDown(Keys.Space)))) && ChainCount == 0)
                Exit();
            if ((GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed ||
Keyboard.GetState().IsKeyDown(Keys.Escape) || (GameOverTrue & GameoverCount > 91)) &&
ChainCount > 0)
            {
                if (ChainCount != 0 & ChainCount != 3)
                {
                    ChainCountSaved = ChainCount;
                    using (var file = new System.IO.StreamWriter("test.dat"))
                    {
                        file.WriteLine(ChainCountSaved);
                        file.WriteLine(type);
                        file.Close();
                    }
                }
                ChainCount = 0;
                GameoverCount = 0;
                System.Threading.Thread.Sleep(100);
            }
            if ((ChainCount == 0 & textesMenuPosition.Y == 0 &
(GamePad.GetState(PlayerIndex.One).Buttons.Start == ButtonState.Pressed || Key-
board.GetState().IsKeyDown(Keys.Space))))
            {
                try
                {
                    using (var data = new System.IO.StreamReader("test.dat"))
```

```
                    {
                        ChainCount = int.Parse(data.ReadLine());
                        type = data.ReadLine();
                        data.Close();
                    }
                }
                catch { }
            }
            if ((ChainCount == 0 & textesMenuPosition.Y == 2 &
(GamePad.GetState(PlayerIndex.One).Buttons.Start == ButtonState.Pressed || Key-
board.GetState().IsKeyDown(Keys.Space))))
            {
                ChainCount = -1;
            }
            if (ChainCount == -1)
            {
                cevent(3, GamePad.GetState(PlayerIndex.One).Buttons.Back ==
ButtonState.Pressed || Keyboard.GetState().IsKeyDown(Keys.Escape));
                if (WU[3] >= 10)
                {
                    try
                    {
                        using (var data = new System.IO.StreamReader("test.dat"))
                        {
                            ChainCountStock = int.Parse(data.ReadLine());
                            data.Close();
                        }
                        using (var file = new System.IO.StreamWriter("test.dat"))
                        {
                            file.WriteLine(ChainCountStock);
                            file.WriteLine(type);
                            file.Close();
                        }
                    }
                    catch { }
                    ChainCount = 0;
                    WU[3] = 0;
                    System.Threading.Thread.Sleep(100);
                }
                if (GamePad.GetState(PlayerIndex.One).Buttons.A == ButtonState.Pressed ||
Keyboard.GetState().IsKeyDown(Keys.Enter))
                    WaitForOptions = true;
                if (WaitForOptions)
                {
                    assign();
                    WaitForOptionsCount = WaitForOptionsCount + 1;
                    if (WaitForOptionsCount > 90)
                    {
                        WaitForOptions = false;
                        WaitForOptionsCount = 0;
                    }
                }
            }
            // TODO: Add your update logic here
            if (ChainCount == 0)
            {
                cevent(1, (GamePad.GetState(PlayerIndex.One).DPad.Up == ButtonState.Pressed
|| Keyboard.GetState().IsKeyDown(KeyUp)));
                if (WD[1])
                {
                    textesMenuPosition.Y = textesMenuPosition.Y - 1;
                    WU[1] = 0;
                }
                cevent(2, (GamePad.GetState(PlayerIndex.One).DPad.Down ==
ButtonState.Pressed || Keyboard.GetState().IsKeyDown(KeyDown)));
                if (WD[2])
```

```
                {
                    textesMenuPosition.Y = textesMenuPosition.Y + 1;
                    WU[2] = 0;
                }
                if (textesMenuPosition.Y > 3)
                    textesMenuPosition.Y = 0;
                if (textesMenuPosition.Y < 0)
                    textesMenuPosition.Y = 3;
                BrightMenuCount = BrightMenuCount + 1;
                if (BrightMenuCount > 20)
                    BrightMenuCount = 0;
                SpaceshipPosition = new Vector2(400 - 160 / 2, 450);
                BulletsOriginPosition = new Vector2(400 - 160 / 2 + 70 / 2, 450);
                astronaute1Position = new Vector2(400 - 160 / 2, 450);
                GameOverTrue = false;
                AsteroidNotAlive = false;
                enemy1NotAlive = false;
                GameoverCount = 0;
                AsteroidRotation = 0;
                BulletsPosition = new Vector2(-200, -200);
                AsteroidPosition = new Vector2(1000, WindowsHeight - 400);
                enemy1RandomPosition = new Vector2(0, 0);
                enemy1SearchSpaceshipPosition = new Vector2(0, 0);
                enemy1OriginPosition = new Vector2(800, WindowsHeight - 650);
                WaitForOptionsCount = 0;
                AsteroidDestroyCount = 0;
                LvlEndCount = 0;
            }
            cevent(4, GamePad.GetState(PlayerIndex.One).Buttons.A == ButtonState.Pressed ||
Keyboard.GetState().IsKeyDown(Keys.Space));
            if (WU[4] >= 1)
            {
                BulletsOriginPosition.X = BulletsOriginPosition.X + 10;
                BulletsOriginPosition.Y = BulletsOriginPosition.Y - 10;
                BulletsPosition.X = BulletsOriginPosition.X + BulletsInitialPosition.X;
                BulletsPosition.Y = BulletsOriginPosition.Y + BulletsInitialPosition.Y;
            }
            else
            {
                if (ChainCount == 1)
                {
                    BulletsInitialPosition.X = SpaceshipPosition.X + 70 / 2;
                    BulletsInitialPosition.Y = SpaceshipPosition.Y;
                }
                if (ChainCount == 2)
                {
                    BulletsInitialPosition.X = 400 - 160 / 2;
                    BulletsInitialPosition.Y = astronaute1Position.Y + 140 / 2;
                }
                BulletsOriginPosition.X = 0;
                BulletsOriginPosition.Y = 0;
                BulletsPosition.X = -200;
                BulletsPosition.Y = -200;
            }
            if (WU[4] >= 20)
                WU[4] = 0;
            cevent(5, (GamePad.GetState(PlayerIndex.One).Buttons.A == ButtonState.Pressed
|| Keyboard.GetState().IsKeyDown(Keys.Space)) & WU[4] >= 10);
            if (WU[5] >= 1)
            {
                Bullets2ndOriginPosition.X = Bullets2ndOriginPosition.X + 10;
                Bullets2ndOriginPosition.Y = Bullets2ndOriginPosition.Y - 10;
                Bullets2ndPosition.X = Bullets2ndOriginPosition.X + Bul-
lets2ndInitialPosition.X;
                Bullets2ndPosition.Y = Bullets2ndOriginPosition.Y + Bul-
lets2ndInitialPosition.Y;
```

```
            }
            else
            {
                if (ChainCount == 1)
                {
                    Bullets2ndInitialPosition.X = SpaceshipPosition.X + 70 / 2;
                    Bullets2ndInitialPosition.Y = SpaceshipPosition.Y;
                }
                if (ChainCount == 2)
                {
                    Bullets2ndInitialPosition.X = 400 - 160 / 2;
                    Bullets2ndInitialPosition.Y = astronaute1Position.Y + 140 / 2;
                }
                Bullets2ndOriginPosition.X = 0;
                Bullets2ndOriginPosition.Y = 0;
                Bullets2ndPosition.X = -100;
                Bullets2ndPosition.Y = -100;
            }
            if (WU[5] >= 20)
                WU[5] = 0;
            AsteroidHitBox = new Rectangle((int)AsteroidPosition.X,
(int)AsteroidPosition.Y, 64, 63);
            if (ChainCount == 1)
            {
                BulletsHitBox = new Rectangle((int)(BulletsInitialPosition.X),
(int)(BulletsPosition.Y), 10, 40);
                Bullets2ndHitBox = new Rectangle((int)(Bullets2ndInitialPosition.X),
(int)(Bullets2ndPosition.Y), 10, 40);
                SpaceshipHitBox = new Rectangle((int)SpaceshipPosition.X,
(int)SpaceshipPosition.Y, 70, 180);
                enemy1HitBox = new Rectangle((int)enemy1Position.X, (int)enemy1Position.Y,
100, 50);
                if ((GamePad.GetState(PlayerIndex.One).DPad.Left == ButtonState.Pressed ||
Keyboard.GetState().IsKeyDown(KeyLeft)))
                    SpaceshipPosition.X = SpaceshipPosition.X - 1;
                if ((GamePad.GetState(PlayerIndex.One).DPad.Right == ButtonState.Pressed ||
Keyboard.GetState().IsKeyDown(KeyRight)))
                    SpaceshipPosition.X = SpaceshipPosition.X + 1;
                if ((GamePad.GetState(PlayerIndex.One).DPad.Up == ButtonState.Pressed ||
Keyboard.GetState().IsKeyDown(KeyUp)))
                    SpaceshipPosition.Y = SpaceshipPosition.Y - 1;
                if ((GamePad.GetState(PlayerIndex.One).DPad.Down == ButtonState.Pressed ||
Keyboard.GetState().IsKeyDown(KeyDown)))
                    SpaceshipPosition.Y = SpaceshipPosition.Y + 1;
                if (SpaceshipHitBox.Intersects(AsteroidHitBox))
                {
                    GameOverTrue = true;
                    AsteroidNotAlive = true;
                }
                if (BulletsHitBox.Intersects(AsteroidHitBox) | Bul-
lets2ndHitBox.Intersects(AsteroidHitBox))
                    AsteroidNotAlive = true;
                moveAsteroid();
                if (SpaceshipHitBox.Intersects(enemy1HitBox))
                {
                    GameOverTrue = true;
                    enemy1NotAlive = true;
                }
                if (BulletsHitBox.Intersects(enemy1HitBox) | Bul-
lets2ndHitBox.Intersects(enemy1HitBox))
                    enemy1NotAlive = true;
                moveenemy1();
                if (Bullets2ndHitBox.Intersects(enemy1HitBox) |
BulletsHitBox.Intersects(enemy1HitBox) | Bullets2ndHitBox.Intersects(AsteroidHitBox) |
BulletsHitBox.Intersects(AsteroidHitBox))
                    AsteroidDestroyCount = AsteroidDestroyCount + 1;
```

```
            }
            if (ChainCount == 2)
            {
                astronaute1TruePosition.X = 400 - 160 / 2;
                astronaute1TruePosition.Y = astronaute1Position.Y;
                BulletsHitBox = new Rectangle((int)(BulletsPosition.X),
(int)(BulletsInitialPosition.Y), 40, 10);
                Bullets2ndHitBox = new Rectangle((int)(Bullets2ndPosition.X),
(int)(Bullets2ndInitialPosition.Y), 40, 10);
                astronaute1OnFlagHitBox = new Rectangle((int)astronaute1TruePosition.X +
astronaute1.Width / 2, (int)astronaute1Position.Y + 140, 1, 1);
                americanflagHitBox = new Rectangle((int)(-(astronaute1Position.X) +
WindowsWidth), WindowsHeight - 100 - 50 / 2, 100, 1);
                SpaceshipHitBox = new Rectangle((int)(-(astronaute1Position.X) +
WindowsWidth), WindowsHeight - 100 - 50 / 2 - 240, 70, 30);
                if ((GamePad.GetState(PlayerIndex.One).DPad.Left == ButtonState.Pressed ||
Keyboard.GetState().IsKeyDown(KeyLeft)))
                    astronaute1Position.X = astronaute1Position.X - 1;
                if ((GamePad.GetState(PlayerIndex.One).DPad.Right == ButtonState.Pressed ||
Keyboard.GetState().IsKeyDown(KeyRight)))
                    astronaute1Position.X = astronaute1Position.X + 1;
                if ((GamePad.GetState(PlayerIndex.One).DPad.Up == ButtonState.Pressed ||
Keyboard.GetState().IsKeyDown(KeyUp)))
                    astronaute1Position.Y = astronaute1Position.Y - 2;
                if ((GamePad.GetState(PlayerIndex.One).DPad.Down == ButtonState.Pressed ||
Keyboard.GetState().IsKeyDown(KeyDown)) & (astronaute1Position.Y <= 450))
                    astronaute1Position.Y = astronaute1Position.Y + 1;
                if (astronaute1Position.Y < 450 & !astro-
naute1OnFlagHitBox.Intersects(americanflagHitBox))
                    astronaute1Position.Y = astronaute1Position.Y + 1;
                astronaute1HitBox = new Rectangle((int)astronaute1TruePosition.X,
(int)astronaute1TruePosition.Y, 130, 140);
                if (astronaute1HitBox.Intersects(AsteroidHitBox))
                {
                    GameOverTrue = true;
                    AsteroidNotAlive = true;
                }
                moveAsteroid();
                if (BulletsHitBox.Intersects(AsteroidHitBox) | Bul-
lets2ndHitBox.Intersects(AsteroidHitBox))
                {
                    AsteroidNotAlive = true;
                    AsteroidDestroyCount = AsteroidDestroyCount + 1;
                }
            }
            AsteroidRotation = AsteroidRotation + (float)0.1;
            base.Update(gameTime);
        }
        public void ShootSpaceshipBullets()
        {
            spriteBatch.Draw(BulletSpaceship, new Rectan-
gle((int)(BulletsInitialPosition.X), (int)(BulletsPosition.Y), 10, 40), Color.White);
        }
        public void ShootastronauteBullets()
        {
            spriteBatch.Draw(BulletSpaceship, new Rectangle((int)(BulletsPosition.X),
(int)(BulletsInitialPosition.Y), 40, 10), Color.White);
        }
        public void ShootSpaceshipBullets2nd()
        {
            spriteBatch.Draw(BulletSpaceship, new Rectan-
gle((int)(Bullets2ndInitialPosition.X), (int)(Bullets2ndPosition.Y), 10, 40), Color.White);
        }
        public void ShootastronauteBullets2nd()
        {
            spriteBatch.Draw(BulletSpaceship, new Rectangle((int)(Bullets2ndPosition.X),
```

```
(int)(Bullets2ndInitialPosition.Y), 40, 10), Color.White);
        }
        public void SpaceshipAsteroids()
        {
            spriteBatch.Draw(Asteroid, AsteroidPosition, null, Color.White,
(float)AsteroidRotation, new Vector2(Asteroid.Width / 2, Asteroid.Height / 2), 1.0f,
SpriteEffects.None, 0);
        }
        public void astronauteAsteroids()
        {
            spriteBatch.Draw(Asteroid, AsteroidPosition, null, Color.White,
(float)AsteroidRotation, new Vector2(Asteroid.Width / 2, Asteroid.Height / 2), 1.0f,
SpriteEffects.None, 0);
        }
        public void Spaceshipenemy1()
        {
            spriteBatch.Draw(enemy1, new Rectangle((int)(enemy1Position.X),
(int)(enemy1Position.Y), 100, 50), Color.White);
        }
        public void ShowTextGameover()
        {
            spriteBatch.Draw(Gameover, new Rectangle(WindowsWidth / 2 - 300 / 2,
WindowsHeight / 2 - 275 / 2, 314, 210), Color.White);
        }
        /// <summary>
        /// This is called when the game should draw itself.
        /// </summary>
        /// <param name="gameTime">Provides a snapshot of timing values.</param>
        protected override void Draw(GameTime gameTime)
        {
            GraphicsDevice.Clear(Color.CornflowerBlue);
            // TODO: Add your drawing code here

            if (ChainCount == -1)
            {
                spriteBatch.Begin();
                spriteBatch.Draw(BacgroundOptions, new Rectangle(0, 0, WindowsWidth,
WindowsHeight), Color.White);
                spriteBatch.Draw(TextesOptions, new Rectangle(10, 100, TextesOptions.Width,
TextesOptions.Height), Color.White);
                if (!WaitForOptions & type == "ARROWS")
                    spriteBatch.Draw(ARROWSOptions, new Rectangle(10, 300,
ARROWSOptions.Width, ARROWSOptions.Height), Color.White);
                if (!WaitForOptions & type == "WASD")
                    spriteBatch.Draw(WASDOptions, new Rectangle(10, 300, WASDOptions.Width,
WASDOptions.Height), Color.White);
                if (!WaitForOptions & type == "ZQSD")
                    spriteBatch.Draw(ZQSDOptions, new Rectangle(10, 300, ZQSDOptions.Width,
ZQSDOptions.Height), Color.White);
                spriteBatch.End();
            }
            if (ChainCount == 0)
            {
                spriteBatch.Begin();
                spriteBatch.Draw(LayoutBackGroundIntro, new Rectangle(0, 0, WindowsWidth,
WindowsHeight), Color.White);
                spriteBatch.Draw(LayoutMenu, new Rectangle(WindowsWidth / 2 - 300 / 2,
WindowsHeight / 2 - 275 / 2, 300, 275), Color.White);
                spriteBatch.Draw(ContinueButtonMenu, new Rectangle(WindowsWidth / 2 - 260 /
2, WindowsHeight / 2 - 90 - 53 / 2, 260, 53), Color.White);
                spriteBatch.Draw(PlayButtonMenu, new Rectangle(WindowsWidth / 2 - 142 / 2,
WindowsHeight / 2 - 30 - 48 / 2, 142, 48), Color.White);
                spriteBatch.Draw(OptionsButtonMenu, new Rectangle(WindowsWidth / 2 - 226 /
2, WindowsHeight / 2 + 30 - 50 / 2, 226, 50), Color.White);
                spriteBatch.Draw(QuitButtonMenu, new Rectangle(WindowsWidth / 2 - 132 / 2,
WindowsHeight / 2 + 90 - 52 / 2, 132, 52), Color.White);
```

```
                if (textesMenuPosition.Y == 0 & BrightMenuCount > 10)
                {
                    spriteBatch.Draw(ContinueButtonMenu, new Rectangle(WindowsWidth / 2 -
(int)(260 * BrightMenuCoef) / 2, WindowsHeight / 2 - 90 - (int)(53 * BrightMenuCoef) / 2,
(int)(260 * BrightMenuCoef), (int)(53 * BrightMenuCoef)), Color.White);
                }
                if (textesMenuPosition.Y == 1 & BrightMenuCount > 10)
                {
                    spriteBatch.Draw(PlayButtonMenu, new Rectangle(WindowsWidth / 2 -
(int)(142 * BrightMenuCoef) / 2, WindowsHeight / 2 - 30 - (int)(48 * BrightMenuCoef) / 2,
(int)(142 * BrightMenuCoef), (int)(48 * BrightMenuCoef)), Color.White);
                }
                if (textesMenuPosition.Y == 2 & BrightMenuCount > 10)
                {
                    spriteBatch.Draw(OptionsButtonMenu, new Rectangle(WindowsWidth / 2 -
(int)(226 * BrightMenuCoef) / 2, WindowsHeight / 2 + 30 - (int)(50 * BrightMenuCoef) / 2,
(int)(226 * BrightMenuCoef), (int)(50 * BrightMenuCoef)), Color.White);
                }
                if (textesMenuPosition.Y == 3 & BrightMenuCount > 10)
                {
                    spriteBatch.Draw(QuitButtonMenu, new Rectangle(WindowsWidth / 2 -
(int)(132 * BrightMenuCoef) / 2, WindowsHeight / 2 + 90 - (int)(52 * BrightMenuCoef) / 2,
(int)(132 * BrightMenuCoef), (int)(52 * BrightMenuCoef)), Color.White);
                }
                spriteBatch.End();
            }
            if (((GamePad.GetState(PlayerIndex.One).Buttons.Start == ButtonState.Pressed ||
Keyboard.GetState().IsKeyDown(Keys.Space)) & ChainCount == 0 & textesMenuPosition.Y == 1) |
ChainCount == 1)
            {
                LvlEndCount = LvlEndCount + 1;
                LayoutBackGroundLvl1PositionCount = LayoutBackGroundLvl1PositionCount + 1;
                ChainCount = 1;
                spriteBatch.Begin();
                spriteBatch.Draw(LayoutBackGroundLvl1, new Rectangle(0,
(int)LayoutBackGroundLvl1PositionCount, WindowsWidth, WindowsHeight), Color.White);
                spriteBatch.Draw(LayoutBackGroundLvl1, new Rectangle(0,
(int)LayoutBackGroundLvl1PositionCount - WindowsHeight, WindowsWidth, WindowsHeight), Col-
or.White);
                if (LayoutBackGroundLvl1PositionCount >= 600)
                    LayoutBackGroundLvl1PositionCount = 0;
                if (LvlEndCount > 2000)
                {
                    ChainCount = 2;
                    LvlEndCount = 0;
                    LayoutBackGroundLvl1PositionCount = 0;
                }
                if (!GameOverTrue)
                {
                    spriteBatch.Draw(Spaceship, new Rectangle((int)SpaceshipPosition.X,
(int)SpaceshipPosition.Y, 70, 180), Color.White);
                    ShootSpaceshipBullets();
                    ShootSpaceshipBullets2nd();
                    if (!AsteroidNotAlive)
                        SpaceshipAsteroids();
                    if (!enemy1NotAlive)
                        Spaceshipenemy1();
                }
                else
                {
                    GameoverCount = GameoverCount + 1;
                    ShowTextGameover();
                }
                spriteBatch.DrawString(AsteroidFont, "score : " +
AsteroidDestroyCount.ToString(), new Vector2(50, 50), Color.Azure);
                //spriteBatch.DrawString(AsteroidFont, , new Vector2(20, 20), Color.White,
```

```
(float)0, new Vector2(20, 20), new Vector2(20, 20), SpriteEffects.None, (float)0);
                spriteBatch.End();
            }
            if (ChainCount == 2)
            {
                LvlEndCount = LvlEndCount + 1;
                spriteBatch.Begin();
                spriteBatch.Draw(LayoutBackGroundLvl2, new Rectangle((int)(-
(astronaute1Position.X)), 0, WindowsWidth, WindowsHeight), Color.White);
                spriteBatch.Draw(LayoutBackGroundLvl2, new Rectangle((int)(-
(astronaute1Position.X) + WindowsWidth), 0, WindowsWidth, WindowsHeight), Color.White);
                spriteBatch.Draw(lunarsoil, new Rectangle((int)(-(astronaute1Position.X)),
WindowsHeight - 50, WindowsWidth, 50), Color.White);
                spriteBatch.Draw(lunarsoil, new Rectangle((int)(-(astronaute1Position.X) +
WindowsWidth), WindowsHeight - 50, WindowsWidth, 50), Color.White);
                spriteBatch.Draw(stickofflag, new Rectangle((int)(-(astronaute1Position.X)
+ WindowsWidth), WindowsHeight - 100 - 50 / 2, 6, 100), Color.White);
                spriteBatch.Draw(americanflag, new Rectangle((int)(-(astronaute1Position.X)
+ WindowsWidth), WindowsHeight - 100 - 50 / 2, 100, 60), Color.White);
                if (astronaute1Position.X >= 800 | astronaute1Position.X <= 0)
                    astronaute1Position.X = 0;
                spriteBatch.Draw(Spaceship, new Rectangle((int)(-(astronaute1Position.X) +
WindowsWidth), WindowsHeight - 100 - 50 / 2 - 240, 70, 180), Color.White);
                if (SpaceshipHitBox.Intersects(astronaute1HitBox))
                {
                    ChainCount = 3;
                    LvlEndCount = 0;
                }
                if (!GameOverTrue)
                {
                    spriteBatch.Draw(astronaute1, new Rectan-
gle((int)astronaute1TruePosition.X, (int)astronaute1TruePosition.Y, 130, 140), Col-
or.White);
                    ShootastronauteBullets();
                    ShootastronauteBullets2nd();
                    if (!AsteroidNotAlive)
                        astronauteAsteroids();
                }
                else
                {
                    GameoverCount = GameoverCount + 1;
                    ShowTextGameover();
                }
                spriteBatch.DrawString(AsteroidFont, "score : " +
AsteroidDestroyCount.ToString(), new Vector2(50, 50), Color.Azure);
                //spriteBatch.DrawString(AsteroidFont, AsteroidDestroyCount.ToString(), new
Vector2(20, 20), Color.White, (float)0, new Vector2(20, 20), new Vector2(20, 20),
SpriteEffects.None, (float)0);
                spriteBatch.End();
            }
            if (ChainCount == 3)
            {
                LvlEndCount = LvlEndCount + 1;
                if (LvlEndCount > 200)
                {
                    ChainCount = 0;
                    LvlEndCount = 0;
                }
                spriteBatch.Begin();
                spriteBatch.Draw(LayoutBackGroundIntro, new Rectangle(0, 0, WindowsWidth,
WindowsHeight), Color.White);
                spriteBatch.Draw(End, new Rectangle(WindowsWidth / 2 - End.Width / 2,
WindowsHeight / 2 - End.Height / 2, End.Width, End.Height), Color.White);
                spriteBatch.End();
            }
            base.Draw(gameTime);
```

```
        }
    }
}
```

6. Use and Agreement Contract

Owner: Michael Andre Franiatte.

Contact: michael.franiatte@gmail.com.

Owning: All works from scratch of the owner.

Proof of Owning: Works published, and writings/speakings all over.

Requirements of Use: Pay the owner, quote the owner, agreement of the owner.

Availability of Works: Only under the shapes of the owner built, only for personal use.

Subjects of Claims: Works published by the owner on Google Play and Google Books.

Concerning Author Rights: Equations and codes from scratch of the owner, softwares built from it, all things of people arising from it.

End User License Agreement: A commercial license is required to use in personal manner. Do not redistributing in any manner, including by computer media, a file server, an email attachment, etc. Do not embedding in or linking it to another programs, source codes and assistances including internal applications, scripts, batch files, etc. Do not use for any kind of technical support including on customer or retailer computer, hardware or software development, research, discovery, teachery, talk, speech, write, etc. Do not use for win money or for commercialisation of any products arising from my programs, source codes and assistances. Do not use and do not copy the way it run in other programs, source codes and assistances. Do not use without pay me, quote me and my agreement. Do not steal or copy or reproduce or modify or peer or share. Do not use in other manner than personal. It stand for my programs, source codes and assistances or programs, source codes and assistances stealing or copying or reproducing or modifying or peering or sharing my programs, source codes, and assistances. If you aren't agree you shall not use.

Terms of License and Price: The present contract acceptance is required to use works of the owner and built from it in all kind of manner. The price for each user shall be defined with the owner by contacting him and this for each subject of works the owner claims. Each user shall contact the owner for asking his agreement. It can be refused by the owner depending who asking and the price defined. People don’t respecting the present contract shall not use the works of the owner.