ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA CÔNG NGHỆ PHẦN MỀM

BÁO CÁO MÔN HỌC

THỰC HÀNH LẬP TRÌNH TRỰC QUAN

ĐỀ TÀI

GAME CRAZYSHOOTER

Giảng viên hướng dẫn: Huỳnh Hồ Thị Mộng Trinh
Phan Nguyệt Minh
Lớp: IT008.I21.PMCL
Sinh viên thực hiện : Nguyễn Thành Luân – 16520703

TP. Hồ Chí Minh, tháng 7 năm 2018

Mục lục

LỜI CẢM ƠN

Đề tài “Game CrazyShooter” là nội dung em chọn để thực hiện sau khi kết thúc môn học Thực hành Lập trình trực quan tại Trường Đại học Công nghệ thông tin – Đại học Quốc gia Thành phố Hồ Chí Minh.

Để hoàn thành quá trình tìm hiểu kiến thức và xây dựng đề tài, lời đầu tiên em xin chân thành cảm ơn sâu sắc đến Cô Huỳnh Hồ Thị Mộng Trinh và Cô Phan Nguyệt Minh thuộc Khoa Công nghệ phần mềm – Trường Đại học Công nghệ thông tin. Các Cô đã trực tiếp giảng dạy, chỉ bảo và hướng dẫn em trong suốt quá trình học tập và tìm hiểu để thực hiện đề tài này. Ngoài ra em xin chân thành cảm ơn các Thầy, Cô trong Khoa Công nghệ phần mềm đã đóng góp những ý kiến quý báu để đề tài được hoàn thiện hơn.

Cuối cùng, xin cảm ơn những anh chị sinh viên khoá trên, bạn bè đã hỗ trợ, giúp đỡ em trong suốt khoảng thời gian vừa qua!

NHẬN XÉT CỦA GIẢNG VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

PHẦN 1: TỔNG QUAN CRAZYSHOOTER

1. Ý tưởng

Bắt nguồn từ bộ phim "Solo: A Star Wars Story" kể về chuyến phiêu lưu giữa những thiên hà trong vũ trụ bao la của chàng trai trẻ tên Han Solo. Thông qua các cuộc xâm lược táo bạo vào sâu bên trong thế giới ngầm, không ít kẻ thù, những kẻ đã truy đuổi theo con tàu cùng những hiểm nguy bất thình lình trong cái tăm tối của vũ trụ khiến cho cuộc phiêu lưu của chàng trai và những người bạn đồng hành của mình càng thêm hấp dẫn và kịch tính!

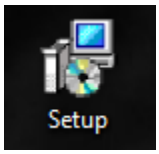
Dựa theo tình tiết bộ phim, CrazyShooter với mục đích mang đến cho người chơi một trải nghiệm tương tự khi tự tay điều khiển một con tàu vượt qua các chướng ngại trong vũ trụ, bắn hạ những kẻ truy đuổi và thoát khỏi lực hút tử thần của hố đen trong suốt cuộc hành trình của mình! Sau đây là khái quát về các thực thể trong trò chơi:

- 'Kẻ lang thang' – Wanderer: Những thực thể trôi nổi trong vũ trụ. Chúng chỉ đơn giản là xuất hiện trên đường đi trong suốt cuộc hành trình và không có mục đích gây hại nào cả. Thế nhưng cũng đừng vì thế mà mất cảnh giác với chúng, bởi chỉ một lần va chạm – con tàu sẽ bị hư hỏng nặng, đồng nghĩa với việc bạn sẽ chết!
- 'Kẻ truy đuổi' – Seeker: Kẻ thù của con tàu. Với mục tiêu truy tìm và huỷ diệt, chúng sẽ không ngần ngại đuổi theo chúng ta trong suốt cuộc hành trình cho đến khi, một trong hai bị tiêu diệt hoàn toàn!
- 'Hố đen' – Black Hole: Nguy hiểm kinh hãi nhất của vũ trụ - 'Lối vào tử thần'. Bạn sẽ không thể thoát khỏi lực hút từ nó một khi đã vướng vào! Ai biết được chuyện gì sẽ xảy ra khi bạn bị cuốn vào nó cơ chứ? Nhưng mà đó là ở trong phim. Còn trong CrazyShooter, cùng với động cơ và hoả lực mạnh mẽ của con tàu bạn có thể thoát khỏi cũng như phá huỷ cánh cổng địa ngục đó. Thật tuyệt vời phải không nào? Tuy nhiên bạn phải thật cẩn trọng, bởi không chỉ bạn mà tất cả thực thể của vũ trụ ở khu vực gần đó cũng bị nó cuốn vào, đồng nghĩa với việc con đường thoát ra của bạn sẽ càng thêm khó khăn đấy!

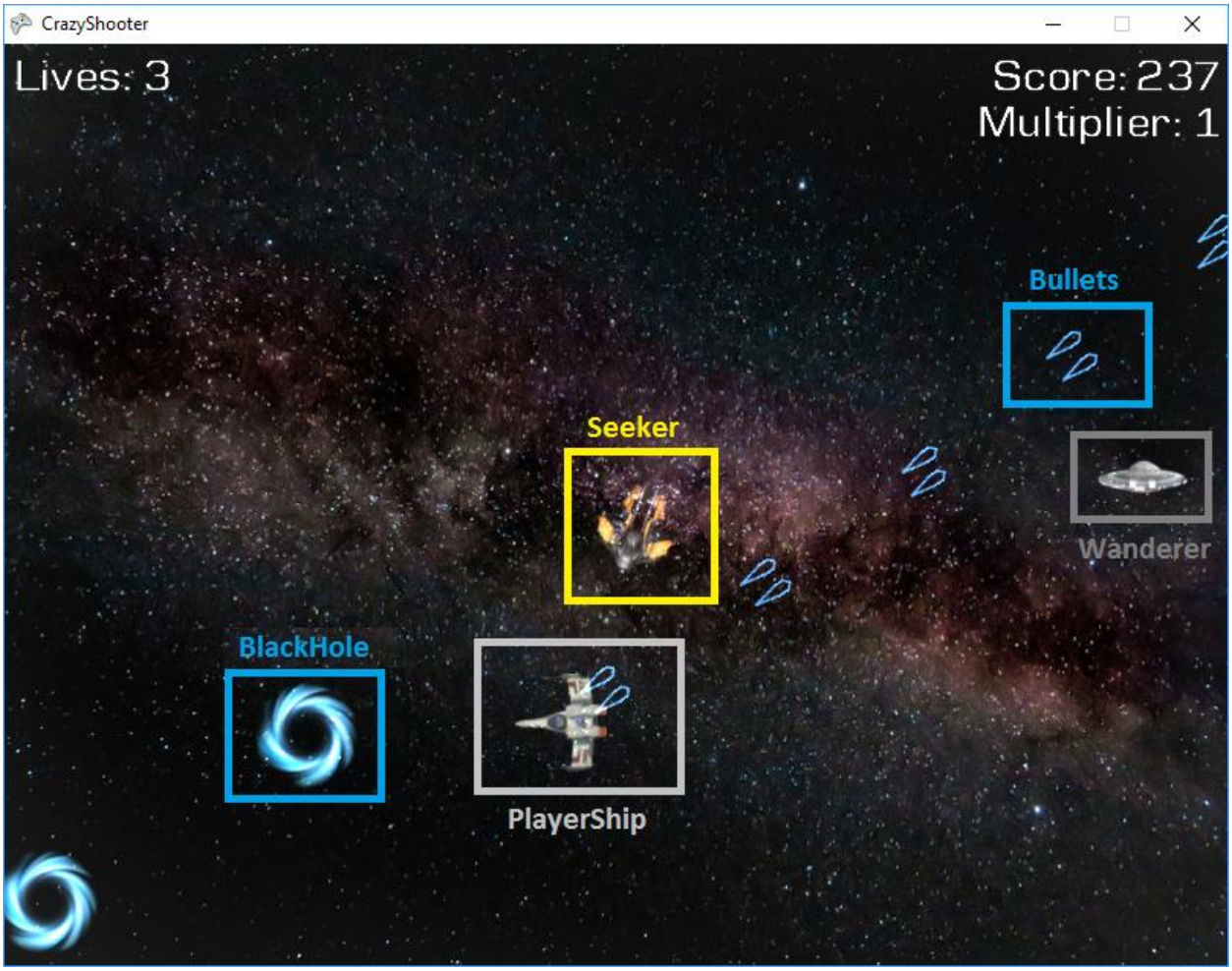
2. Cài đặt

- Tiến hành cài đặt CrazyShooter bằng file Setup.exe

Lưu ý: Vì CrazyShooter đọc ghi file highscore.txt và hiện tại vẫn chưa tìm cách để thiết lập quyền truy cập vào ổ đĩa C được, do đó khi cài đặt cần chọn đường dẫn khác ổ C.



3. Cách chơi



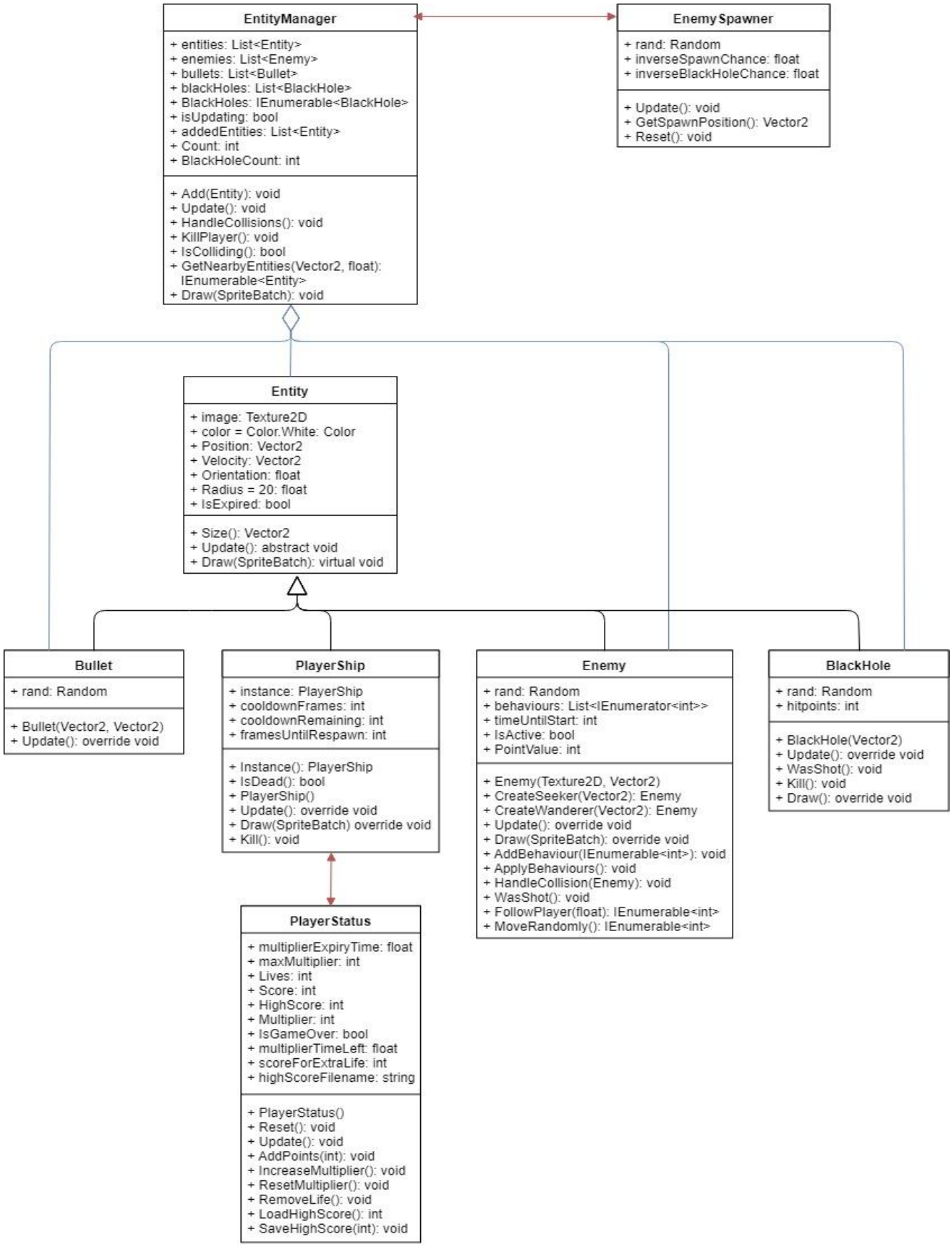
- Trong CrazyShooter, bạn sẽ bắt đầu với 3 mạng (lives), và được tặng thêm mạng sau mỗi 2000 điểm kiếm được.
- Tiêu diệt enemies với nhiều thể loại khác nhau sẽ mang lại mức điểm thưởng khác nhau. Thêm mỗi một enemy bị tiêu diệt sẽ nâng mức hệ số nhân Multiplier lên 1. Nếu không tiêu diệt bất kì một enemy nào trong một khoảng thời gian thì hệ số nhân sẽ tự reset lại bằng 1.
  *Điểm thưởng khi tiêu diệt enemy = Điểm gốc của loại enemy đó * Hệ số nhân*
- Nếu mất tất cả mạng, trò chơi sẽ kết thúc và bạn sẽ bắt đầu lại một trò chơi mới với điểm số và các thông số được reset lại trở về ban đầu.
- Sử dụng các phím 'A', 'W', 'S', 'D' hoặc các phím điều hướng để di chuyển con tàu và sử dụng chuột để ngắm mục tiêu. Bạn không cần phải lo lắng về việc hết đạn trong suốt cuộc hành trình, chỉ việc ngắm mà thôi.
- Sử dụng phím 'P' để tạm dừng trò chơi.
- Chú ý tránh va chạm với các thực thể khác và lực hút từ hố đen.
- Hố đen có thể phản hồi lại đạn! Để huỷ diệt hố đen, bạn phải đến đủ gần!

4. Môi trường phát triển

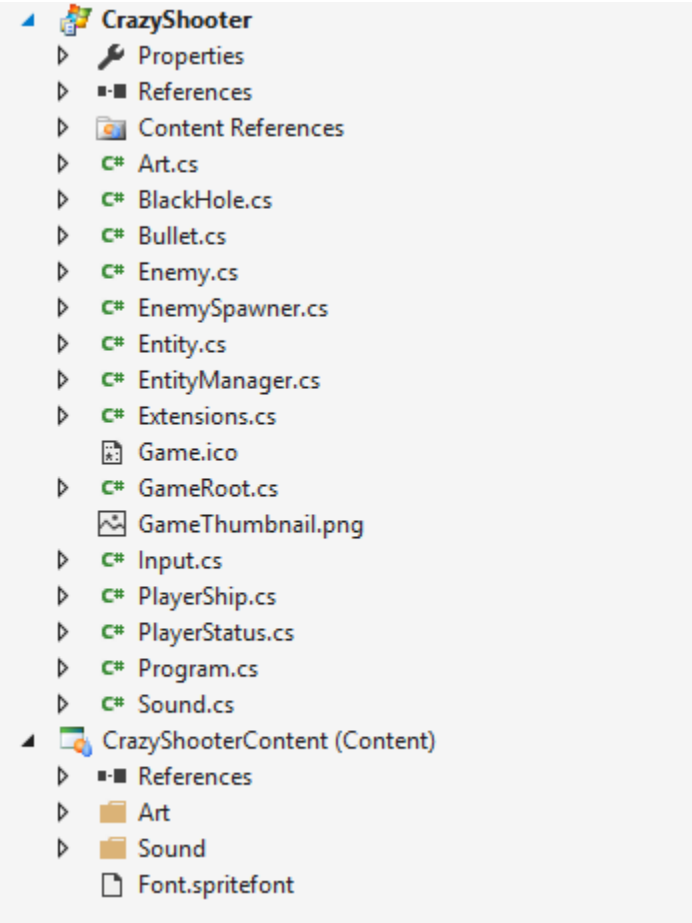
- XNA Game Studio 4.0
  Là môi trường lập trình cho phép người dùng sử dụng Visual Studio để tạo ra trò chơi cho máy tính chạy Windows, Windows Phone và Xbox360. XNA cung cấp sẵn cho người lập trình các công cụ để thiết kế game 2d và 3d, đầy đủ các thành phần đồ hoạ cần thiết để thiết kế game. Bên cạnh đó, XNA xây dựng cấu trúc file riêng để có thể đảm bảo được mã nguồn cho các tài nguyên như hình ảnh, âm thanh…
- C# Language
- IDE Microsoft Visual Studio 2015

PHẦN 2: HIỆN THỰC CRAZYSHOOTER

1. Sơ đồ UML về mối quan hệ giữa các đối tượng


EntityManager
+ entities: List<Entity>
+ enemies: List<Enemy>
+ bullets: List<Bullet>
+ blackHoles: List<BlackHole>
+ BlackHoles: IEnumerable<BlackHole>
+ isUpdating: bool
+ addedEntities: List<Entity>
+ Count: int
+ BlackHoleCount: int
+ Add(Entity): void
+ Update(): void
+ HandleCollisions(): void
+ KillPlayer(): void
+ IsColliding(): bool
+ GetNearbyEntities(Vector2, float): IEnumerable<Entity>
+ Draw(SpriteBatch): void
EnemySpawner
+ rand: Random
+ inverseSpawnChance: float
+ inverseBlackHoleChance: float
+ Update(): void
+ GetSpawnPosition(): Vector2
+ Reset(): void
Entity
+ image: Texture2D
+ color = Color.White: Color
+ Position: Vector2
+ Velocity: Vector2
+ Orientation: float
+ Radius = 20: float
+ IsExpired: bool
+ Size(): Vector2
+ Update(): abstract void
+ Draw(SpriteBatch): virtual void
Bullet
+ rand: Random
+ Bullet(Vector2, Vector2)
+ Update(): override void
PlayerShip
+ instance: PlayerShip
+ cooldownFrames: int
+ cooldownRemaining: int
+ framesUntilRespawn: int
+ Instance(): PlayerShip
+ IsDead(): bool
+ PlayerShip()
+ Update(): override void
+ Draw(SpriteBatch) override void
+ Kill(): void
Enemy
+ rand: Random
+ behaviours: List<IEnumerator<int>>
+ timeUntilStart: int
+ IsActive: bool
+ PointValue: int
+ Enemy(Texture2D, Vector2)
+ CreateSeeker(Vector2): Enemy
+ CreateWanderer(Vector2): Enemy
+ Update(): override void
+ Draw(SpriteBatch): override void
+ AddBehaviour(IEnumerable<int>): void
+ ApplyBehaviours(): void
+ HandleCollision(Enemy): void
+ WasShot(): void
+ FollowPlayer(float): IEnumerable<int>
+ MoveRandomly(): IEnumerable<int>
BlackHole
+ rand: Random
+ hitpoints: int
+ BlackHole(Vector2)
+ Update(): override void
+ WasShot(): void
+ Kill(): void
+ Draw(): override void
PlayerStatus
+ multiplierExpiryTime: float
+ maxMultiplier: int
+ Lives: int
+ Score: int
+ HighScore: int
+ Multiplier: int
+ IsGameOver: bool
+ multiplierTimeLeft: float
+ scoreForExtraLife: int
+ highScoreFilename: string
+ PlayerStatus()
+ Reset(): void
+ Update(): void
+ AddPoints(int): void
+ IncreaseMultiplier(): void
+ ResetMultiplier(): void
+ RemoveLife(): void
+ LoadHighScore(): int
+ SaveHighScore(int): void

2. Tổ chức mã nguồn



Nhiệm vụ của từng class:

- **Art**: Load dữ liệu và tham chiếu đến hình ảnh.
- **Sound**: Load dữ liệu và tham chiếu đến âm thanh.
- **Entity**: Lớp cơ sở trừu tượng cho Bullet, Enemy, BlackHole và PlayerShip.
- **EntityManager**: Quản lý, theo dõi tất cả các loại Entity trong trò chơi và xử lý xung đột.
- **BlackHole**: Black Hole và các phương thức liên quan (Update, Draw…)
- **Enemy**: Khởi tạo và thiết đặt hình thức hoạt động cho 2 loại enemy (Seeker và Wanderer).
- **EnemySpawner**: Quản lý thời gian phát sinh một Enemy mới (Quản lý độ khó của trò chơi).
- **Bullet**: Bullet và các phương thức liên quan (Update, Draw…)
- **PlayerShip**: PlayerShip và các phương thức liên quan (Update, Draw…)

- **PlayerStatus**: Theo dõi điểm số, mạng của người chơi.
- **Input**: Quản lý input từ chuột và bàn phím.
- **Extensions**: Chứa một số phương thức mở rộng.
- **GameRoot**: Vòng lặp chính của trò chơi.
- **Program**: Entry point.

3. Vòng lặp chương trình

3.1. Entry point

```
static void Main(string[] args)
{
    using (GameRoot game = new CrazyShooter.GameRoot())
    {
        game.Run();
    }
}
```

3.2. Khởi tạo

- Thiết lập kích thước buffer (800, 600).
- Thêm instance của PlayerShip vào EntityManager để quản lý.
- Load hình ảnh và âm thanh.

```
// constructor GameRoot
public GameRoot()
{
    // khởi tạo
    Instance = this;
    graphics = new GraphicsDeviceManager(this);
    Content.RootDirectory = "Content";

    // thiết lập kích thước buffer
    graphics.PreferredBackBufferWidth = 800;
    graphics.PreferredBackBufferHeight = 600;
}

protected override void Initialize()
{
    base.Initialize();

    // thêm PlayerShip vào EntityManager
    EntityManager.Add(PlayerShip.Instance);

    MediaPlayer.IsRepeating = true;
    MediaPlayer.Play(Sound.Music);
}

// Load hình ảnh, âm thanh
protected override void LoadContent()
{
    spriteBatch = new SpriteBatch(GraphicsDevice);
    Art.Load(Content);
```

```
30.     Sound.Load(Content);
31. }
```

3.3. Cập nhật

3.3.1. Cập nhật EntityManager.cs

- Xử lý đụng độ.
- Cập nhật trạng thái cho từng loại entity (Enemy, Bullet, BlackHole, PlayerShip)
    - Enemy: Cho phép enemy đó hoạt động hay dừng hoạt động, thiết lập vận tốc ban đầu, hành vi (MoveRandomly đối với Wanderer và FollowPlayer đối với Seeker).
    - Bullet: Thay đổi góc xoay của bullet, xoá các bullets ra khỏi màn hình.
    - BlackHole: Lấy danh sách các entities nằm trong tầm ảnh hưởng của lực hút (250 pixel) và xử lý lực hút, đẩy lùi đạn.
    - PlayerShip: Xử lý Killed, lấy góc nhắm (từ chuột) và thêm 2 viên bullets song song. Xử lý thời gian cooldown giữa các frame để đạn không ra quá nhanh.
- Cập nhật lại danh sách các loại entity (thêm mới và xoá những entity bị expired).

```
1.  public static void Update()
2.  {
3.      isUpdating = true;
4.
5.      // xử lý đụng độ
6.      HandleCollisions();
7.
8.      // update cho từng entity
9.      foreach (var entity in entities)
10.         entity.Update();
11.
12.     isUpdating = false;
13.
14.     // cập nhật lại các danh sách entities, bullets, enemies, blackHoles
15.     // Note: Nếu thay đổi trực tiếp ở các list trên trong quá trình lặp sẽ gây ra vòng
    lặp vô hạn
16.     // Do đó cần một list addedEntities để lưu những entities mới và tiến hành cập nhật
     list cũ như sau.
17.     foreach (var entity in addedEntities)
18.     {
19.         entities.Add(entity);
20.         if (entity is Bullet)
21.             bullets.Add(entity as Bullet);
22.         else if (entity is Enemy)
23.             enemies.Add(entity as Enemy);
24.         else if (entity is BlackHole)
25.             blackHoles.Add(entity as BlackHole);
26.     }
27.
28.     addedEntities.Clear();
29.
30.     // loại bỏ những entities bị expired.
31.     entities = entities.Where(x => !x.IsExpired).ToList();
```

```
    bullets = bullets.Where(x => !x.IsExpired).ToList();
    enemies = enemies.Where(x => !x.IsExpired).ToList();
    blackHoles = blackHoles.Where(x => !x.IsExpired).ToList();
}
```

3.3.2. Cập nhật EnemySpawner.cs

- Quyết định sự xuất hiện của một entity bằng việc random và chọn giá trị mặc định trong khoảng cho trước (inverseSpawnChance, inverseBlackHoleChance).
- Thay đổi độ khó của game bằng cách giảm giới hạn của miền chọn random xuống sau mỗi frame.

```
public static void Update()
{
    if (!PlayerShip.Instance.IsDead && EntityManager.Count < 200)
    {
        // random trong khoảng (0,inverseSpawnChance) để quyết định thêm mới Seeker
        if (rand.Next((int)inverseSpawnChance) == 0)
            EntityManager.Add(Enemy.CreateSeeker(GetSpawnPosition()));

        // random trong khoảng (0,inverseSpawnChance) để quyết định thêm mới Wanderer
        if (rand.Next((int)inverseSpawnChance) == 0)
            EntityManager.Add(Enemy.CreateWanderer(GetSpawnPosition()));

        // random trong khoảng (0,inverseBlackHoleChance) để quyết định thêm mới BlackHole
        if (EntityManager.BlackHoleCount < 2 && rand.Next((int)inverseBlackHoleChance) == 0)
            EntityManager.Add(new BlackHole(GetSpawnPosition()));
    }

    // giảm mức random time -> đẩy nhanh thời gian xuất hiện ngẫu nhiên của enemy -> tăng độ khó
    if (inverseSpawnChance > 30)
        inverseSpawnChance -= 0.005f;
    if (inverseBlackHoleChance > 200)
        inverseBlackHoleChance -= 0.01f;
}
```

3.3.3. Cập nhật PlayerStatus.cs

- Thay đổi hệ số nhân sau mỗi multiplierExpiryTime.

```
public static void Update()
{
    if (Multiplier > 1)
    {
        // cập nhật lại thông số cho multiplier
        if ((multiplierTimeLeft -
= (float)GameRoot.GameTime.ElapsedGameTime.TotalSeconds) <= 0)
        {
            multiplierTimeLeft = multiplierExpiryTime;
            Multiplier = 1;
```

```
        }
    }
}
```

3.4. Xử lý đụng độ

- Sử dụng thuộc tính Radius ở class Entity để xác định phạm vi va chạm của thực thể.
- Vòng lặp duyệt tất cả các list trong EntityManager để xác định va chạm và đưa ra xử lý thích hợp.
  - Va chạm giữa 2 enemies: Đẩy ra xa nhau.
  - Va chạm giữa bullet và enemy: gán IsExpired = true ở cả bullet và enemy để lần cập nhật tới xoá bullet và enemy đã va chạm, đồng thời AddPoints và IncreaseMultiplier cho PlayerStatus.
  - Va chạm giữa PlayerShip và Enemies hoặc BlackHole: KillPlayer và reset lại thông số độ khó trong EnemySpawner.
  - Va chạm giữa Enemies và BlackHole: IsExpired = true, AddPoints và IncreaseMultiplier cho PlayerStatus.
  - Va chạm giữa Bullets và BlackHole: giảm mức hitpoints của BlackHole xuống mỗi 1 đơn vị. Nếu hitpoints <= 0 thì BlackHole bị tiêu diệt và expired. Mặc định hitpoints = 20.

```
static void HandleCollisions()
{
    // xử lý đụng độ giữa enemies
    for (int i = 0; i < enemies.Count; i++)
        for (int j = i + 1; j < enemies.Count; j++)
        {
            if (IsColliding(enemies[i], enemies[j]))
            {
                enemies[i].HandleCollision(enemies[j]);
                enemies[j].HandleCollision(enemies[i]);
            }
        }

    // bullets và enemies
    for (int i = 0; i < enemies.Count; i++)
        for (int j = 0; j < bullets.Count; j++)
        {
            if (IsColliding(enemies[i], bullets[j]))
            {
                enemies[i].WasShot();
                bullets[j].IsExpired = true;
            }
        }

    // player và enemies
    for (int i = 0; i < enemies.Count; i++)
    {
        if (enemies[i].IsActive && IsColliding(PlayerShip.Instance, enemies[i]))
        {
```

```
            KillPlayer();
            break;
        }
    }

    // enemies và bullets với black holes
    for (int i = 0; i < blackHoles.Count; i++)
    {
        for (int j = 0; j < enemies.Count; j++)
            if (enemies[j].IsActive && IsColliding(blackHoles[i], enemies[j]))
                enemies[j].WasShot();

        for (int j = 0; j < bullets.Count; j++)
        {
            if (IsColliding(blackHoles[i], bullets[j]))
            {
                bullets[j].IsExpired = true;
                blackHoles[i].WasShot();
            }
        }

        // playership với blackhole
        if (IsColliding(PlayerShip.Instance, blackHoles[i]))
        {
            KillPlayer();
            break;
        }
    }
}
```

3.5. Vẽ ra màn hình

- Clear buffer.
- Vẽ danh sách entity trong EntityManager.
- Hiển thị thông tin người chơi (Lives, Score, Multiplier).
- Hiển thị con trỏ chuột.
- Xét trường hợp Game Over (Xuất Your Score, High Score và reset lại game sau 500 frame).

```
protected override void Draw(GameTime gameTime)
{
    base.Draw(gameTime);

    // clear buffer
    GraphicsDevice.Clear(Color.Black);

    // vẽ danh sách các entity
    spriteBatch.Begin(SpriteSortMode.Texture, BlendState.Additive);
    EntityManager.Draw(spriteBatch);
    spriteBatch.End();

    // xuất thông tin người chơi
    spriteBatch.Begin(SpriteSortMode.Deferred, BlendState.Additive);

```

```
16.     spriteBatch.DrawString(Art.Font, "Lives: " + PlayerStatus.Lives, new Vector2(5), Co
    lor.White);
17.     spriteBatch.DrawString(Art.Font, "Score: " + PlayerStatus.Score, new Vector2(Screen
    Size.X - Art.Font.MeasureString("Score: " + PlayerStatus.Score).X - 5, 5), Color.White)
    ;
18.     spriteBatch.DrawString(Art.Font, "Multiplier: " + PlayerStatus.Multiplier, new Vect
    or2(ScreenSize.X - Art.Font.MeasureString("Multiplier: " + PlayerStatus.Multiplier).X -
     5, 35), Color.White);
19.
20.     if (PlayerStatus.IsGameOver)
21.     {
22.         string text = "Game Over\n" +
23.             "Your Score: " + PlayerStatus.Score + "\n" +
24.             "High Score: " + PlayerStatus.HighScore;
25.
26.         Vector2 textSize = Art.Font.MeasureString(text);
27.         spriteBatch.DrawString(Art.Font, text, ScreenSize / 2 - textSize / 2, Color.Whi
    te);
28.     }
29.
30.     // con trỏ chuột
31.     spriteBatch.Draw(Art.Pointer, Input.MousePosition, Color.White);
32.
33.     spriteBatch.End();
34. }
```

4. Các phương thức phụ trợ (Extensions)

4.1. Xoay hình ảnh

```
1.  public static float ToAngle(this Vector2 vector)
2.  {
3.      return (float)Math.Atan2(vector.Y, vector.X);
4.  }
```

4.2. Thu hẹp khoảng cách giữa 2 entities

- BlackHole hút các Entities.
- Seeker theo dấu PlayerShip.

```
1.  public static Vector2 ScaleTo(this Vector2 vector, float length)
2.  {
3.      return vector * (length / vector.Length());
4.  }
```

PHẦN 3: TỔNG KẾT

1. Tổng kết

Đề tài CrazyShooter đã được phát triển và hoàn thiện đúng theo ý tưởng ban đầu đề ra. Tuy nhiên vì được làm trong thời gian khá ngắn, CrazyShooter chỉ dừng lại ở mức hiện thực được ý tưởng và còn nhiều vấn đề cần giải quyết thêm để tăng giá trị của trò chơi cũng như trải nghiệm cho người dùng.

2. Nhận xét và đánh giá

- Trong quá trình tìm hiểu và hoàn thiện đề tài, em đã đạt được những điều sau:
  - Kiến thức, kỹ năng mới trong việc thiết kế, xây dựng game với XNA.
  - Áp dụng kiến thức toán để xử lý chuyển động, hướng chuyển động.
  - Tư tưởng hướng đối tượng được cải thiện rõ rệt.
  - Sự hỗ trợ, giúp đỡ từ thầy cô và bạn bè.
- Tuy CrazyShooter đã hoàn thiện theo những định hướng ban đầu, nếu có thời gian em sẽ:
  - Cải thiện, nâng cao đồ hoạ trò chơi.
  - Thêm stage, thêm các loại thực thể của vũ trụ, thêm hình thức thưởng/phạt cho người chơi…
  - Mở rộng và release sản phẩm cho các nền tảng hỗ trợ khác nhau.

3. Tiến độ công việc

No.	Thời gian	Công việc	% Finished
1	Trước 07/2018	Lên ý tưởng đề tài, tìm hiểu các kiến thức về nền tảng XNA có liên quan.	100%
2	02-03/07/2018	Thiết kế dữ liệu, vẽ sơ đồ mối quan hệ giữa các thực thể.	100%
3	04/07/2018	Hiện thực các class Entity, EntityManager, Input, PlayerShip, Art, Sound, Bullet (di chuyển tàu, bắn đạn, hình ảnh, âm thanh)	80%
4	05/07/2018	*Máy tính bị hư nên phải tạo lại project.* Thực hiện lại công việc của ngày 04/07.	90%

		Tiếp tục tạo thêm các class thực thể mới (Enemy, BlackHole)	
5	06/07/2018	Thêm class hỗ trợ xuất hình ảnh Extension. Thêm class PlayerStatus. Hoàn thành sản phẩm.	80%
6	06-07/07/2018	Tiếp tục hoàn thiện sản phẩm (chỉnh sửa lại lượt đạn, tầm ảnh hưởng của hố đen…) Viết báo cáo.	90%
7	07/07/2018	Hoàn thiện và đóng gói sản phẩm	100%

4. Tài liệu tham khảo

- Install XNA 4.0 in Visual Studio 2015 https://stackoverflow.com/questions/28008970/how-to-install-xna-in-visual-studio-2015-preview
- XNA Game 2D Tutorials http://blogs.microsoft.co.il/pavely/2010/11/01/xna-2d-game-tutorial-part-1/
- Load Content in XNA http://rbwhitaker.wikidot.com/another-way-to-do-audio-in-xna
- Entity moving https://gamedev.stackexchange.com/questions/28334/make-objects-follow-a-strict-path-xna
- XNA 4.0 https://docs.microsoft.com/en-us/previous-versions/windows/xna/bb200104(v=xnagamestudio.41)
- C# Language https://docs.microsoft.com/en-us/dotnet/csharp/