ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA CÔNG NGHỆ PHẦN MỀM

BÁO CÁO ĐỒ ÁN MÔN HỌC

LẬP TRÌNH TRỰC QUAN

TÊN ĐỀ TÀI

2D SPACE SHOOTER

Môn học : **Lập trình trực quan**

Giảng viên lý thuyết : **Phan Nguyệt Minh**

Giảng viên thực hành : **Huỳnh Hồ Thị Mộng Trinh**

Nhóm thực hiện : **Nguyễn Công Minh - 16520740**

TP. Hồ Chí Minh, tháng 6 năm 2018

LỜI CẢM ƠN

Trong suốt quá trình học tập môn **Lập trình trực quan** và hoàn thành đồ án **2D Space Shooter**, nhóm đã nhận được những kiến thức vô cùng bổ ích từ **cô Phan Nguyệt Minh và cô Huỳnh Hồ Thị Mộng Trinh**. Thông qua việc hoàn thành đồ án, em xin được gửi lời cám ơn đến hai cô vì sự tận tâm và vì những gì mà hai cô đã truyền đạt cho chúng em.

Trân trọng.

TP. Hồ Chí Minh, ngày 30 tháng 06 năm 2018

NHÓM THỰC HIỆN

NHẬN XÉT CỦA GIẢNG VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

1 GIỚI THIỆU ĐỀ TÀI

1.1 Bối cảnh thực tiễn

- Việc thám hiểm vũ trụ đã được các nhà khoa học thực hiện từ những năm 50 của thế kỉ XX. Nó bắt đầu từ cuộc chạy đua vào vũ trụ hay cuộc chạy đua vào không gian (là cuộc cạnh tranh thám hiểm vũ trụ giữa Hoa Kỳ và Liên Xô) kéo dài từ khoảng 1957 đến 1975. Và nó liên quan đến các nỗ lực khám hiểm không gian bằng các vệ tinh nhân tạo và việc đưa con người vào vũ trụ và lên mặt trăng.
- Ngày nay, dù cuộc chạy đua vào vũ trụ không còn nhưng các quốc gia trên thế vẫn đang tiếp tục nghiên cứu vũ trụ và các hành tinh trong hệ ngân hà để có thể tìm hiểu rõ hơn những gì đang có trong vũ trụ rộng lớn này.
- Đề tài vũ trụ cũng được các nhà làm phim chú ý đến và có những phim nổi tiếng về đề tài này như Star Wars (Chiến tranh giữa các vì sao), Star Trek (Du hành giữa các vì sao), Alien (Quái vật không gian, Người ngoài hành tinh),…
- Alien hay sinh vật ngoài Trái đất là những sinh vật tồn tại ngoài Trái đất. Sự tồn tại của nó đến nay vẫn chỉ là giả thuyết và vẫn chưa có bằng chứng cụ thể về sự sống ngoài Trái đất như các nhà khoa học đã công nhận một cách rộng rãi. Tuy nhiên, alien là một đề tài mà người làm phim cũng như làm game không thể không dùng đến trong các dự án game cũng như phim. Dù các biến thể của alien trong mỗi dự án là khác nhau nhưng không thể bỏ qua được nội dung về alien trong mỗi dự án

1.2 Bối cảnh game

- Trong bối cảnh loài người đang không ngừng tăng thêm về số lượng thì năm 2050, các quốc gia trên thế giới đã họp lại và quyết định cử những cá nhân cực kì xuất sắc từ khắp nơi trên thế giới để lái những con tàu cực kì tối tân, có khả năng đi đến tất cả các ngóc ngách trong vũ trụ này mà không cần phải quay trở về nạp năng lượng.
- Sau 3 năm tìm kiếm, bỗng một ngày các thành viên phi hành đoàn gặp những phi thuyền khác. Cứ ngỡ là bạn từ Trái đất nhưng không, họ là phi thuyền của những người ngoài hành tinh. Bọn người ngoài hành tinh cực kì hiếu chiến đã nã đạn vào phi thuyền của các phi hành gia Trái đất, buộc họ phải chiến đấu.

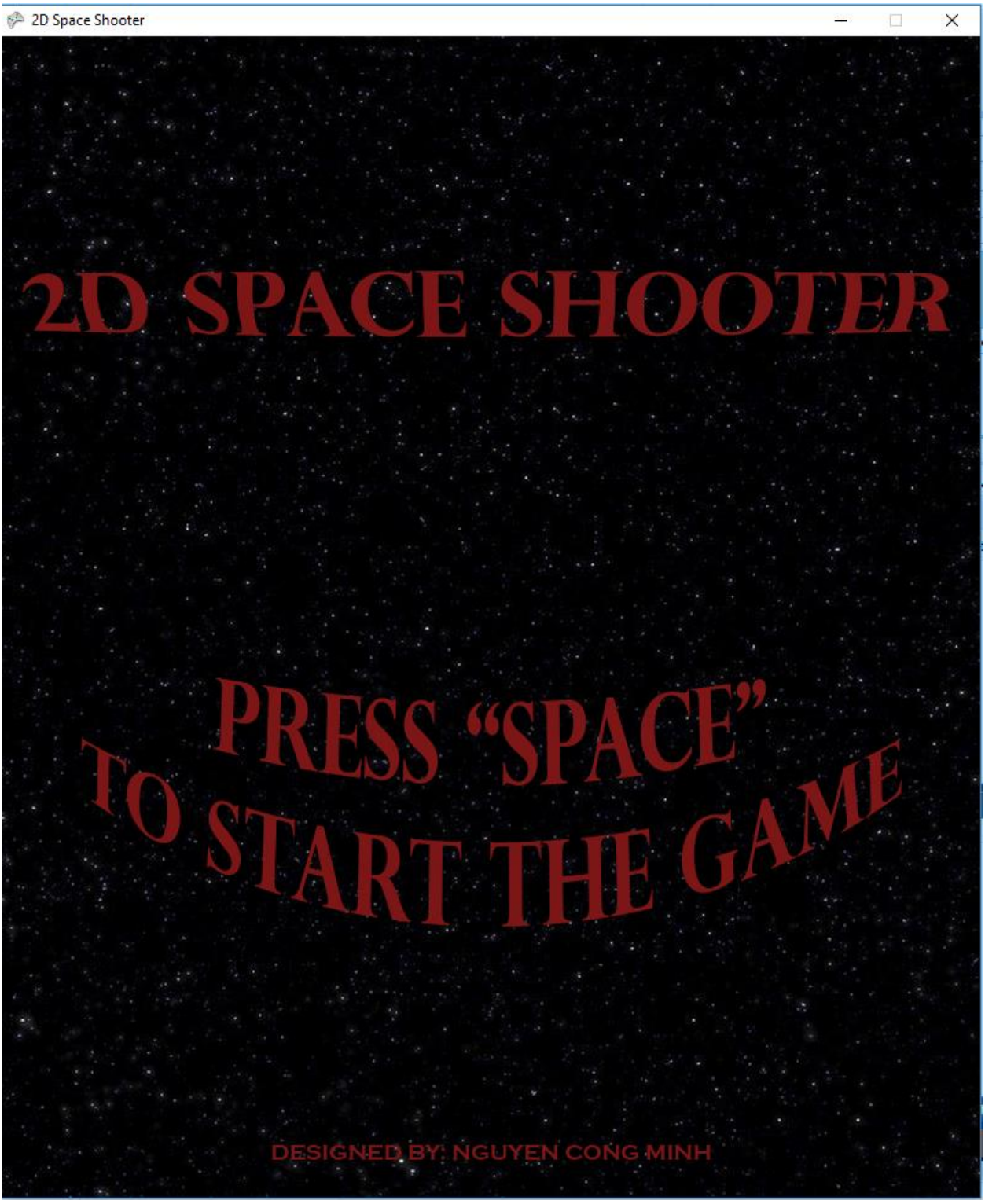
1.3 Lý do thực hiện

- Nhằm tạo một game mà mọi người có thể chơi nhằm giải trí sau những giờ học căng thẳng

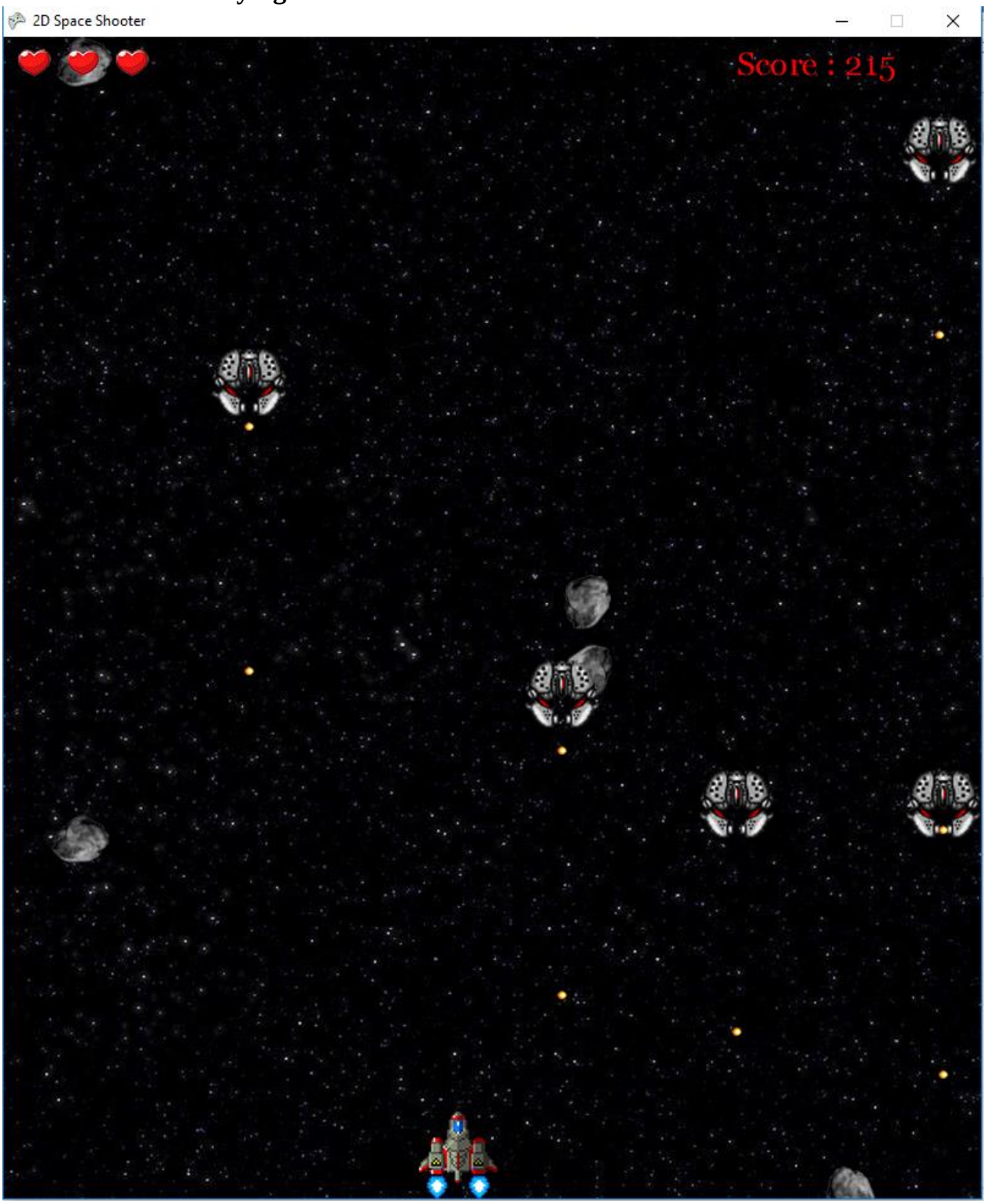
2 HIỆN THỰC

2.1 Giao diện

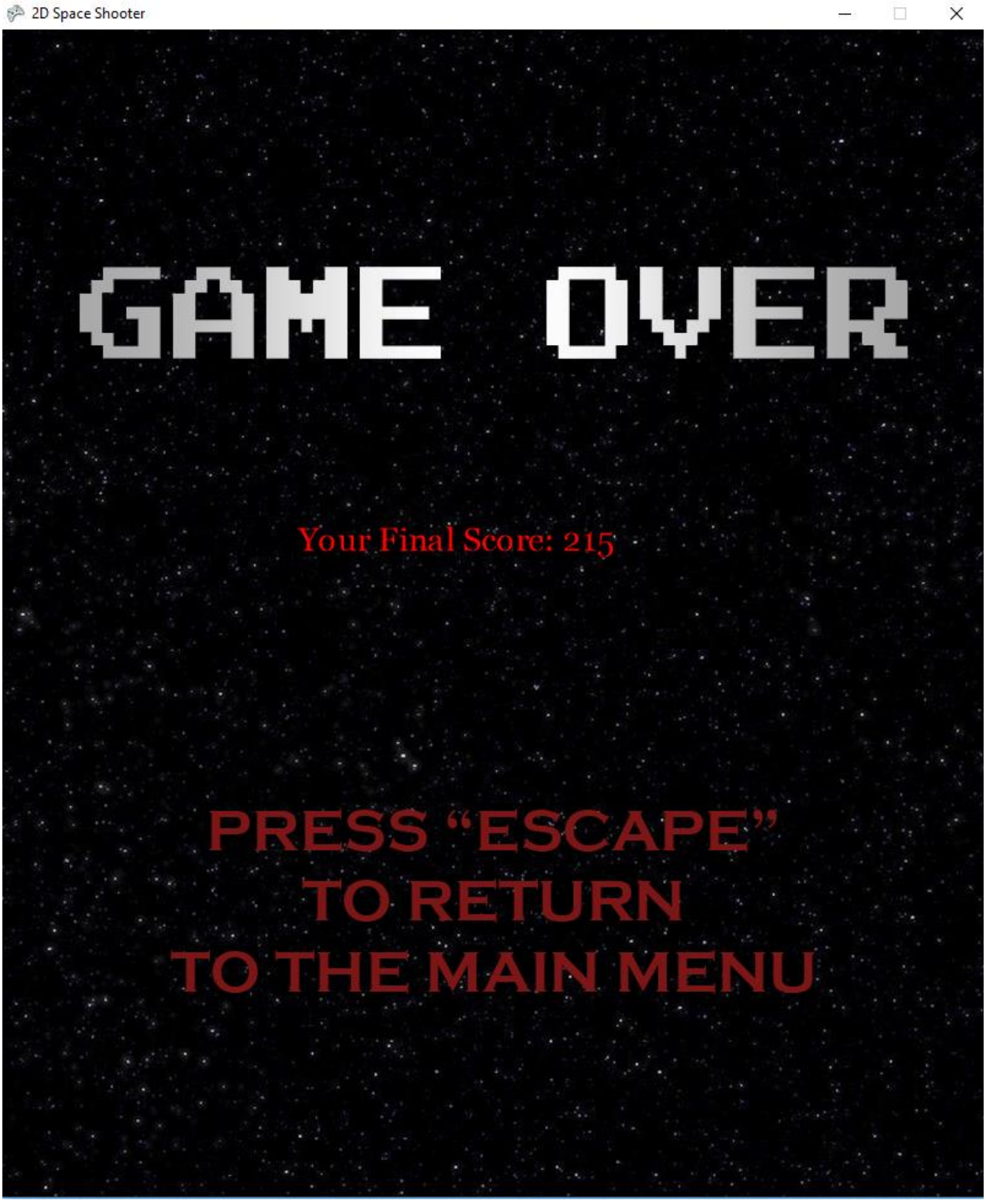
2.1.1 Màn hình Menu



2.1.2 Màn hình Playing



2.1.3 Màn hình Gameover



2.2 Chương trình thực thi

2.2.1 Class Player

- Class Player có nhiệm vụ load hình ảnh phi thuyền (ship), thanh mạng (life), và đạn của người chơi (bullet), khởi tạo vị trí ban đầu của phi thuyền trong mỗi lần vào chơi.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;

namespace _2DSpaceShooter
{
    public class Player
    {
        public Texture2D texture, bulletTexture, lifeTexture;
        public Vector2 position, lifeBarPosition;
        public int speed, life;
        public float bulletDelay;
        public Rectangle boundingBox, lifeRectangle;
        public bool isColliding;
        private List<Bullet> bulletList;

        internal List<Bullet> BulletList
        {
            get
            {
                return bulletList;
            }

            set
            {
                bulletList = value;
            }
        }

        // Constructor
        public Player()
        {
            BulletList = new List<Bullet>();
            texture = null;
            position = new Vector2(400, 800);
            bulletDelay = 10;
            speed = 10;
            isColliding = false;
            life = 10;
        }

        // Load Content
        public void LoadContent(ContentManager Content)
        {
            texture = Content.Load<Texture2D>("ship");
            bulletTexture = Content.Load<Texture2D>("bullet");
```

```
            lifeTexture = Content.Load<Texture2D>("life");
        }

        // Draw
        public void Draw(SpriteBatch spriteBatch)
        {
            // draw ship
            spriteBatch.Draw(texture, position, Color.White);

            // draw bullet
            foreach (Bullet b in BulletList)
                b.Draw(spriteBatch);

            // draw lifeBar
            Vector2 lifePosition = new Vector2(10, 10);
            for (int i = 0; i < life; i++)
            {
                spriteBatch.Draw(lifeTexture, lifePosition, Color.White);
                lifePosition.X += lifeTexture.Width + 10;
            }
        }

        // Update
        public void Update(GameTime gameTime)
        {
            // Getting Keyboard State
            KeyboardState keyState = Keyboard.GetState();

            // BoundingBox for our PlayerShip
            boundingBox = new Rectangle((int)position.X, (int)position.Y, texture.Width,
texture.Height);

            // Set Rectangle for lifeBar
            lifeRectangle = new Rectangle((int)lifeBarPosition.X, (int)lifeBarPosition.Y,
life, 25);


            // Fire Bullet
            if (keyState.IsKeyDown(Keys.Space))
            {
                Shoot();
            }

            UpdateBullets();

            // Ship Controls
            // Up
            if (keyState.IsKeyDown(Keys.W))
                position.Y = position.Y - speed;
            // Down
            if (keyState.IsKeyDown(Keys.A))
                position.X = position.X - speed;
            // Left
            if (keyState.IsKeyDown(Keys.S))
                position.Y = position.Y + speed;
            // Right
            if (keyState.IsKeyDown(Keys.D))
                position.X = position.X + speed;
```

```
            // keep Player Ship In Screen Bounds
            // minX = 0
            if (position.X <= 0)
                position.X = 0;
            // minX = maxX.ScreenBound - ship.Width
            if (position.X >= 800 - texture.Width)
                position.X = 800 - texture.Width;


            // minY = 0
            if (position.Y <= 0)
                position.Y = 0;
            // maxY = maxY.ScreenBound - ship.Height
            if (position.Y >= 950 - texture.Height)
                position.Y = 950 - texture.Height;
        }

        // Shoot Method: used to set starting position of out bullets
        public void Shoot()
        {
            // Shoot only if bullet delay resets
            if (bulletDelay >= 0)
                bulletDelay--;

            // If bulletDelay is at 0: create new bullet at player position, make it
visible on the screen, then add that bullet to the List
            if (bulletDelay <= 0)
            {
                Bullet newBullet = new Bullet(bulletTexture);
                newBullet.position = new Vector2(position.X + 32 -
newBullet.texture.Width / 2, position.Y + 30);

                newBullet.isVisible = true;

                if (BulletList.Count() < 20)
                    BulletList.Add(newBullet);
            }

            // reset bullet delay
            if (bulletDelay == 0)
                bulletDelay = 10;
        }

        // Update bullet function
        public void UpdateBullets()
        {
            // for each bullet in our bulletList: update the movement and if the bullet
hits the top of the screen remove it from the list
            foreach (Bullet b in BulletList)
            {
                // BoundingBox for our every bullet in our bulletList
                b.boundingBox = new Rectangle((int)b.position.X, (int)b.position.Y,
b.texture.Width, b.texture.Height);

                // set movement for bullet
                b.position.Y = b.position.Y - b.speed;
```

```
                // if bullet hits the top of the screen, then make visible flase
                if (b.position.Y <= 0)
                    b.isVisible = false;
            }

            // Iterate through bulletList and see if any of the bullets are not visible,
if they arent then remove that bullet from our bullet list
            for(int i = 0; i < BulletList.Count; i++)
            {
                if(!BulletList[i].isVisible)
                {
                    BulletList.RemoveAt(i);
                    i--;
                }
            }
        }

    }
}
```

- Constructor có nhiệm vụ tạo một BulletList của người chơi, khởi tạo vị trí ban đầu và các chức năng khác như tốc độ đạn, mạng sống

```
        // Constructor
        public Player()
        {
            BulletList = new List<Bullet>();
            texture = null;
            position = new Vector2(400, 800);
            bulletDelay = 10;
            speed = 10;
            isColliding = false;
            life = 10;
        }
```

- Load Content sẽ load các hình ảnh đã chuẩn bị trước

```
        // Load Content
        public void LoadContent(ContentManager Content)
        {
            texture = Content.Load<Texture2D>("ship");
            bulletTexture = Content.Load<Texture2D>("bullet");
            lifeTexture = Content.Load<Texture2D>("life");
         }
```

- Draw sẽ vẽ các hình ảnh lên màn hình. Trong đó, thanh mạng sống sẽ được vẽ lại trong mỗi lần Update dựa vào biến life. Vòng lặp for() có nhiệm vụ vẽ lại thanh mạng sống, mỗi một life sẽ được vẽ lên và cập nhật lại vị trí cho lần vẽ tiếp theo

```
        // Draw
        public void Draw(SpriteBatch spriteBatch)
        {
            // draw ship
            spriteBatch.Draw(texture, position, Color.White);
```

```
            // draw bullet
            foreach (Bullet b in BulletList)
                b.Draw(spriteBatch);

            // draw lifeBar
            Vector2 lifePosition = new Vector2(10, 10);
            for (int i = 0; i < life; i++)
            {
                spriteBatch.Draw(lifeTexture, lifePosition, Color.White);
                lifePosition.X += lifeTexture.Width + 10;
            }
        }
```

- Update được dùng để thực thi hàm Shoot() và hàm UpdateBullet() sau mỗi lần bắn, cũng như cài đặt các phím di chuyển cho phi thuyền của người chơi

```
        // Update
        public void Update(GameTime gameTime)
        {
            // Getting Keyboard State
            KeyboardState keyState = Keyboard.GetState();

            // BoundingBox for our PlayerShip
            boundingBox = new Rectangle((int)position.X, (int)position.Y, texture.Width,
texture.Height);

            // Set Rectangle for lifeBar
            lifeRectangle = new Rectangle((int)lifeBarPosition.X, (int)lifeBarPosition.Y,
life, 25);


            // Fire Bullet
            if (keyState.IsKeyDown(Keys.Space))
            {
                Shoot();
            }

            UpdateBullets();

            // Ship Controls
            // Up
            if (keyState.IsKeyDown(Keys.W))
                position.Y = position.Y - speed;
            // Down
            if (keyState.IsKeyDown(Keys.A))
                position.X = position.X - speed;
            // Left
            if (keyState.IsKeyDown(Keys.S))
                position.Y = position.Y + speed;
            // Right
            if (keyState.IsKeyDown(Keys.D))
                position.X = position.X + speed;

            // keep Player Ship In Screen Bounds
            // minX = 0
            if (position.X <= 0)
                position.X = 0;
            // minX = maxX.ScreenBound - ship.Width
```

```
            if (position.X >= 800 - texture.Width)
                position.X = 800 - texture.Width;


            // minY = 0
            if (position.Y <= 0)
                position.Y = 0;
            // maxY = maxY.ScreenBound - ship.Height
            if (position.Y >= 950 - texture.Height)
                position.Y = 950 - texture.Height;
        }
```

- Hàm Shoot() dùng để lấy vị trí hiện tại của player ship và lấy đó làm vị trí bắt đầu của viên đạn (kích thước của "ship" là 64x71 pixel nên công thức "`position.X + 32 - newBullet.texture.Width / 2`" nhằm lấy vị trí giữa "ship"). bulletDelay sẽ quyết định khoảng thời gian mà viên đạn xuất hiện kể từ lúc người dùng nhấn nút bắn. Khi số lượng đạn hiện tại của player nhỏ hơn 20 thì viên đạn mới sẽ được bắn ra (tức một bullet mới sẽ được thêm vào BulletList và được vẽ ra)

```
        // Shoot Method: used to set starting position of out bullets
        public void Shoot()
        {
            // Shoot only if bullet delay resets
            if (bulletDelay >= 0)
                bulletDelay--;

            // If bulletDelay is at 0: create new bullet at player position, make it
visible on the screen, then add that bullet to the List
            if (bulletDelay <= 0)
            {
                Bullet newBullet = new Bullet(bulletTexture);
                newBullet.position = new Vector2(position.X + 32 -
newBullet.texture.Width / 2, position.Y + 30);

                newBullet.isVisible = true;

                if (BulletList.Count() < 20)
                    BulletList.Add(newBullet);
            }

            // reset bullet delay
            if (bulletDelay == 0)
                bulletDelay = 10;
        }
```

- Hàm UpdateBullets() sẽ quyết định đường đi cũng như tốc độ di chuyển của viên đạn của người chơi bắn ra cũng như tốc độ di chuyển của nó. Nếu vị trí `b.position.Y` của viên đạn (tức tung độ) của viên đạn chạm vị trí cạnh trên của màn hình (tức tung độ bằng 0) thì viên đạn đó sẽ mất đi. Khi viên đạn mất đi thì nó sẽ được remove khỏi BulletList

```
        // Update bullet function
        public void UpdateBullets()
        {
```

```
            // for each bullet in our bulletList: update the movement and if the bullet
hits the top of the screen remove it from the list
            foreach (Bullet b in BulletList)
            {
                // BoundingBox for our every bullet in our bulletList
                b.boundingBox = new Rectangle((int)b.position.X, (int)b.position.Y,
b.texture.Width, b.texture.Height);

                // set movement for bullet
                b.position.Y = b.position.Y - b.speed;

                // if bullet hits the top of the screen, then make visible flase
                if (b.position.Y <= 0)
                    b.isVisible = false;
            }

            // Iterate through bulletList and see if any of the bullets are not visible,
if they arent then remove that bullet from our bullet list
            for(int i = 0; i < BulletList.Count; i++)
            {
                if(!BulletList[i].isVisible)
                {
                    BulletList.RemoveAt(i);
                    i--;
                }
            }
        }
```

2.2.2 Class Asteroid

- Class Asteroid có nhiệm vụ load hành ảnh thiên thạch lên màn hình cũng như cập nhật vị trí của thiên thạch

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;

namespace _2DSpaceShooter
{
    class Asteroid
    {
        public Rectangle boundingBox;
        public Texture2D texture;
        public Vector2 position;
        public Vector2 origin;
        public float rotationAngle;
        public int speed;

        public bool isVisible;
        Random random = new Random();
        public float randX, randY;

        // Constructor
```

```
        public Asteroid(Texture2D newTextture, Vector2 newPosition)
        {
            position = newPosition;
            texture = newTextture;
            speed = 4;
            isVisible = true;
            randX = random.Next(0, 750);
            randY = random.Next(-600, -50);
        }

        // Load Content
        public void LoadContent(ContentManager Content)
        {
        }

        // Update
        public void Update(GameTime gameTime)
        {
            // Set boundingBox for collision
            boundingBox = new Rectangle((int)position.X, (int)position.Y, 45, 45);

            // Updating origin for rotation
            origin.X = texture.Width / 2;
            origin.Y = texture.Height / 2;

            // Update Movement
            position.Y = position.Y + speed;
            if (position.Y >= 950)
                position.Y = -50;

            // Rotate Asteroid
            float elapsed = (float)gameTime.ElapsedGameTime.TotalSeconds;
            rotationAngle += elapsed;
            float circle = MathHelper.Pi * 2;
            rotationAngle = rotationAngle % circle;
        }

        // Draw
        public void Draw(SpriteBatch spriteBatch)
        {
            if(isVisible)
            {
                spriteBatch.Draw(texture, position, null, Color.White, rotationAngle,
origin, 1.0f, SpriteEffects.None, 0f);
            }
        }
    }
}
```

- Constructor sẽ khởi tạo tốc độ di chuyển của thiên thạch cũng như random vị trí bắt đầu của thiên thạch

```
        // Constructor
        public Asteroid(Texture2D newTextture, Vector2 newPosition)
        {
            position = newPosition;
```

```
            texture = newTextture;
            speed = 4;
            isVisible = true;
            randX = random.Next(0, 750);
            randY = random.Next(-600, -50);
        }
```

- Update có nhiệm vụ set boundingBox phục vụ cho xử lí va chạm được thực hiện bên class Game1. Bên cạnh đó là xử lí khi thiên thạch qua cạnh dưới của màn hình. Ngoài ra ở đây còn có nhiệm vụ thiết lập công thức cũng như xử lí để thiên thạch có thể quay

```
        // Update
        public void Update(GameTime gameTime)
        {
            // Set boundingBox for collision
            boundingBox = new Rectangle((int)position.X, (int)position.Y, 45, 45);

            // Updating origin for rotation
            origin.X = texture.Width / 2;
            origin.Y = texture.Height / 2;

            // Update Movement
            position.Y = position.Y + speed;
            if (position.Y >= 950)
                position.Y = -50;

            // Rotate Asteroid
            float elapsed = (float)gameTime.ElapsedGameTime.TotalSeconds;
            rotationAngle += elapsed;
            float circle = MathHelper.Pi * 2;
            rotationAngle = rotationAngle % circle;
        }
```

- Draw sẽ kiểm tra xem có cho xuất hiện thiên thạch ra màn hình hay không, nếu có thì sẽ vẽ ra

```
        // Draw
        public void Draw(SpriteBatch spriteBatch)
        {
            if(isVisible)
            {
                spriteBatch.Draw(texture, position, null, Color.White, rotationAngle,
origin, 1.0f, SpriteEffects.None, 0f);
            }
        }
```

2.2.3 Class Starfield

- Class Starfield sẽ có nhiệm vụ vẽ lên background và tạo di chuyển giúp game thực tế hơn

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

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

namespace _2DSpaceShooter
{
    public class Starfield
    {
        public Texture2D texture;
        public Vector2 bgPos1, bgPos2; //backgroundPosition
        public int speed;

        // Constructor
        public Starfield()
        {
            texture = null;
            bgPos1 = new Vector2(0, 0);
            bgPos2 = new Vector2(0, -950);
            speed = 5;
        }

        // Load Content
        public void LoadContent(ContentManager Content)
        {
            texture = Content.Load<Texture2D>("background");
        }

        // Draw
        public void Draw(SpriteBatch spriteBatch)
        {
            spriteBatch.Draw(texture, bgPos1, Color.White);
            spriteBatch.Draw(texture, bgPos2, Color.White);
        }

        // Update
        public void Update(GameTime gameTime)
        {
            bgPos1.Y = bgPos1.Y + speed;
            bgPos2.Y = bgPos2.Y + speed;

            // Scrolling beckground
            if(bgPos1.Y >= 950)
            {
                bgPos1.Y = 0;
                bgPos2.Y = -950;
            }
        }
    }
}
```

- Constructor sẽ khởi tạo hai bgPos1, bgPos2 và tốc độ di chuyển của background. bgPos1 và bgPos2 có nhiệm vụ chạy liên tục từ trên xuống dưới giúp người chơi có cảm giác đang đi tới. Và khi `bgPos1.Y = 0` thì nó sẽ reset lại cho lần xuất hiện tiếp theo, tránh để bị đứt quảng

```
        // Constructor
        public Starfield()
        {
            texture = null;
            bgPos1 = new Vector2(0, 0);
            bgPos2 = new Vector2(0, -950);
            speed = 5;
        }

        // Load Content
        public void LoadContent(ContentManager Content)
        {
            texture = Content.Load<Texture2D>("background");
        }

        // Draw
        public void Draw(SpriteBatch spriteBatch)
        {
            spriteBatch.Draw(texture, bgPos1, Color.White);
            spriteBatch.Draw(texture, bgPos2, Color.White);
        }

        // Update
        public void Update(GameTime gameTime)
        {
            bgPos1.Y = bgPos1.Y + speed;
            bgPos2.Y = bgPos2.Y + speed;

            // Scrolling background
            if(bgPos1.Y >= 950)
            {
                bgPos1.Y = 0;
                bgPos2.Y = -950;
            }
        }
```

2.2.4 Class Bullet

- Class Bullet dùng để khởi tạo các biến cũng như tốc độ di chuyển của đạn của phi thuyền người ngoài hành tinh

```
      using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;

namespace _2DSpaceShooter
{
    // Main
    class Bullet
    {
        public Rectangle boundingBox;
        public Texture2D texture;
        public Vector2 origin;
```

```
        public Vector2 position;
        public bool isVisible;
        public float speed;

        // Constructor
        public Bullet(Texture2D newTexture)
        {
            speed = 10;
            texture = newTexture;
            isVisible = false;
        }

        // Draw
        public void Draw(SpriteBatch spriteBatch)
        {
            spriteBatch.Draw(texture, position, Color.White);
        }
    }
}
```

2.2.5 Class Enemy

- Class Enemy có nhiệm vụ khởi tạo các biến cũng như giúp máy bay địch có thể bắn đạn

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;

namespace _2DSpaceShooter
{
    class Enemy
    {
        public Rectangle boundingBox;
        public Texture2D texture, bulletTexture;
        public Vector2 position;
        public int speed, bulletDelay;
        public bool isVisible;
        public List<Bullet> bulletList;

        // Constructor
        public Enemy(Texture2D newTexture, Vector2 newPosition, Texture2D
newBulletTexture)
        {
            bulletList = new List<Bullet>();
            texture = newTexture;
            bulletTexture = newBulletTexture;
            position = newPosition;
            bulletDelay = 60;
            speed = 5;
            isVisible = true;
        }
```

```
        // Update
        public void Update(GameTime gameTime)
        {
            // Update Collision Rectangle
            boundingBox = new Rectangle((int)position.X, (int)position.Y, texture.Width,
texture.Height);

            // Update Enemy Movement
            position.Y += speed;

            // Move enemy to top of the screen if he fly's off buttom
            if (position.Y >= 950)
                position.Y = -75;

            EnemyShoot();
            UpdateBullets();
        }

        // Draw
        public void Draw(SpriteBatch spriteBatch)
        {
            // Draw Enemy Ship
            spriteBatch.Draw(texture, position, Color.White);

            // Draw ememy bullets
            foreach (Bullet b in bulletList)
            {
                b.Draw(spriteBatch);
            }
        }

        // Update bullet function
        public void UpdateBullets()
        {
            // for each bullet in our bulletList: update the movement and if the bullet
hits the top of the screen remove it from the list
            foreach (Bullet b in bulletList)
            {
                // BoundingBox for our every bullet in our bulletList
                b.boundingBox = new Rectangle((int)b.position.X, (int)b.position.Y,
b.texture.Width, b.texture.Height);

                // set movement for bullet
                b.position.Y = b.position.Y + b.speed;

                // if bullet hits the top of the screen, then make visible flase
                if (b.position.Y >= 950)
                    b.isVisible = false;
            }

            // Iterate through bulletList and see if any of the bullets are not visible,
if they arent then remove that bullet from our bullet list
            for (int i = 0; i < bulletList.Count; i++)
            {
                if (!bulletList[i].isVisible)
                {
                    bulletList.RemoveAt(i);
```

```
                    i--;
                }
            }
        }

        // Enemy Shoot Function
        public void EnemyShoot()
        {
            // Shoot inly if bulletdelay resets
            if (bulletDelay >= 0)
                bulletDelay--;

            if(bulletDelay <= 0)
            {
                // Create new bullet and position it front and center of enemy ship
                Bullet newBullet = new Bullet(bulletTexture);
                newBullet.position = new Vector2(position.X + texture.Width / 2 -
newBullet.texture.Width / 2, position.Y + 30);

                newBullet.isVisible = true;

                if (bulletList.Count() < 20)
                    bulletList.Add(newBullet);
            }

            // reset bullet delay
            if (bulletDelay == 0)
                bulletDelay = 40;
        }
    }
}
```

- Constructor sẽ khởi tạo khoảng cách giữa các viên đạn (bulletDelay), tốc độ đạn di chuyển và cho đạn xuất hiện

```
        // Constructor
        public Enemy(Texture2D newTexture, Vector2 newPosition, Texture2D
newBulletTexture)
        {
            bulletList = new List<Bullet>();
            texture = newTexture;
            bulletTexture = newBulletTexture;
            position = newPosition;
            bulletDelay = 60;
            speed = 5;
            isVisible = true;
        }
```

- Update có nhiệm vụ set boundingBox phục vụ cho xử lí va chạm bên class Game1. Bên cạnh đó là xử lí đường di chuyển của phi thuyền địch và xử lí khi phi thuyền địch đi qua cạnh dưới của màn hình

```
        // Update
        public void Update(GameTime gameTime)
        {
            // Update Collision Rectangle
```

```
            boundingBox = new Rectangle((int)position.X, (int)position.Y, texture.Width,
texture.Height);

            // Update Enemy Movement
            position.Y += speed;

            // Move enemy to top of the screen if he fly's off buttom
            if (position.Y >= 950)
                position.Y = -75;

            EnemyShoot();
            UpdateBullets();
        }
```

- Hàm UpdateBullets() có nhiệm vụ tạo đường di chuyển của viên đạn của phi thuyền địch. Khi viên đạn chạm cạnh dưới của màn hình thì nó sẽ biến mất, lúc này nó cũng sẽ bị remove khỏi bulletList của enemy

```
        // Update bullet function
        public void UpdateBullets()
        {
            // for each bullet in our bulletList: update the movement and if the bullet
hits the top of the screen remove it from the list
            foreach (Bullet b in bulletList)
            {
                // BoundingBox for our every bullet in our bulletList
                b.boundingBox = new Rectangle((int)b.position.X, (int)b.position.Y,
b.texture.Width, b.texture.Height);

                // set movement for bullet
                b.position.Y = b.position.Y + b.speed;

                // if bullet hits the top of the screen, then make visible flase
                if (b.position.Y >= 950)
                    b.isVisible = false;
            }

            // Iterate through bulletList and see if any of the bullets are not visible,
if they arent then remove that bullet from our bullet list
            for (int i = 0; i < bulletList.Count; i++)
            {
                if (!bulletList[i].isVisible)
                {
                    bulletList.RemoveAt(i);
                    i--;
                }
            }
        }
```

- Hàm EnemyShoot() có nhiệm vụ vẽ viên đạn mới tại vị trí giữa phi thuyền. Khi số lượng đạn hiện tại của enemy nhỏ hơn 20 thì nó sẽ tiếp tục được bắn (bullet được thêm vào bulletList và được vẽ ra). Công thức "`position.X + texture.Width / 2 - newBullet.texture.Width / 2, position.Y + 30`" để lấy vị trí giữa phi thuyền địch

```
        // Enemy Shoot Function
        public void EnemyShoot()
```

```
        {
            // Shoot inly if bulletdelay resets
            if (bulletDelay >= 0)
                bulletDelay--;

            if(bulletDelay <= 0)
            {
                // Create new bullet and position it front and center of enemy ship
                Bullet newBullet = new Bullet(bulletTexture);
                newBullet.position = new Vector2(position.X + texture.Width / 2 -
newBullet.texture.Width / 2, position.Y + 30);

                newBullet.isVisible = true;

                if (bulletList.Count() < 20)
                    bulletList.Add(newBullet);
            }

            // reset bullet delay
            if (bulletDelay == 0)
                bulletDelay = 40;
        }
```

2.2.6 Class Explosion

- Class explosion có nhiệm vụ load chuỗi hình ảnh tạo vụ nổ khi xảy ra va chạm giữa đạn của người chơi với thiên thạch hoặc giữa đạn của người chơi với phi thuyền địch. Trong đó, sau khi load xong chuỗi hình ảnh thì isVisible được trả về false

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;

namespace _2DSpaceShooter
{
    class Explosion
    {
        public Texture2D texture;
        public Vector2 position;
        public float timer;
        public float interval;
        public Vector2 origin;
        public int currentFrame, spriteWidth, spriteHeight;
        public Rectangle sourceRectangle;
        public bool isVisible;

        // Constructor
        public Explosion(Texture2D newTexture, Vector2 newPosition)
        {
            position = newPosition;
            texture = newTexture;
            timer = 0f;
            interval = 15f;
```

```
            currentFrame = 1;
            spriteWidth = 64;
            spriteHeight = 64;
            isVisible = true;
        }

        // Load Content
        public void LoadContent(ContentManager Content)
        {
        }

        // Update
        public void Update(GameTime gameTime)
        {
            // Increase the timer by the number of milliseconds since update was last
called
            timer += (float)gameTime.ElapsedGameTime.TotalMilliseconds;

            // Check the timer is more than the chosen interval
            if (timer > interval)
            {
                // Show next frame
                currentFrame++;
                // Reser Timer
                timer = 0f;
            }

            // If were on the last frame, make the explosion invisible and reset
currentFrame to beginning of spritesheet
            if (currentFrame == 10)
            {
                isVisible = false;
                currentFrame = 0;
            }

            sourceRectangle = new Rectangle(currentFrame * spriteWidth, 0, spriteWidth,
spriteHeight);
            origin = new Vector2(sourceRectangle.Width / 2, sourceRectangle.Height / 2);
        }

        // Draw
        public void Draw(SpriteBatch spriteBatch)
        {
            // if visible then draw
            if(isVisible == true)
            {
                spriteBatch.Draw(texture, position, sourceRectangle, Color.White, 0f,
origin, 1.0f, SpriteEffects.None, 0);
            }
        }
    }
}
```

2.2.7 Class HUD

- Class HUD có nhiệm vụ vẽ ra khu vực ghi điểm của người chơi

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;

namespace _2DSpaceShooter
{
    public class HUD
    {
        public int playerScore, screenWidth, screenHeight;
        public SpriteFont playerScoreFont;
        public Vector2 playerScorePos;
        public bool showHud;

        // Constructor
        public HUD()
        {
            playerScore = 0;
            showHud = true;
            screenHeight = 950;
            screenWidth = 800;
            playerScoreFont = null;
            playerScorePos = new Vector2(screenWidth - 200, 7);
        }

        // Load Content
        public void LoadContent(ContentManager Content)
        {
            playerScoreFont = Content.Load<SpriteFont>("georgia");
        }

        // Update
        public void Update(GameTime gameTime)
        {
            // Get Keyboard state
            KeyboardState keyState = Keyboard.GetState();
        }

        // Draw
        public void Draw(SpriteBatch spriteBatch)
        {
            // If we are showing our HUD (if showHUD == true) then display our HUD
            if (showHud)
                spriteBatch.DrawString(playerScoreFont, "Score : " + playerScore,
playerScorePos, Color.Red);

        }
    }
}
```

2.2.8 Class Game1

- Class Game1 là class chính của chương trình, chứa các xử lí va chạm cũng như các xử lí quan trọng khác

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
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.Media;

namespace _2DSpaceShooter
{
    // Main
    public class Game1 : Microsoft.Xna.Framework.Game
    {
        // State Enum
        public enum State
        {
            Menu,
            Playing,
            Gameover
        }
        GraphicsDeviceManager graphics;
        SpriteBatch spriteBatch;
        Random random = new Random();
        public int enemyBulletDamage;
        public Texture2D menu;
        public Texture2D gameover;

        // Lists
        List<Asteroid> asteroidList = new List<Asteroid>();
        List<Enemy> enemyList = new List<Enemy>();
        List<Explosion> explosionList = new List<Explosion>();

        // Instantiating our Player and Starfield objects
        Player player = new Player();
        Starfield starfield = new Starfield();
        HUD hud = new HUD();

        // Set first State
        State gameState = State.Menu;

        // Constructor
        public Game1()
        {
            graphics = new GraphicsDeviceManager(this);
            graphics.IsFullScreen = false;
            graphics.PreferredBackBufferWidth = 800;
            graphics.PreferredBackBufferHeight = 950;
            this.Window.Title = "2D Space Shooter";
            Content.RootDirectory = "Content";
            enemyBulletDamage = 1;
            menu = null;
            gameover = null;
        }

        // Init
        protected override void Initialize()
```

```
        {
            base.Initialize();
        }

        // Load Content
        protected override void LoadContent()
        {
            // Create a new SpriteBatch, which can be used to draw textures.
            spriteBatch = new SpriteBatch(GraphicsDevice);
            hud.LoadContent(Content);
            player.LoadContent(Content);
            starfield.LoadContent(Content);
            menu = Content.Load<Texture2D>("menu");
            gameover = Content.Load<Texture2D>("gameover");
        }

        // Unload Content
        protected override void UnloadContent()
        {
        }

        // Update
        protected override void Update(GameTime gameTime)
        {
            // Allows the game to exit
            if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
                this.Exit();

            // Updating playing state
            switch (gameState)
            {
                case State.Playing:
                    {
                        // Updating Enemy's and checking collision of enemyShip to
playerShip
                        starfield.speed = 5;
                        foreach (Enemy e in enemyList)
                        {
                            // Check if enemyShip is colliding with player
                            if (e.boundingBox.Intersects(player.boundingBox))
                            {
                                player.life -= 1;
                                e.isVisible = false;
                            }

                            // Check enemy bullet collision with player ship
                            for (int i = 0; i < e.bulletList.Count; i++)
                            {
                                if
(player.boundingBox.Intersects(e.bulletList[i].boundingBox))
                                {
                                    player.life -= enemyBulletDamage;
                                    e.bulletList[i].isVisible = false;
                                }
                            }

                            // Check player bullet collision to enemy ship
                            for (int i = 0; i < player.BulletList.Count; i++)
```

```
                            {
                                if
(player.BulletList[i].boundingBox.Intersects(e.boundingBox))
                                {
                                    explosionList.Add(new
Explosion(Content.Load<Texture2D>("explosion"), new Vector2(e.position.X,
e.position.Y)));
                                    hud.playerScore += 10;
                                    player.BulletList[i].isVisible = false;
                                    e.isVisible = false;
                                }
                            }

                            e.Update(gameTime);
                        }

                        // Update Explosions
                        foreach (Explosion ex in explosionList)
                        {
                            ex.Update(gameTime);
                        }

                        // foreach asteroid in our asteroidList, update and check for
collision
                        foreach (Asteroid a in asteroidList)
                        {
                            // Check to see if any of the asteroids are colliding with
our playership, if they are.. set isVisible to False(remove them from the asteroidList)
                            if (a.boundingBox.Intersects(player.boundingBox))
                            {
                                player.life -= 1;
                                a.isVisible = false;
                            }

                            // Interate through our bulletList if any asteroids come in
contacts with these bullets, destroy bullet and asteroid
                            for (int i = 0; i < player.BulletList.Count; i++)
                            {
                                if
(a.boundingBox.Intersects(player.BulletList[i].boundingBox))
                                {
                                    explosionList.Add(new
Explosion(Content.Load<Texture2D>("explosion"), new Vector2(a.position.X,
a.position.Y)));
                                    hud.playerScore += 5;
                                    a.isVisible = false;
                                    player.BulletList.ElementAt(i).isVisible = false;
                                }
                            }

                            a.Update(gameTime);
                        }


                        //hud.Update(gameTime);

                        // If playerlife hits 0 then go to gameover state
                        if (player.life <= 0)
```

```
                            gameState = State.Gameover;

                        player.Update(gameTime);
                        starfield.Update(gameTime);
                        ManageExplosions();
                        LoadAsteroids();
                        LoadEnemies();
                        break;
                    }
                    // Updating menu state
                case State.Menu:
                    {
                        // Get Keyboard State
                        KeyboardState keyState = Keyboard.GetState();

                        if(keyState.IsKeyDown(Keys.Space))
                        {
                            gameState = State.Playing;
                        }
                        starfield.Update(gameTime);
                        starfield.speed = 1;
                        break;
                    }
                    // Updating gameover state
                case State.Gameover:
                    {
                        // Get Keyboard State
                        KeyboardState keyState = Keyboard.GetState();

                        // If in the gameover screen and user hits "Escape" key, Return
to the main menu
                        if (keyState.IsKeyDown(Keys.Escape))
                        {
                            player.position = new Vector2(400, 800);
                            enemyList.Clear();
                            asteroidList.Clear();
                            player.life = 10;
                            hud.playerScore = 0;
                            gameState = State.Menu;
                        }
                        starfield.Update(gameTime);
                        starfield.speed = 1;
                        break;
                    }
            }

            base.Update(gameTime);
        }

        // Draw
        protected override void Draw(GameTime gameTime)
        {
            GraphicsDevice.Clear(Color.Black);
            spriteBatch.Begin();

            switch(gameState)
            {
                // Drawing playing state
```

```
                case State.Playing:
                    {
                        // background
                        starfield.Draw(spriteBatch);

                        // Asteroids
                        foreach (Asteroid a in asteroidList)
                        {
                            a.Draw(spriteBatch);
                        }

                        foreach (Explosion ex in explosionList)
                        {
                            ex.Draw(spriteBatch);
                        }

                        // PlayerShip, Life
                        player.Draw(spriteBatch);

                        // EnemiesShip
                        foreach (Enemy e in enemyList)
                        {
                            e.Draw(spriteBatch);
                        }

                        // HUD: score
                        hud.Draw(spriteBatch);

                        break;
                    }

                    // Drawing menu state
                case State.Menu:
                    {
                        starfield.Draw(spriteBatch);
                        spriteBatch.Draw(menu, new Vector2(0, 0), Color.White);
                        break;
                    }

                    // Drawing gameover state
                case State.Gameover:
                    {
                        starfield.Draw(spriteBatch);
                        spriteBatch.Draw(gameover, new Vector2(0, 0), Color.White);
                        spriteBatch.DrawString(hud.playerScoreFont, "Your Final Score: "
+ hud.playerScore.ToString(), new Vector2(240, 400), Color.Red);
                        break;
                    }
            }

            spriteBatch.End();

            base.Draw(gameTime);
        }

        // Load Asteroids
        public void LoadAsteroids()
        {
```

```
            // Creating random variable for our X and Y axis of our asteroids
            int randX = random.Next(0, 750);
            int randY = random.Next(-600, -50);

            // if there are less than 5 asteroids on the screen, then create more until
there is 5 again
            if (asteroidList.Count() < 5)
            {
                asteroidList.Add(new Asteroid(Content.Load<Texture2D>("asteroids"), new
Vector2(randX, randY)));
            }

            // if any of the asteroids in the list were destroyed(or invisible), then
remove them from the list
            for (int i = 0; i < asteroidList.Count;i++)
            {
                if(!asteroidList[i].isVisible)
                {
                    asteroidList.RemoveAt(i);
                    i--;
                }
            }
        }

        // Load Enemy Function
        public void LoadEnemies()
        {
            // Creating random variable for our X and Y axis of our asteroids
            int randX = random.Next(0, 750);
            int randY = random.Next(-600, -50);

            // if there are less than 5 enemies on the screen, then create more until
there is 5 again
            if (enemyList.Count() < 5)
            {
                enemyList.Add(new Enemy(Content.Load<Texture2D>("enemy"), new
Vector2(randX, randY), Content.Load<Texture2D>("enemybullet")));
            }

            // if any of the enemies in the list were destroyed(or invisible), then
remove them from the list
            for (int i = 0; i < enemyList.Count; i++)
            {
                if (!enemyList[i].isVisible)
                {
                    enemyList.RemoveAt(i);
                    i--;
                }
            }
        }

        // Manage Explosions
        public void ManageExplosions()
        {
            for(int i = 0; i < explosionList.Count; i++)
            {
                if(!explosionList[i].isVisible)
                {
```

```
                    explosionList.RemoveAt(i);
                    i--;
                }
            }
        }
    }
}
```

- State enum dùng để phục vụ cho các xử lí tạo màn hình Menu, màn hình Playing và màn hình Gameover

```
        // State Enum
        public enum State
        {
            Menu,
            Playing,
            Gameover
        }
```

```
        GraphicsDeviceManager graphics;
        SpriteBatch spriteBatch;
        Random random = new Random();
        public int enemyBulletDamage;
        public Texture2D menu;
        public Texture2D gameover;
```

graphics: dùng để phục vụ tạo của sổ game
random: phục vụ việc random vị trí của thiên thạch và vị trí của phi thuyền địch
enemyBulletDamage: sát thương mà đạn của địch gây ra cho người chơi
menu, gameover: hai texture dùng để phục vụ việc vẽ giao diện menu và gameover

```
        // Lists
        List<Asteroid> asteroidList = new List<Asteroid>();
        List<Enemy> enemyList = new List<Enemy>();
        List<Explosion> explosionList = new List<Explosion>();
```

asteroidList: chứa list thiên thạch trên màn hình
enemyList: chứa list phi thuyền địch trên màn hình
explosionList: list explosion khi xảy ra va chạm

```
        // Instantiating our Player and Starfield objects
        Player player = new Player();
        Starfield starfield = new Starfield();
        HUD hud = new HUD();
```

player: người chơi
starfield: background
hud: hiển thị

- Constructor

```
        // Constructor
        public Game1()
```

```
        {
            graphics = new GraphicsDeviceManager(this);
            graphics.IsFullScreen = false;
            graphics.PreferredBackBufferWidth = 800;
            graphics.PreferredBackBufferHeight = 950;
            this.Window.Title = "2D Space Shooter";
            Content.RootDirectory = "Content";
            enemyBulletDamage = 1;
            menu = null;
            gameover = null;
        }
```

Tạo một cửa sổ không full screen (`graphics.IsFullScreen = false;`) với kích thước 800x950 pixel, có title: "`2D Space Shooter`"

Khởi tạo enemyBulletDamage = 1.

- Load Content

```
        // Load Content
        protected override void LoadContent()
        {
            // Create a new SpriteBatch, which can be used to draw textures.
            spriteBatch = new SpriteBatch(GraphicsDevice);
            hud.LoadContent(Content);
            player.LoadContent(Content);
            starfield.LoadContent(Content);
            menu = Content.Load<Texture2D>("menu");
            gameover = Content.Load<Texture2D>("gameover");
        }
```

Load các content của hud, player, starfield cũng như load các màn hình menu và gameover

- Update

`State.Playing:` chứa các xử lí va chạm cũng như cập nhật lại các giá trị sau mỗi lần chơi

```
            case State.Playing:
                {
                    // Updating Enemy's and checking collision of enemyShip to
playerShip
                    starfield.speed = 5;
                    foreach (Enemy e in enemyList)
                    {
                        // Check if enemyShip is colliding with player
                        if (e.boundingBox.Intersects(player.boundingBox))
                        {
                            player.life -= 1;
                            e.isVisible = false;
                        }

                        // Check enemy bullet collision with player ship
                        for (int i = 0; i < e.bulletList.Count; i++)
                        {
                            if
(player.boundingBox.Intersects(e.bulletList[i].boundingBox))
                            {
                                player.life -= enemyBulletDamage;
```

```
                                    e.bulletList[i].isVisible = false;
                                }
                            }

                            // Check player bullet collision to enemy ship
                            for (int i = 0; i < player.BulletList.Count; i++)
                            {
                                if
(player.BulletList[i].boundingBox.Intersects(e.boundingBox))
                                {
                                    explosionList.Add(new
Explosion(Content.Load<Texture2D>("explosion"), new Vector2(e.position.X,
e.position.Y)));
                                    hud.playerScore += 10;
                                    player.BulletList[i].isVisible = false;
                                    e.isVisible = false;
                                }
                            }

                            e.Update(gameTime);
                        }

                        // Update Explosions
                        foreach (Explosion ex in explosionList)
                        {
                            ex.Update(gameTime);
                        }

                        // foreach asteroid in our asteroidList, update and check for
collision
                        foreach (Asteroid a in asteroidList)
                        {
                            // Check to see if any of the asteroids are colliding with
our playership, if they are.. set isVisible to False(remove them from the asteroidList)
                            if (a.boundingBox.Intersects(player.boundingBox))
                            {
                                player.life -= 1;
                                a.isVisible = false;
                            }

                            // Interate through our bulletList if any asteroids come in
contacts with these bullets, destroy bullet and asteroid
                            for (int i = 0; i < player.BulletList.Count; i++)
                            {
                                if
(a.boundingBox.Intersects(player.BulletList[i].boundingBox))
                                {
                                    explosionList.Add(new
Explosion(Content.Load<Texture2D>("explosion"), new Vector2(a.position.X,
a.position.Y)));
                                    hud.playerScore += 5;
                                    a.isVisible = false;
                                    player.BulletList.ElementAt(i).isVisible = false;
                                }
                            }

                            a.Update(gameTime);
                        }
```

```
                    //hud.Update(gameTime);

                    // If playerlife hits 0 then go to gameover state
                    if (player.life <= 0)
                        gameState = State.Gameover;

                    player.Update(gameTime);
                    starfield.Update(gameTime);
                    ManageExplosions();
                    LoadAsteroids();
                    LoadEnemies();
                    break;
                }
```

```
        // Check if enemyShip is colliding with player
        if (e.boundingBox.Intersects(player.boundingBox))
        {
             player.life -= 1;
             e.isVisible = false;
        }
```

Khi phi thuyền địch va chạm với người chơi thì mạng của người chơi sẽ bị trừ đi 1 và phi thuyền địch sẽ biến mất

```
        // Check enemy bullet collision with player ship
        for (int i = 0; i < e.bulletList.Count; i++)
        {
               if (player.boundingBox.Intersects(e.bulletList[i].boundingBox))
               {
                      player.life -= enemyBulletDamage;
                      e.bulletList[i].isVisible = false;
               }
         }
```

Khi đạn của phi thuyền địch va chạm với người chơi thì mạng của người chơi bị trừ đi 1 và viên đạn đó sẽ biến mất

```
        // Check player bullet collision to enemy ship
        for (int i = 0; i < player.BulletList.Count; i++)
        {
               if (player.BulletList[i].boundingBox.Intersects(e.boundingBox))
               {
                     explosionList.Add(new
Explosion(Content.Load<Texture2D>("explosion"), new Vector2(e.position.X,
e.position.Y)));
                      hud.playerScore += 10;
                      player.BulletList[i].isVisible = false;
                      e.isVisible = false;
               }
        }
```

Khi một viên đạn của người chơi va chạm với phi thuyền địch thì một explosion được hình thành, viên đạn đó bị mất đi và phi thuyền đó cũng biến mất. Điểm của người chơi được cộng thêm 10.

```
        // Check to see if any of the asteroids are colliding with our playership
        if (a.boundingBox.Intersects(player.boundingBox))
        {
                player.life -= 1;
                a.isVisible = false;
        }
```

Khi thiên thạch va chạm với người chơi thì mạng của người chơi bị trừ đi 1 và thiên thạch đó bị biến mất

```
        // Check player bullet collision to asteroid
        for (int i = 0; i < player.BulletList.Count; i++)
        {
                if (a.boundingBox.Intersects(player.BulletList[i].boundingBox))
                {
                        explosionList.Add(new
Explosion(Content.Load<Texture2D>("explosion"), new Vector2(a.position.X,
a.position.Y)));
                        hud.playerScore += 5;
                        a.isVisible = false;
                        player.BulletList.ElementAt(i).isVisible = false;
                 }
        }
```

Khi đạn của người chơi va chạm với thiên thạch thì một explosion được hình thành, viên đạn đó sẽ mất đi và thiên thạch cũng biến mất. Điểm của người chơi cộng thêm 5.

```
        // Manage Explosions
        public void ManageExplosions()
        {
            for(int i = 0; i < explosionList.Count; i++)
            {
                if(!explosionList[i].isVisible)
                {
                    explosionList.RemoveAt(i);
                    i--;
                }
            }
           }
```

Sau khi vụ nổ được load xong thì nó sẽ được trả về không (bên class explosion). Hàm này dùng để remove explosion khỏi explosionList sau khi đã load xong chuỗi hình ảnh.

```
        // Load Asteroids
        public void LoadAsteroids()
        {
            // Creating random variable for our X and Y axis of our asteroids
            int randX = random.Next(0, 750);
            int randY = random.Next(-600, -50);
```

```
            // if there are less than 5 asteroids on the screen, then create more until
there is 5 again
            if (asteroidList.Count() < 5)
            {
                asteroidList.Add(new Asteroid(Content.Load<Texture2D>("asteroids"), new
Vector2(randX, randY)));
            }

            // if any of the asteroids in the list were destroyed(or invisible), then
remove them from the list
            for (int i = 0; i < asteroidList.Count;i++)
            {
                if(!asteroidList[i].isVisible)
                {
                    asteroidList.RemoveAt(i);
                    i--;
                }
            }
        }
```

Mỗi lần sẽ có 5 thiên thạch được xuất hiện. Vị trí của các thiên thạch sẽ được random với hoành độ từ 0 đến 750, tung độ từ -600 đến -50. Trong lúc di chuyển nếu có va chạm thì thiên thạch sẽ biến mất và bị remove khỏi asteroidList

```
        // Load Enemy Function
        public void LoadEnemies()
        {
            // Creating random variable for our X and Y axis of our asteroids
            int randX = random.Next(0, 750);
            int randY = random.Next(-600, -50);

            // if there are less than 5 enemies on the screen, then create more until
there is 5 again
            if (enemyList.Count() < 5)
            {
                enemyList.Add(new Enemy(Content.Load<Texture2D>("enemy"), new
Vector2(randX, randY), Content.Load<Texture2D>("enemybullet")));
            }

            // if any of the enemies in the list were destroyed(or invisible), then
remove them from the list
            for (int i = 0; i < enemyList.Count; i++)
            {
                if (!enemyList[i].isVisible)
                {
                    enemyList.RemoveAt(i);
                    i--;
                }
            }
        }
```

Mỗi lần sẽ có 5 phi thuyền địch xuất hiện. Vị trí của nó được random với hoành độ từ 0 đến 750, tung độ từ -600 đến -50. Trong lúc di chuyển nếu có va chạm xảy ra thì nó sẽ bị remove khỏi enemyList

```
        State.Menu:
```

```
        // Updating menu state
        case State.Menu:
        {
                // Get Keyboard State
                KeyboardState keyState = Keyboard.GetState();

                if(keyState.IsKeyDown(Keys.Space))
                {
                        gameState = State.Playing;
                }
                starfield.Update(gameTime);
                starfield.speed = 1;
                break;
           }
```

Cài đặt trạng thái bàn phím. Nếu nút Space được nhấn thì sẽ chuyển đến `State.Playing`. Tiếp theo là Update starfield theo thời gian

```
// Updating gameover state
case State.Gameover:
{
        // Get Keyboard State
        KeyboardState keyState = Keyboard.GetState();

        // If in the gameover screen and user hits "Escape" key, Return to the main menu
        if (keyState.IsKeyDown(Keys.Escape))
        {
                player.position = new Vector2(400, 800);
                enemyList.Clear();
                asteroidList.Clear();
                player.life = 10;
                hud.playerScore = 0;
                gameState = State.Menu;
         }
         starfield.Update(gameTime);
         starfield.speed = 1;
         break;
}
```

Cài đặt trạng thái bàn phím. Nếu nút Escape được nhấn thì chuyển sang `State.Menu` trả vị trí người chơi về vị trí (400, 800), mạng được trả về 10, điểm về 0 và enemyList và asteroidList được clear. Tiếp theo là update starfield theo thời gian.

- Draw

```
// Draw
protected override void Draw(GameTime gameTime)
{
        GraphicsDevice.Clear(Color.Black);
        spriteBatch.Begin();

        switch(gameState)
        {
                // Drawing playing state
                case State.Playing:
                {
```

```
                        // background
                        starfield.Draw(spriteBatch);

                        // Asteroids
                        foreach (Asteroid a in asteroidList)
                        {
                              a.Draw(spriteBatch);
                        }

                        foreach (Explosion ex in explosionList)
                        {
                              ex.Draw(spriteBatch);
                        }

                        // PlayerShip, Life
                        player.Draw(spriteBatch);

                        // EnemiesShip
                        foreach (Enemy e in enemyList)
                        {
                              e.Draw(spriteBatch);
                        }

                        // HUD: score
                        hud.Draw(spriteBatch);

                        break;
                  }

                  // Drawing menu state
                  case State.Menu:
                  {
                        starfield.Draw(spriteBatch);
                        spriteBatch.Draw(menu, new Vector2(0, 0), Color.White);
                        break;
                  }

                  // Drawing gameover state
                  case State.Gameover:
                   {
                              starfield.Draw(spriteBatch);
                              spriteBatch.Draw(gameover, new Vector2(0, 0), Color.White);
                              spriteBatch.DrawString(hud.playerScoreFont, "Your Final Score: "
+ hud.playerScore.ToString(), new Vector2(240, 400), Color.Red);
                              break;
                   }
            }

            spriteBatch.End();

            base.Draw(gameTime);
}
```

Trong `State.Playing` thì background, asteroids, explosion, player, enemies, score sẽ được vẽ lên theo thức tự background, asteroids, explosion, player, enemies, score.

Trong `State.Gameover` thì score được đưa ra dưới dạng string.

3 CÀI ĐẶT, CÁCH CHƠI

3.1 Cài đặt

- Nhấn vào file 2DSpaceShooter.exe để cài đặt và chơi game

3.2 Cách chơi:

- Dùng các phím W, S, A, D để di chuyển lên, xuống, trái, phải
- Dùng phím Space để bắn
- Trong màn hình Gameover, nhấn phím Escape để trở về màn hình chính