# TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHỆ TPHCM

## KHOA CÔNG NGHỆ THÔNG TIN

***★***


# ĐỀ TÀI MÔN HỌC LẬP TRÌNH CHO THIẾT BỊ DI ĐỘNG

## Đề tài:

# WINDOW PHONE 7 – GAME NKO

**GVHD:** *Gv.Ths. Nguyễn Hà Giang*

Thành viên:


- Lê Hoàng Khiêm - 1091021073
- Lê Thuý Oanh - 1091021124
- Lê Hoàng Nguyên – 1091021117


**Lớp: 10HTHH2**

# Mục Lục

# I. Giới thiệu windows phone 7

## 1. Tổng quan:

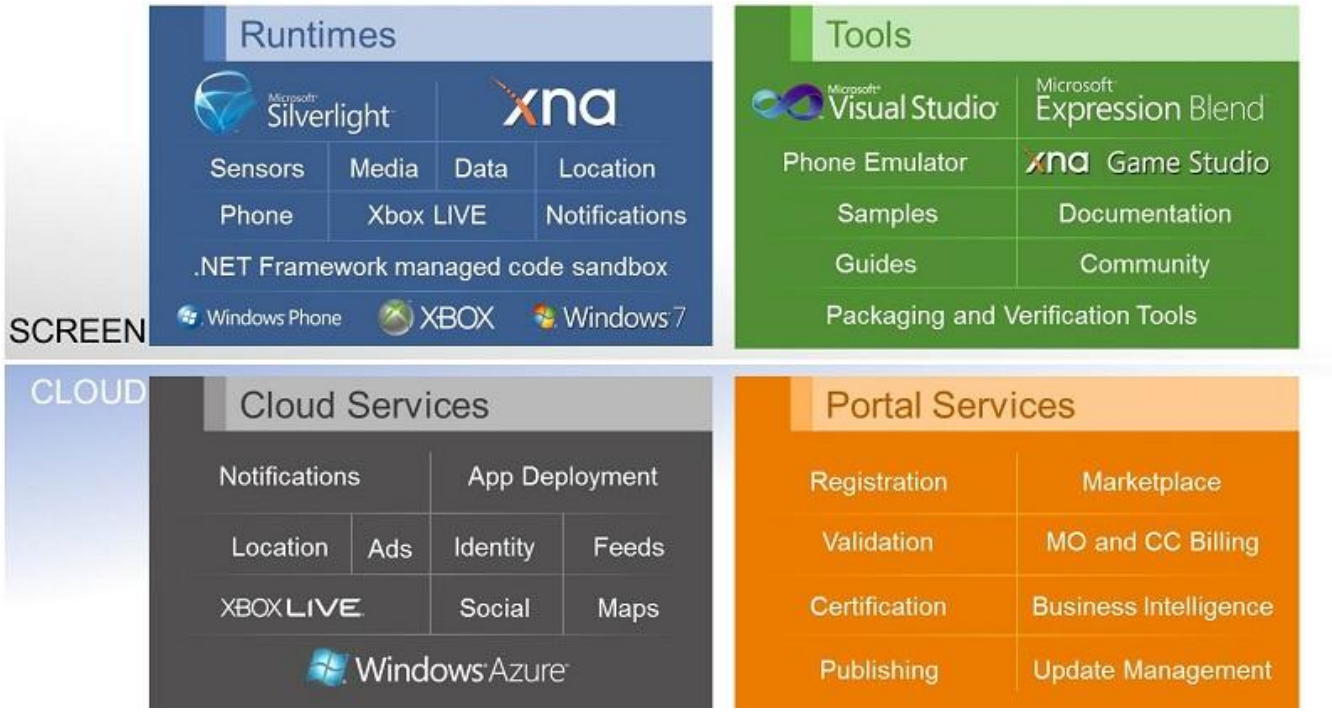
- Windows Phone 7(WP7) ra đời không lâu nhưng hệ điều hành này nhanh chóng được quan tâm bởi những ưu điểm và tính năng mà các hệ điều hành khác chưa có. Sau khi người dùng trải nghiệm và nhận xét rằng WP7 không hề thua kém Android hay iOS.
- Tuy nhiên, có thể nhận thấy rằng, các ứng dụng và game trên WP7 hiện nay chưa nhiều, hầu hết tính phí … và đặc biệt vẫn chưa có nhiều ứng dụng Việt

## 2. Giới thiệu nền tảng Windows Phone 7 (Windows Phone 7 platform)

- Ứng dụng nền tảng của Windows Phone cho phép nhà phát triển tạo ra những ứng dụng hấp dẫn cho người dùng. Những ứng dụng này được xây dựng dựa trên những công cụ và công nghệ hiện tại của Microsoft như Visual Studio, Expression Blend, Silverlight, và XNA Framework(khung xây dựng ứng dụng game)
- Nền tảng Windows Phone cung cấp 2 khung (framework) để phát triển ứng dụng:
    - Silverlight hướng sự kiện, phát triển dựa trên ứng dụng XAML cho phép nhà phát triển xây dựng một cách sáng tạo dựa trên những trải nghiệm của người dùng.
    - Khung XNA cho phép xây dựng các ứng dụng game thú vị.
- Nền tảng WP giúp các nhà phát triển tạo ra các ứng dụng bằng cách cung cấp:
    - Một bộ công cụ quen thuộc và ít tốn kém.
    - Một gắn kết và thiết kế tập API
    - Một sandbox độc lập cho mỗi ứng dụng
    - Thời gian chạy dịch vụ trên thiết bị có thể truy cập vào các dịch vụ web trong đám mây, ví dụ như Xbox LIVE, Windows azure, dịch vụ định vị, dịch vụ thông báo.
    - Windows Phone MarketPlace sẽ phân phối những ứng dụng này.
- Nền tảng WP, giống hầu hết những nền tảng khác, sẽ tiếp tục phát triển theo thời gian nhưng mục tiêu sẽ luôn được hướng đến là:
    - Ứng dụng phong phú hơn, phát triển với tiêu chuẩn cao.
    - Hỗ trợ cho người dùng cá nhân và kết nối dữ liệu trên nhiều thiết bị.
    - Một cổng thông tin manh mẽ và hoàn thiện hệ thống quản lý xoay vòng của MarketPlace.

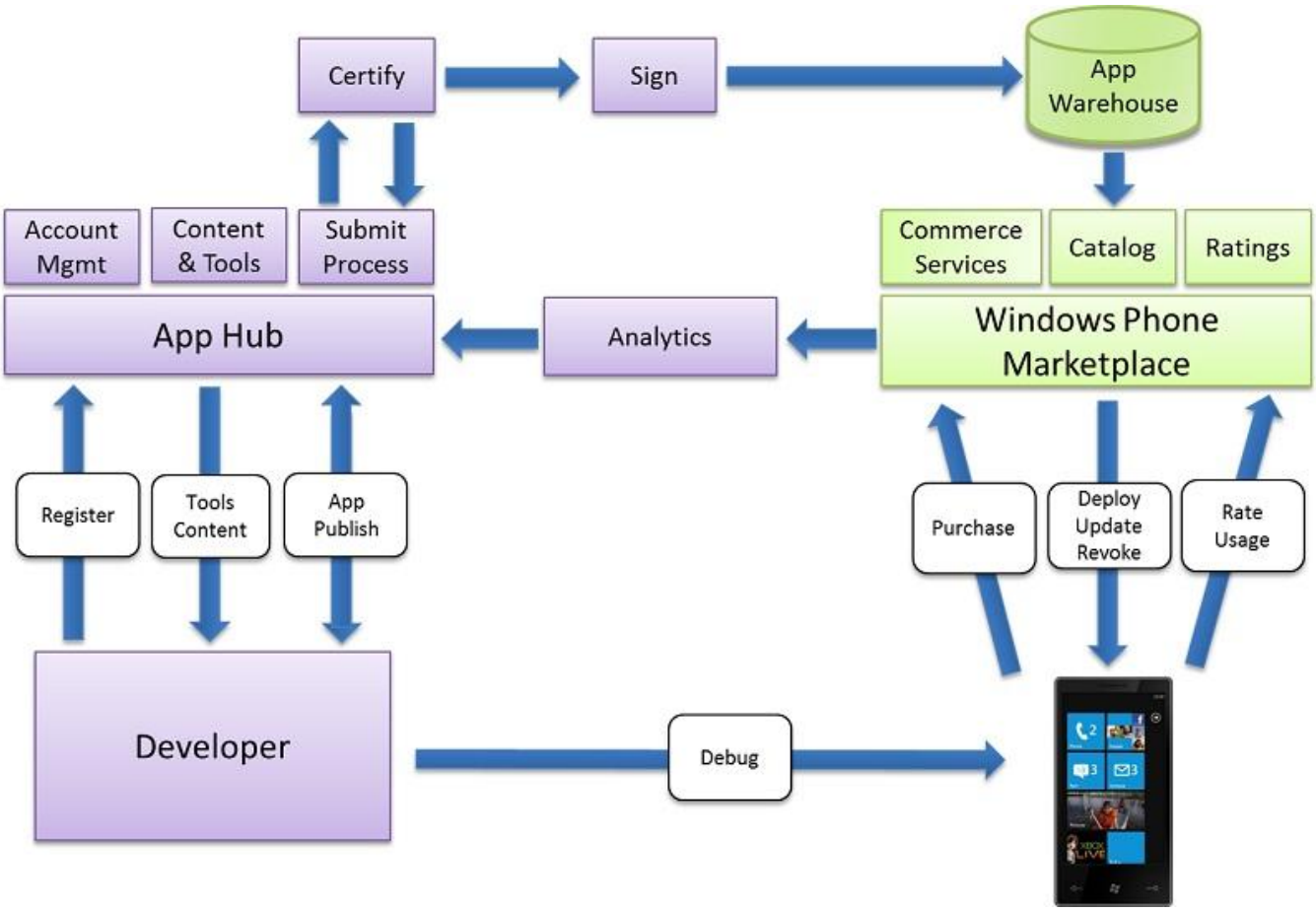
### 3. Cấu trúc nền tảng Windows Phone:

Có 4 thành phần chính:




- Runtimes: Silverlight và khung XNA, cùng với nhựng tính năng Windows phone cụ thể, kết hợp để cung cấp một môi trường trưởng thành hơn trong việc xây dựng an toàn và các ứng dụng đồ họa phong phú.
- Tools: Visual Studio, Expression Blend và các công cụ, tài liệu giúp tạo ra ứng dụng một cách nhanh chóng cũng như sửa lỗi, triển khai và cập nhật ứng dụng.
- Cloud Service(dịch vụ điện toán đám mây): Windows Azure, Xbox LIVE, dịch vụ thông báo, dịch vụ định vị cùng các loại dịch vụ web khác, cho phép nhà phát triển chia sẽ dữ liệu qua các đám mây và cung cấp bất cứ điều gì mà người dùng mong muốn khi sử dung dịch vụ. Kết nối với các dịch vụ web của bên thứ ba cũng được hỗ trộ đầy đủ.
- Portal services: Windows Phone Marketplace cung cấp các dịch vụ mạnh mẽ cho phép nhà phát triển đăng ký, xác nhận và quảng bá ứng dụng của họ.

## 4. Ứng dụng phát triển ứng dụng vòng tròn(LifeCycle)




Phần này mô tả làm thế nào các nhà phát triển có thể sử dụng ứng dụng nền tảng WP từ đến cuối của chu kỳ phát triển:

- App Hub(trung tâm ứng dụng) sẽ là điểm khởi đầu cho các developer(nhà phát triển). Các nhà phát triển có thể bắt đầu bằng cách đăng ký một ID Windows LIVE. Tiếp theo, họ có thế đăng ký để có được Windows Phone DSK và thông tin bản quyền liên quan để phát triển các ứng dụng bằng cách sử dụng Visual Studio và Expression Blend. App Hub là nơi duy nhất chứa tất cả công cụ để xây dựng ứng dụng Windows Phone. Nhà phát triển cũng có thể đăng ký một hoặc nhiều điện thoại để sử dụng để  kiểm nghiệm ứng dụng đang phát triển. App Hub chứa mẫu, tài liệu và cộng đồng những nhà phát triển giúp cho những nhà phát triển mới có thể xây dựng ứng dụng Windown Phone thành công.

## 5. Giới thiệu tính năng mới trong Windows Phone 7

- Sử dụng cảm biến gia tốc, điều hướng v.v…
- Có thể thực hiện đa nhiệm (vừa nghe nhạc, vừa lướt web, nhắn tin…)
- Các ứng dụng có thể giao tiếp bằng giao thức TCP, UDP
- Có thể viết code để truy cập vào camera
- Hỗ trợ viết code bằng ngôn ngữ VB, C#
- IE9 được đưa vào… có hỗ trợ HTML5
- Có thể sử dụng LINQ để lập trình

# II. Project game NKO

a. ***Giới thiệu:*** game NKO là một ứng dụng game được xây dựng trên nền XNA sử dụng bộ công cụ Visual studio 2010. Đây là một game đồ họa 2D mang phong cách cổ điển. Bước đầu game được xây dựng với 2 màn, 3 mức độ dễ đến khó, có tính điểm, đồ họa khá…

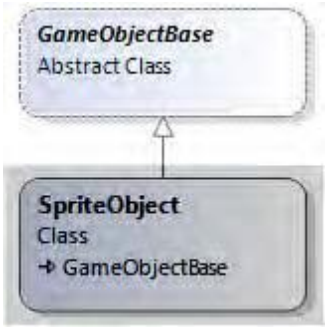
b. ***Nội dung (kich bản) game:*** người chơi bắt đầu game với việc chọn mức độ game với cảnh đầu tiên! Sau khi bắn hạ hết mục tiêu sẽ qua cảnh 2 và kết thúc nếu vượt qua hết các mục tiêu.

# III. Xây dựng một ứng dụng game:

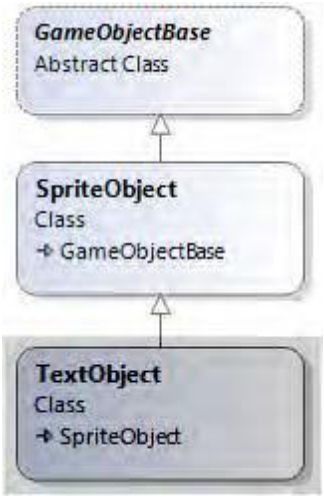
## 1. Game FrameWork (khung game)

***Giới thiệu:*** Game Framework là một cách thức giúp cho việc khởi tạo trò chơi trở nên dễ dàng, linh hoạt. Nhưng nó sẽ không cung cấp cơ chế xây dựng quản lý các đối tượng game bên trong trò chơi.

- Lớp GameObjectBase(đối tượng cơ sở): là một lớp trừu tượng và là nơi bắt đầu của bất cứ đối tượng game.
- Lớp Sprite: kế thừa từ lớp GameObjectBase là lớp SpriteObject. Đây là một lớp (không trừu tượng) trong đó chúng ta sẽ thêm tất cả các chức năng cơ bản của các đối tượng game. Các chức năng cơ bản như vị trí đối tượng game(sprite), scaling(độ lớn), rotate(hướng), origin(nguồn)…

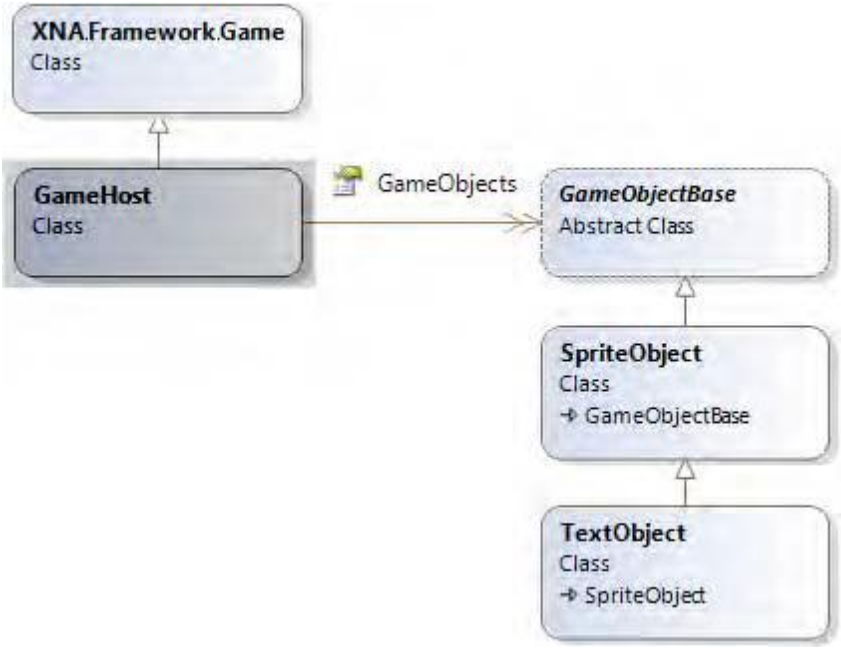



- Lớp TextObject(văn bản đối tượng): giống như lớp SpriteObject, lớp TextObject cho chúng ta một cách dễ dàng để sử dụng cơ chế đại diện cho tất cả các thuộc tính có thể có của một đoạn văn bản được hiển thị trong một game




- Game Host: Lớp này nắm giữ các đối tượng khác nhau mà chúng ta sẽ muốn sử dụng trong game, đối tượng Dictionary đặc biệt có chứa các kết cấu và phông chữ, và một danh sách các đối tượng game thực tế. Các đối tượng game được lưu trữ trong một danh sách chứa các đối tượng của loại GameObjectBase, cho phép chúng ta lưu trữ trong các SpriteObject và các đối tượng có nguồn gốc từ TextObject.

  Lớp cũng có chứa một số phương pháp đơn giản mà chúng ta sẽ tiết kiệm hơn khi viết các chức năng trong lớp Game chính




- Lớp GameHelper(trợ giúp): Lớp này tạo ra các phương thức ngẫu nhiên khi khởi tạo

một đối tượng game. Ví dụ tạo ra ngẫu nhiên vị trí đối tượng, ngẫu nhiên kích thước đối tượng, hay số lượng đối tượng game…

## 2. Các lớp đối tượng game (Một phần hình ảnh code)

### a. ***Lớp clsGame(lớp chính)***

Khởi tạo các đối tương game, thời gian, xử lý các thao tác của các đối tương game, hình nền game, các sự kiện chuyển màn, tính điểm, update, Draw… của nhân vật game

```
public class clsGame : KhungTroChoi.GameHost
{

    public static int diemso = 0, MaxBlood = 5, MaxAlive = 1, map=1, kichhoat=1, num_ufo = 0, level=1;
    private GraphicsDeviceManager _graphics;
    private SpriteBatch _spriteBatch;
    private SpriteFont _spiteFont;
    private MayBay _playerShip;
    int batdau = 0;

    enum BState
    {
        HOVER,
        UP,
        JUST_RELEASED,
        DOWN
    }
    const int NUMBER_OF_BUTTONS = 13,
        STARTGAME_BUTTON_INDEX =0,
        EASY_BUTTON_INDEX = 1,
        MEDIUM_BUTTON_INDEX = 2,
        HARD_BUTTON_INDEX = 3,
        RESETGAME_BUTTON_INDEX=4,
        BACKGROUND=5,
        NEXT_MAP_BUTTON_INDEX =6,
        BACKGROUND_MAP2 = 7,
        NEWGAME_BUTTON_INDEX=8,
        ADVAND_BUTTON_INDEX =9,
        OPTION_BUTTON_INDEX=10,
        EXIT_BUTTON_INDEX=11,
        HEAD_INDEX=12,

        BUTTON_HEIGHT = 45,
        BUTTON_WIDTH = 150;
    Color background_color;
    Color[] button_color = new Color[NUMBER_OF_BUTTONS];
    Rectangle[] button_rectangle = new Rectangle[NUMBER_OF_BUTTONS];
    BState[] button_state = new BState[NUMBER_OF_BUTTONS];
```

```
protected override void Draw(GameTime gameTime)
{

    if (batdau == 0)
    {
        _spriteBatch.Begin();

        _spriteBatch.Draw(button_texture[5], button_rectangle[5], button_color[5]);

        _spriteBatch.Draw(button_texture[0], button_rectangle[0], button_color[0]);
        _spriteBatch.Draw(button_texture[8], button_rectangle[8], button_color[8]);
        _spriteBatch.Draw(button_texture[9], button_rectangle[9], button_color[9]);
        _spriteBatch.Draw(button_texture[10], button_rectangle[10], button_color[10]);
        _spriteBatch.Draw(button_texture[11], button_rectangle[11], button_color[11]);
        _spriteBatch.Draw(button_texture[12], button_rectangle[12], button_color[12]);

        _spriteBatch.End();
    }
    if (batdau == 1)
    {
        _spriteBatch.Begin();

        _spriteBatch.Draw(button_texture[5], button_rectangle[5], button_color[5]);
        _spriteBatch.End();

        _spriteBatch.Begin();
        for (int i = 1; i < 4; i++)
            _spriteBatch.Draw(button_texture[i], button_rectangle[i], button_color[i]);
        _spriteBatch.End();
    }
    if(batdau==2 && map==1)
    {
        GraphicsDevice.Clear(Color.Black);
        _spriteBatch.Begin();
        DrawSprites(gameTime, _spriteBatch, Textures["Star"]);
        DrawSprites(gameTime, _spriteBatch, Textures["Rock1"]);
        DrawSprites(gameTime, _spriteBatch, Textures["Rock2"]);
        DrawSprites(gameTime, _spriteBatch, Textures["Rock3"]);
        DrawSprites(gameTime, _spriteBatch, Textures["maybay"]);
```

```
            DrawSprites(gameTime, _spriteBatch, Textures["Bullet"]);
            _spriteBatch.DrawString(_spiteFont, "SCORE: " + diemso, new Vector2(0, 0), Color.White);
            _spriteBatch.DrawString(_spiteFont, "LEVEL: " + level, new Vector2(0, 20), Color.White);
            _spriteBatch.DrawString(_spiteFont, "BLOOD: " + MaxBlood, new Vector2(380, 0), Color.Red);
            _spriteBatch.DrawString(_spiteFont, "Alive: " + MaxAlive, new Vector2(380, 20), Color.Red);
            DrawText(gameTime, _spriteBatch);
            _spriteBatch.End();
            if (MaxAlive == 0)
            {
                _playerShip.ScaleX = 0.0f;
                _playerShip.ScaleY = 0.0f;
                _spriteBatch = new SpriteBatch(GraphicsDevice);
                button_texture[RESETGAME_BUTTON_INDEX] =
                  Content.Load<Texture2D>(@"ResetGame");
                _spriteBatch.Begin();
                _spriteBatch.Draw(button_texture[4], button_rectangle[4], button_color[4]);
                update_buttons_option(4);
                _spriteBatch.End();
            }

           _spriteBatch.Begin(SpriteSortMode.Immediate, BlendState.Additive);

            DrawSprites(gameTime, _spriteBatch, Textures["SmokeParticle"]);

            _spriteBatch.End();
        }
        if (batdau == 2 && map == 2)
        {
            _spriteBatch.Begin();

            _spriteBatch.Draw(button_texture[7], button_rectangle[7], button_color[7]);
            _spriteBatch.End();

            _spriteBatch.Begin();

            DrawSprites(gameTime, _spriteBatch, Textures["Star"]);
            DrawSprites(gameTime, _spriteBatch, Textures["bachtuoc1"]);
            DrawSprites(gameTime, _spriteBatch, Textures["bachtuoc2"]);
            DrawSprites(gameTime, _spriteBatch, Textures["bachtuoc3"]);
```

```
            DrawSprites(gameTime, _spriteBatch, Textures["taungam"]);
            DrawSprites(gameTime, _spriteBatch, Textures["Bullet"]);
            _spriteBatch.DrawString(_spiteFont, "SCORE: " + diemso, new Vector2(0, 0), Color.White);
            _spriteBatch.DrawString(_spiteFont, "LEVEL: " + level, new Vector2(0, 20), Color.White);
            _spriteBatch.DrawString(_spiteFont, "BLOOD: " + MaxBlood, new Vector2(380, 0), Color.Red);
            _spriteBatch.DrawString(_spiteFont, "Alive: " + MaxAlive, new Vector2(380, 20), Color.Red);
            DrawText(gameTime, _spriteBatch);
            _spriteBatch.End();

            if (MaxAlive == 0)
            {

                _playerShip.ScaleX = 0.0f;
                _playerShip.ScaleY = 0.0f;
                _spriteBatch = new SpriteBatch(GraphicsDevice);
                button_texture[RESETGAME_BUTTON_INDEX] =
                  Content.Load<Texture2D>(@"ResetGame");
                _spriteBatch.Begin();
                _spriteBatch.Draw(button_texture[4], button_rectangle[4], button_color[4]);
                update_buttons_option(4);
                _spriteBatch.End();
            }
            else
                if (map == 2 && kichhoat == 1 && num_ufo==0)
                {

                    _spriteBatch = new SpriteBatch(GraphicsDevice);
                    button_texture[NEXT_MAP_BUTTON_INDEX] =
                      Content.Load<Texture2D>(@"NextMap");
                    _spriteBatch.Begin();
                    _spriteBatch.Draw(button_texture[6], button_rectangle[6], button_color[6]);
                    update_buttons_option(6);
                    _spriteBatch.End();
                }


            _spriteBatch.Begin(SpriteSortMode.Immediate, BlendState.Additive);
```

### b. ***Lớp đối tượng UFO:***

Khởi tạo:

```
internal class UFO : KhungTroChoi.DoiTuongSprite
{

    private clsGame _game;


    private float _constructorSpeed;

    //tốc độ di chuyển
    private float _moveSpeed;

    //hướng di chuyển
    private Vector2 _direction;

    //tốc độ xoay
    private float _rotateSpeed;

    //Đối tượng sẽ chia nhỏ bao nhiêu lần
    private int _generation;

    //-----------------------------------------------------------------------------------
    // Class constructors

    internal UFO(clsGame game, Texture2D texture, int generation, float size, float speed)
        : base(game, Vector2.Zero, texture)
    {
        _game = game;

        // vị trí khởi tạo random
        PositionX = TroGiupGame.RandomNext(0, _game.Window.ClientBounds.Width);
        PositionY = TroGiupGame.RandomNext(0, 0);

        // thiết lập gốc
        Origin = new Vector2(texture.Width, texture.Height) / 2;

        // tốc độ và số lần chia nhỏ
        _constructorSpeed = speed;
```

Phương thức di chuyển

```
base.Update(gameTime);

// cập nhật vi trí UFO
Position += _direction * _moveSpeed;

// vượt qua cạnh của màn hình, thiết lập lại phía cạnh bên kia
if (BoundingBox.Bottom < _game.Window.ClientBounds.Top && _direction.Y < 0)
{
    PositionY = _game.Window.ClientBounds.Height + SpriteTexture.Height;
}
if (BoundingBox.Top > _game.Window.ClientBounds.Bottom && _direction.Y > 0)
{
    PositionY = -SpriteTexture.Height;
}
if (BoundingBox.Right < _game.Window.ClientBounds.Left && _direction.X < 0)
{
    PositionX = _game.Window.ClientBounds.Width + SpriteTexture.Width;
}
if (BoundingBox.Left > _game.Window.ClientBounds.Right && _direction.X > 0)
{
    PositionX = -SpriteTexture.Width;
}

// góc xoay
Angle += MathHelper.ToRadians(_rotateSpeed);
```

Phương thức bị đụng độ, bị phá hủy

```
internal void DamageRock()
{
    UFO newRock;
    HatVatChat[] particleObjects;
    HatVatChat particleObj;
    //tạo các đám khói khi bị phá hủy hoặc chia nhỏ
    particleObjects = _game.GetParticleObjects(5);

    for (int i = 0; i < particleObjects.Length; i++)
    {
        particleObj = particleObjects[i];
        particleObj.ResetProperties(Position, _game.Textures["SmokeParticle"]);
        switch (TroGiupGame.RandomNext(3))
        {
            case 0: particleObj.SpriteColor = Color.DarkGray; break;
            case 1: particleObj.SpriteColor = Color.LightGray; break;
            default: particleObj.SpriteColor = Color.White; break;
        }
        particleObj.Scale = new Vector2(0.4f, 0.4f);
        particleObj.IsActive = true;
        particleObj.Intensity = 255;
        particleObj.IntensityFadeAmount = 3 + TroGiupGame.RandomNext(1.5f);
        particleObj.Speed = TroGiupGame.RandomNext(3.0f);
        particleObj.Inertia = 0.9f + TroGiupGame.RandomNext(0.015f);
    }


    // khi UFO bị chia đến nhỏ nhất
    // UFO bị phá hủy
    if (_generation == 0)
    {
        //Xóa UFO khỏi game
        _game.GameObjects.Remove(this);
    }
    else
    {
        // giảm kích thước của UFO
        InitializeRock(ScaleX * 0.7f);
        // cập nhật số lần bị chia nhỏ
        _generation -= 1;
```

Phương thức xoay

```
private void InitializeRock(float size)
{
    // kích thước
    Scale = new Vector2(size, size);

    // tốc độ di chuyển random
    _moveSpeed = TroGiupGame.RandomNext(_constructorSpeed) + _constructorSpeed;

    // tốc độ xoay random
    if(clsGame.map==2)
        _rotateSpeed = TroGiupGame.RandomNext(0.0f, 0.0f);
    else
        _rotateSpeed = TroGiupGame.RandomNext(-5.0f, 5.0f);

    // hướng di chuyển
    do
    {
        _direction = new Vector2(TroGiupGame.RandomNext(-1.0f, 1.0f), TroGiupGame.RandomNext(-1.0f, 1.0f));
    } while (_direction == Vector2.Zero);
    // các vecto chuyển động với 1 đơn vị độ dài
    _direction.Normalize();
}
```

…

### c. ***Lớp đối tượng MayBay***

Phương thức di chuyển

```
// di chuyển máy bay
Position += _velocity;

// Nếu vượt qua các cạnh của cửa sổ, thiết lập lại ở phía đối diện
if (BoundingBox.Bottom < _game.Window.ClientBounds.Top && _velocity.Y < 0)
{
    PositionY = _game.Window.ClientBounds.Height + SpriteTexture.Height;
}
if (BoundingBox.Top > _game.Window.ClientBounds.Bottom && _velocity.Y > 0)
{
    PositionY = -SpriteTexture.Height;
}
if (BoundingBox.Right < _game.Window.ClientBounds.Left && _velocity.X < 0)
{
    PositionX = _game.Window.ClientBounds.Width + SpriteTexture.Width;
}
if (BoundingBox.Left > _game.Window.ClientBounds.Right && _velocity.X > 0)
{
    PositionX = -SpriteTexture.Width;
}

// Xoay máy bay về phía mục tiêu
if (Angle != _targetAngle)
{
    Angle += (_targetAngle - Angle) * 0.2f;
}
```

Phương thức kiểm tra va chạm và phá hủy

```
DoiTuongSprite collisionObj;

//Không còn mạng sống
if (clsGame.MaxAlive == 0)
{
    return;
}
DoiTuongCoSo uf;
clsGame.num_ufo = 0;
for (int i = _game.GameObjects.Count - 1; i >= 0; i--)
{
    uf = _game.GameObjects[i];
    if (uf is UFO)
        clsGame.num_ufo++;
}


if (clsGame.num_ufo == 0)
{
    clsGame.kichhoat = 1;
    clsGame.map = 2;
    return;
}

// máy bay còn tồn tại chưa nổ?
if (IsExploding)
{
    // đang bị nổ, giảm lần cập nhật
    ExplosionUpdateCount -= 1;

    if (ExplosionUpdateCount == 0)
    {
        //thiết lập lại máy bay
        Position = new Vector2(_game.Window.ClientBounds.Width, _game.Window.ClientBounds.Height) / 2;
        Angle = 0;
        _targetAngle = 0;
        _shieldTime = MaxShieldTime;
```

Phương thức xoay

```
// tính toán góc cần xoay
private void RotateToFacePoint(Vector2 point)
{
    // Tìm góc giữa máy bay và điểm chỉ.
    // đầu tiên tìm khoảng cách từ máy bay đến điểm chỉ.
    point -= Position;

    //nếu chỉ đúng vào ngay máy bay thì không làm gì!
    if (point == Vector2.Zero) return;

    // chắc chắn góc trong khoảng 0 đến 360 độ
    while (Angle < 0) { Angle += MathHelper.TwoPi; }
    while (Angle > MathHelper.TwoPi) { Angle -= MathHelper.TwoPi; }
    // lấy góc hiện tại
    float angleDegrees;
    angleDegrees = MathHelper.ToDegrees(Angle);

    // Tính toán góc giữa con tàu và điểm chạm, chuyển sang độ
    float targetAngleDegrees;
    targetAngleDegrees = MathHelper.ToDegrees((float)Math.Atan2(point.Y, point.X));
    // XNA đặt 0 độ trở lên, trong khi hàm Atan2 trả về tọa độ phía đối diện nên phải thêm 90 độ mới đúng

    targetAngleDegrees += 90;
    // Atan2 trả giá trị giữa -180 và +180, nên thêm 90 độ
    // chúng ta có giá trị trong khoảng -90 đến +270.
    //trong trường hợp giá trị nhỏ hơn 0, thêm 360 để lấy được góc trong 0 đến 360.
    if (targetAngleDegrees < 0) targetAngleDegrees += 360;

    // ...
    if (targetAngleDegrees < angleDegrees - 180)
    {
        targetAngleDegrees += 360;
    }

    if (targetAngleDegrees > angleDegrees + 180)
    {
        targetAngleDegrees -= 360;
    }
```

Phương thức tạo lực đẩy

```
private void Thrust()
{
    Vector2 shipFacing;
    HatVatChat[] particleObjects;
    HatVatChat particleObj;

    //Tính toán vị trí máy bay bị đẩy đến
    shipFacing = new Vector2((float)Math.Sin(Angle), (float)-Math.Cos(Angle));
    // giảm dần và thêm vận tốc
    _velocity += shipFacing / 10;

    // Thêm một số hạt lực đẩy phía sau đuôi máy bay.

    particleObjects = _game.GetParticleObjects(10);

    for (int i = 0; i < particleObjects.Length; i++)
    {
        particleObj = particleObjects[i];
        particleObj.ResetProperties(Position, _game.Textures["SmokeParticle"]);

        particleObj.Position -= shipFacing * SpriteTexture.Width / 2 * ScaleX;
        switch (TroGiupGame.RandomNext(3))
        {
            case 0: particleObj.SpriteColor = Color.White; break;
            case 1: particleObj.SpriteColor = Color.Cyan; break;
            case 2: particleObj.SpriteColor = Color.Blue; break;
        }
        particleObj.Scale = new Vector2(0.25f, 0.25f);
        particleObj.IsActive = true;
        particleObj.Intensity = 50;
        particleObj.IntensityFadeAmount = 3 + TroGiupGame.RandomNext(1.5f);
        particleObj.Speed =TroGiupGame.RandomNext(3.0f);
        particleObj.Inertia = 0.9f + TroGiupGame.RandomNext(0.015f);
        particleObj.Direction += -shipFacing * 5;
    }
}
```

…

### d. ***Lớp đối tượng NgoiSao(Ngôi sao)***

Phương thức tạo nhấp nháy ánh sáng, khởi tạo ngẫu nhiên vị trí

```
_game = game;

PositionX = TroGiupGame.RandomNext(0, _game.Window.ClientBounds.Width);
PositionY = TroGiupGame.RandomNext(0, _game.Window.ClientBounds.Height);

Origin = new Vector2(texture.Width, texture.Height) / 2;


_scaleMin = TroGiupGame.RandomNext(0.1f, 0.6f);
_scaleMax = _scaleMin + TroGiupGame.RandomNext(0.1f, 0.4f);
```

### e. ***Lớp đối tượng HatVatChat (Hạt vật chất)***

Khởi tạo màu sắc và thuộc tính của các hạt vật chất( tạo khói, bụi…)

```
public override Color SpriteColor
{
    get
    {
        Color col;
        byte alpha = 0;

        col = base.SpriteColor;

        if (Intensity >= 0 && Intensity <= 255) alpha = (byte)Intensity;
        if (Intensity > 255) alpha = 255;

        col.A = alpha;

        return col;
    }
}

public override void Update(GameTime gameTime)
{

    if (!IsActive) return;


    Position += Direction * Speed;

    Speed *= Inertia;

    Intensity -= IntensityFadeAmount;

    if (Intensity <= 0)
    {

        IsActive = false;
    }
}
```

### f. ***Lớp đối tượng viên đạn***

Phương thức di chuyển

```
{
    base.Update(gameTime);

    // Chỉ cập nhật nếu viên đạn được kích hoạt
    if (!IsActive) return;

    // thêm vận tốc đến vị trí
    Position += _velocity * 10;

    // bắn vào mục tiêu?
    CheckForCollision();

    // nếu gặp cạnh của cửa sổ thì thiết lập lại vị trí phí đối diện
    if (BoundingBox.Bottom < _game.Window.ClientBounds.Top && _velocity.Y < 0)
    {
        PositionY = _game.Window.ClientBounds.Height + SpriteTexture.Height;
    }
    if (BoundingBox.Top > _game.Window.ClientBounds.Bottom && _velocity.Y > 0)
    {
        PositionY = -SpriteTexture.Height;
    }
    if (BoundingBox.Right < _game.Window.ClientBounds.Left && _velocity.X < 0)
    {
        PositionX = _game.Window.ClientBounds.Width + SpriteTexture.Width;
    }
    if (BoundingBox.Left > _game.Window.ClientBounds.Right && _velocity.X > 0)
    {
        PositionX = -SpriteTexture.Width;
    }
```

Phương thức va chạm

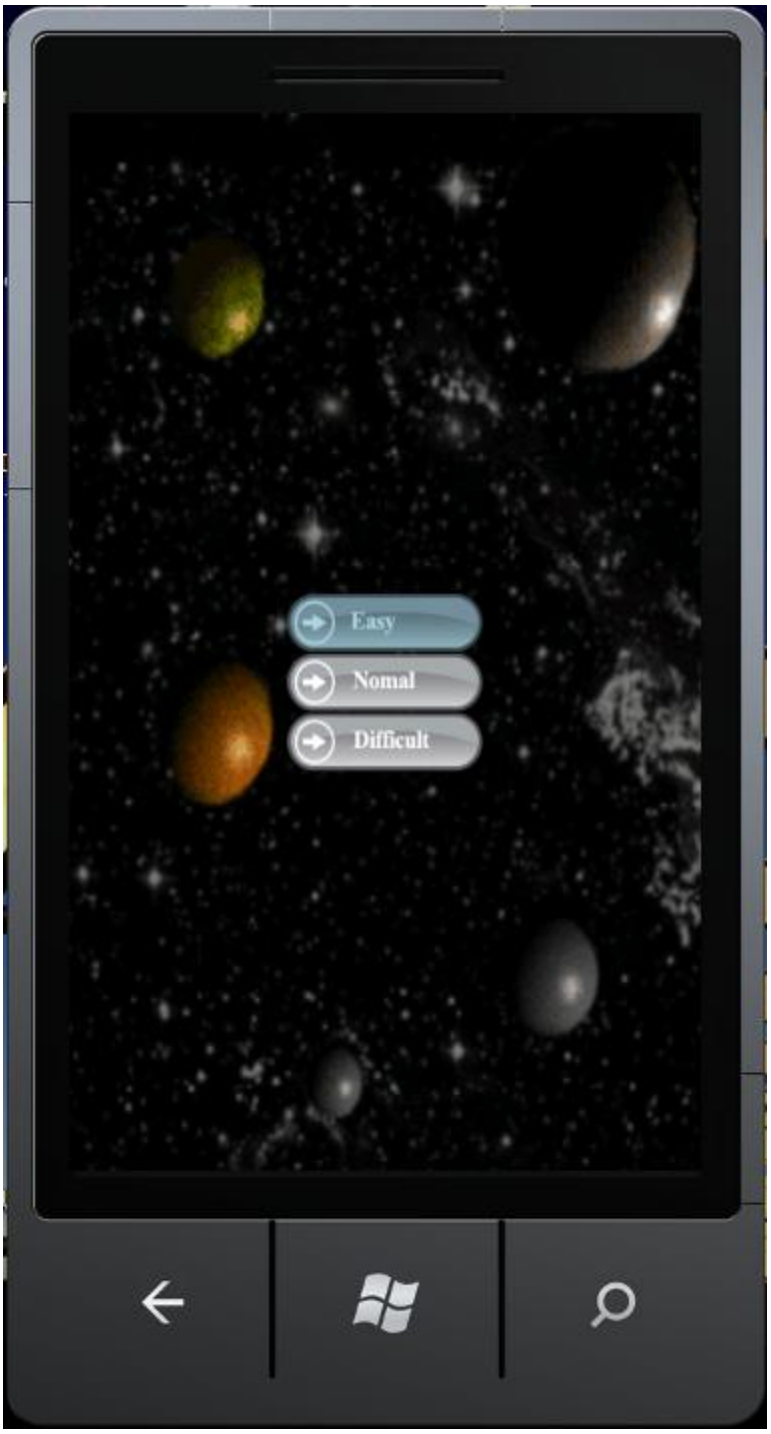
```
private void CheckForCollision()
{
    int objectCount;
    DoiTuongCoSo gameObj;
    UFO rockObj;
    float rockSize;
    float rockDistance;

    //
    objectCount = _game.GameObjects.Count;
    for (int i = objectCount - 1; i >= 0;  i--)
    {
        //tham chiếu đến đối tượng tại vị trí này
        gameObj = _game.GameObjects[i];
        // có phải mục tiêu?
        if (gameObj is UFO)
        {
            // đúng là mục tiêu
            rockObj = (UFO)gameObj;
            if (rockObj.BoundingBox.Contains((int)PositionX, (int)PositionY))
            {
                // khoảng cách đủ nhỏ để chúng va chạm
                // tính toán kích thước của mục tiêu
                rockSize = rockObj.SpriteTexture.Width / 2.0f * rockObj.ScaleX;
                // Tìm khoảng cách giữa hai điểm
                rockDistance = Vector2.Distance(Position, rockObj.Position);
                // nếu khoảng cách nhỏ hơn kích thước mục tiêu?
                if (rockDistance < rockSize)
                {
                    // có sự va chạm giữa viên đạn và  mục tiêu, mục tiêu bị phá hủy
                    rockObj.DamageRock();
                    // viên đạn biến mất
                    IsActive = false;
                    // cập nhật số điểm
                    clsGame.diemso++;
                    clsGame.diemso++;
                    if (clsGame.diemso == clsGame.level * 10 * clsGame.level)
                    {
                        clsGame.level = clsGame.level + 1;
                    }
```

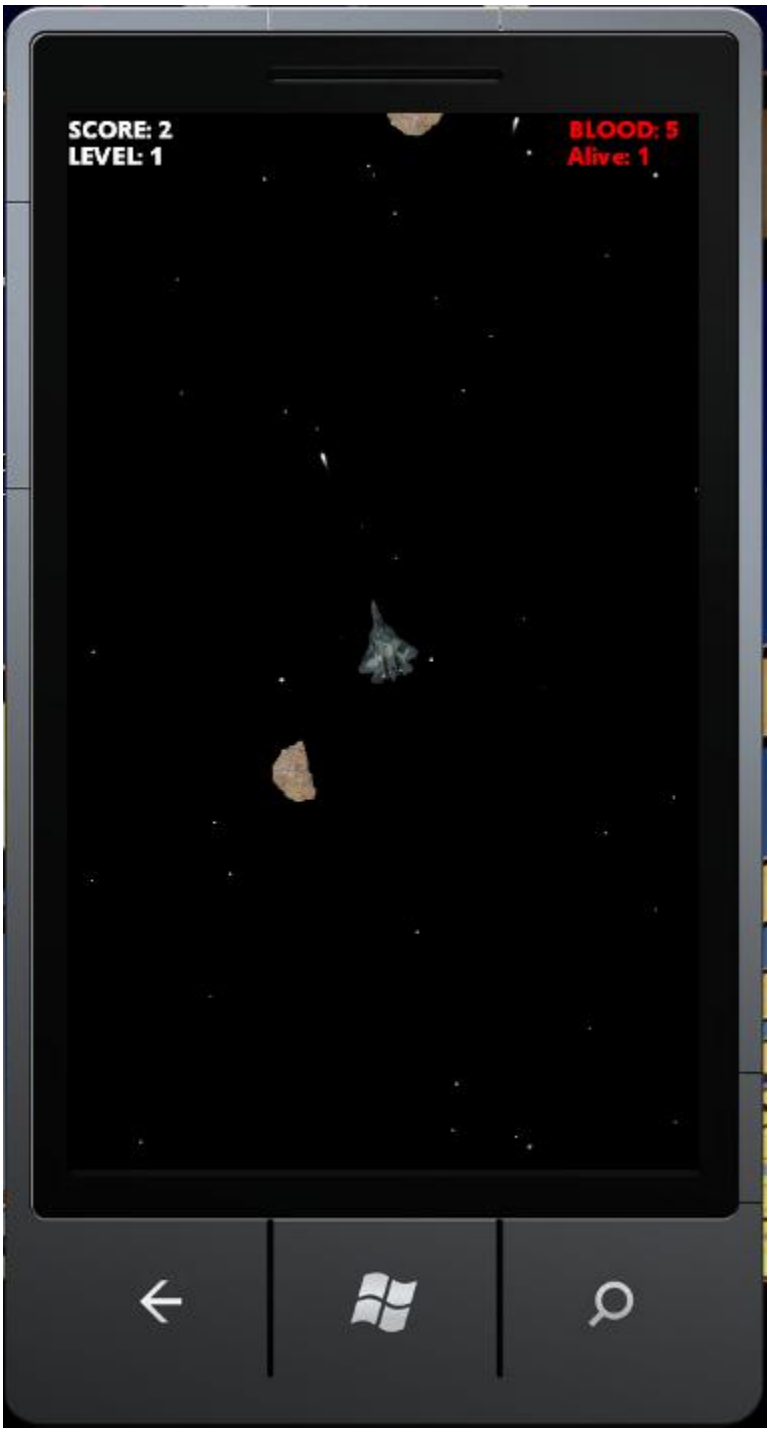
# IV. Demo bằng hình ảnh

1. Giao diện menu game:

2. Sau khi bấm Start Game hiển thị danh mục độ phức tạp game



3. Do hạn chế về thời gian nên nhóm chỉ  mới xây dựng game ở chế độ easy. Bấm vào Easy hiện ra màn chơi 1 của game:



4. Khi hoàn thành màn 1 sẽ có hướng dẫn chuyển sang màn 2!

5. Bấm vào Next map sẽ hiện lên màn 2 của game

6. Nếu không vượt qua được  sẽ xuất hiện thông báo chơi lại màn 2:

7. Nếu vượt qua thì game hoàn thành và có nút chuyển map hiện lên (chức năng chuyển map này hiện tại sẽ reset lại màn chơi này)!

# V. Đánh giá và hướng phát triển

## 1. Đánh giá:

### a. ***Ưu điểm:***

- Game được xây dựng trên nền tảng các lớp đối tượng rõ ràng, dễ dàng thao tác tạo thêm các đối tượng game mới, cũng như thêm các chức năng cho các đối tượng game.
- Đồ họa các chi tiết khá sắc nét (các vụ nổ, lực đẩy,….), khá hài hòa trong giao diên.

### b. ***Nhược điểm:***

- Do thời gian không cho phép nên nhiều chức năng chưa được hoàn thành!
- Game mới có 2 màn, khá đơn giản về nội dung.

## 2. Hướng phát triển

- Tiếp tục hoàn chỉnh chức năng và mở rộng các chức năng trong game như xây dựng thêm các lựa chọn màn chơi, độ khó dễ, tạo thêm các đối tượng game khác tương tác đến đối tượng game chính.
- Từ project đơn giản này giúp nhóm em tự tin hơn trong xây dựng các ứng dung game khác.

# VI. Nguồn và tài liệu tham khảo

1. 3D graphic with xna 4.0 -Sean James
2. Windows Phone 7 Game development – Adam Dawes
3. BeGinning XNA 3.0 Game Programming – Alexandre Santos Lobao, Bruno Evan