C# 프로젝트

Project Galag

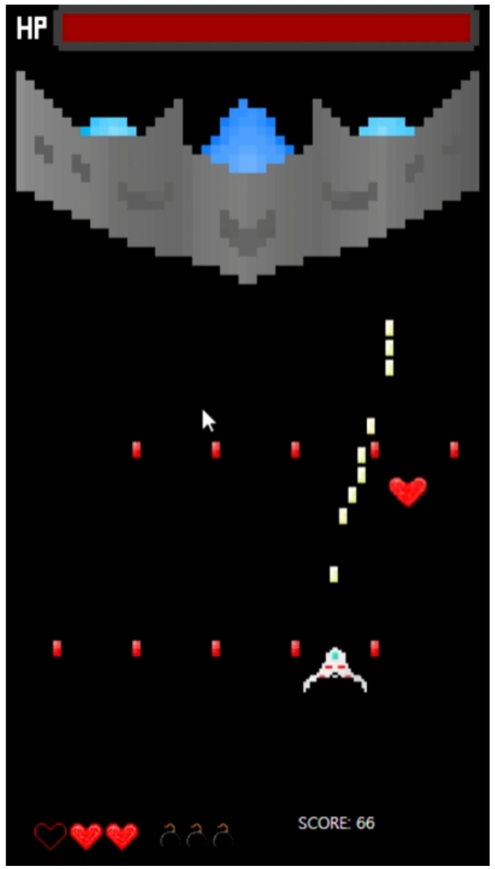
선동운

CONTENTS

01

게임 소개

게임 소개



제목 : Galag

장르 : 슈팅 게임

주요 라이브러리 : Windows forms

언어 : C#

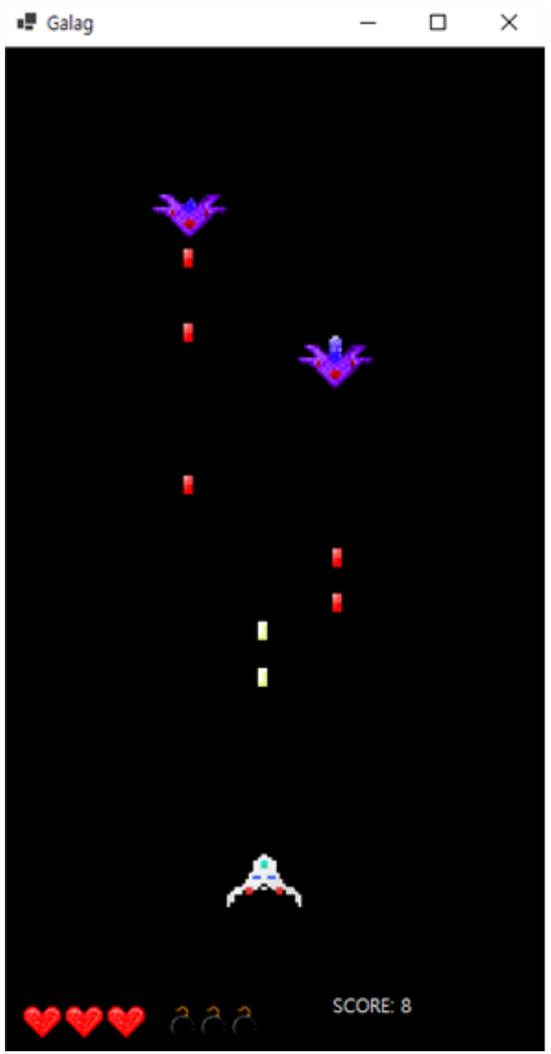
개발 인원 : 1명

시연 영상 : https://youtu.be/8r_5vPcWxvI?si=cAmTrVmBokNc4txC

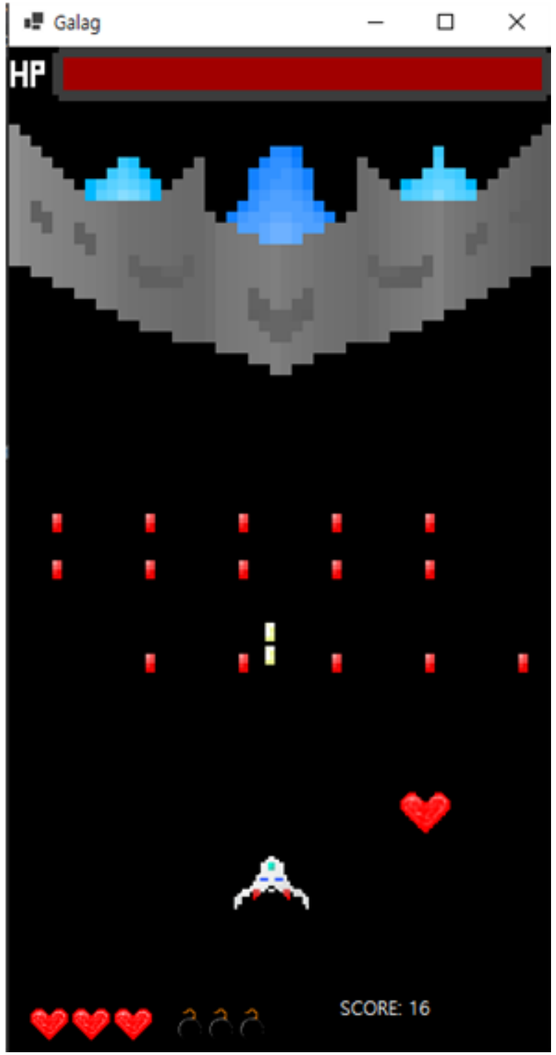
02

게임 구성

기본 화면 구성

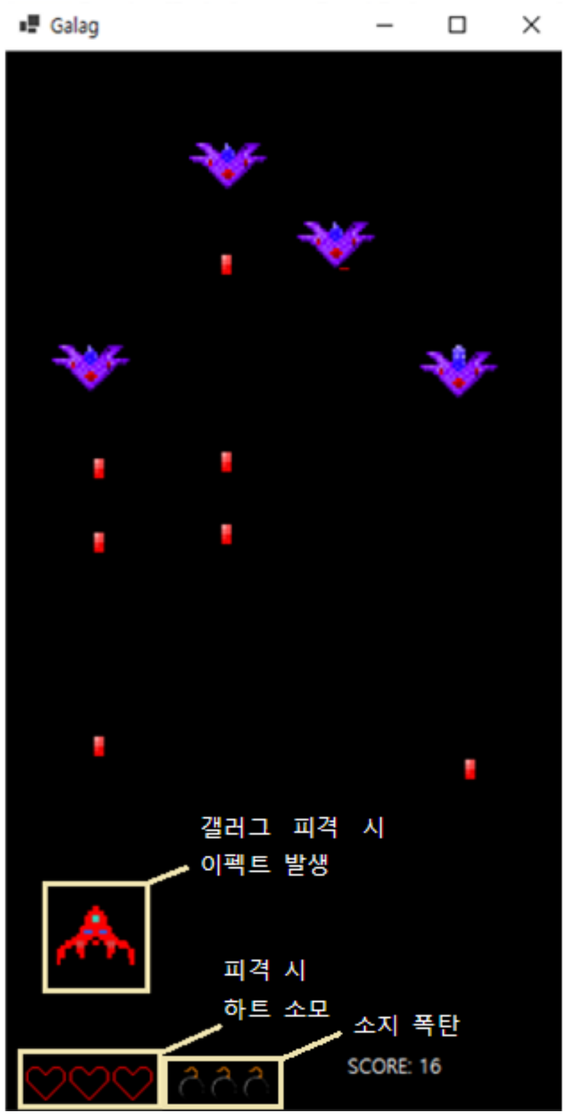


[보스 등장 전 1페이즈]



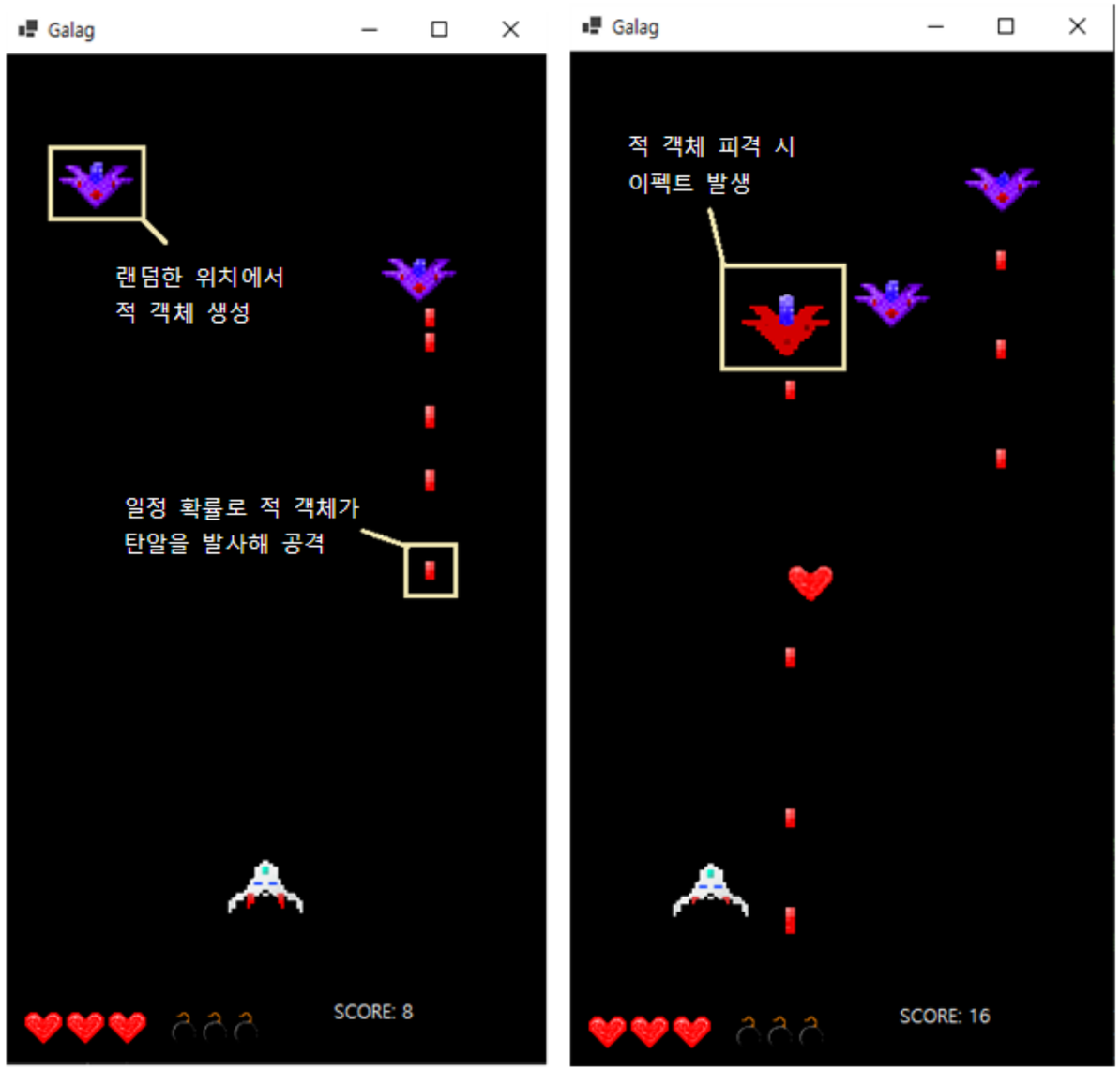
[보스 등장 후 2페이즈]

갤러그 동작 방식



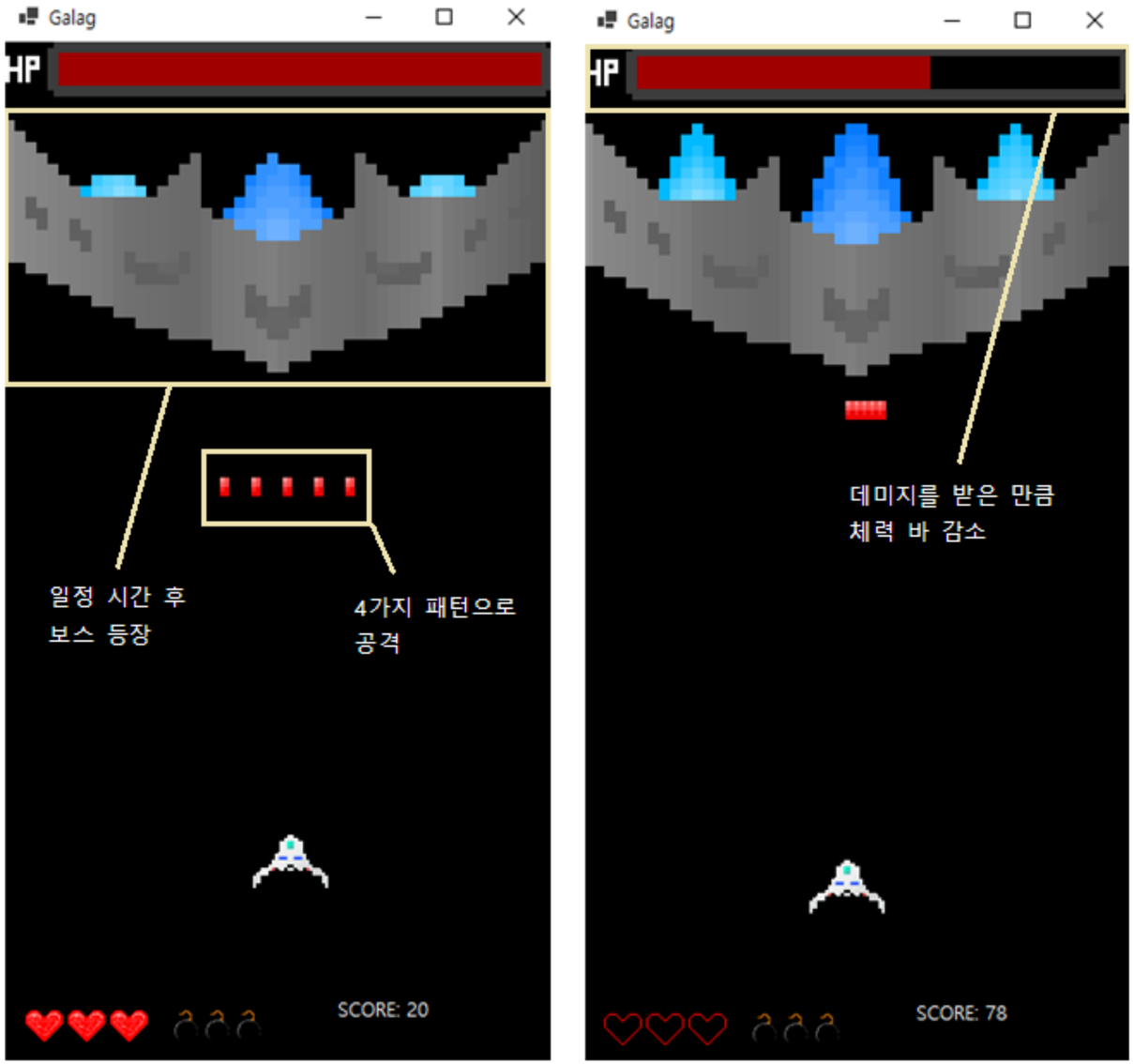
- 기본적으로 방향키를 이용해 이동할 수 있으며 Space 바 키를 이용해 갤러그 탄알을 발사하여 공격할 수 있습니다.
- 아이템의 경우 F키를 사용하여 소지하고 있는 폭탄을 1개 소모하여 사용할 수 있습니다.
- 적 탄알에 피격 시 피격 이펙트를 보여주며 하트 1개가 소모됩니다.

일반 적 동작 방식



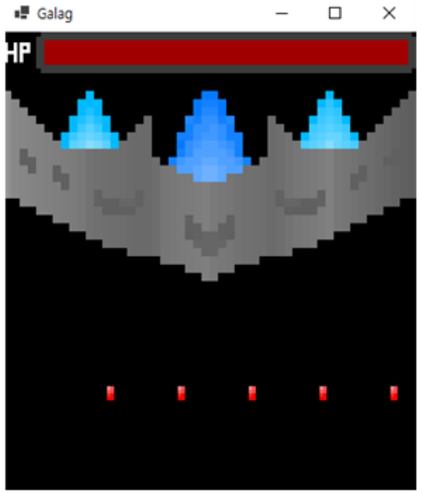
- 기본적으로 화면의 일정한 공간 내에서 랜덤한 위치에서 생성됩니다.
- 생성된 위치에 고정되어 있고 일정 확률로 탄알을 발사하여 갤러그를 공격합니다.
- 갤러그 탄알에 피격 시 피격 이펙트를 보여주고 1의 데미지를 받습니다. 총 3의 데미지를 받은 적은 제거됩니다.

보스 동작 방식

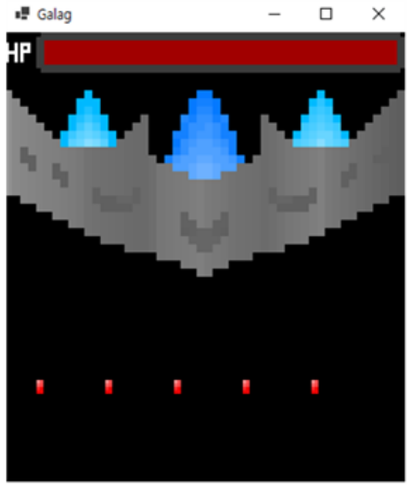


- 일정 시간 동안 일반 적의 공격을 버티게 되면 화면에 있던 모든 적들이 사라지고 보스가 생성됩니다.
- 정해진 위치에 생성되어 고정되어 있고 일정 확률로 탄알을 발사해 갤러그를 공격합니다.
- 갤러그 탄알에 피격 시 1의 데미지, 폭탄으로 5데미지를 받습니다. 피격된 데미지에 따라 체력 바가 감소합니다. 총 50의 데미지를 받게 되면 보스는 제거되고 게임을 클리어하게 됩니다.

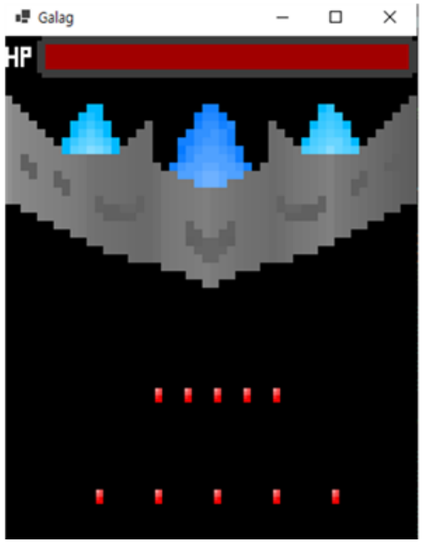
보스 동작 방식



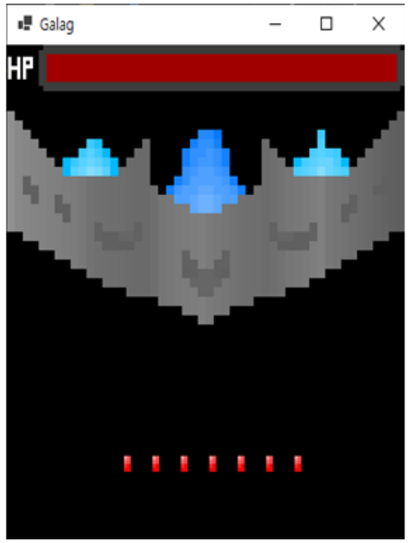
[우측 공격 패턴]



[좌측 공격 패턴]



[방사형 공격 패턴]



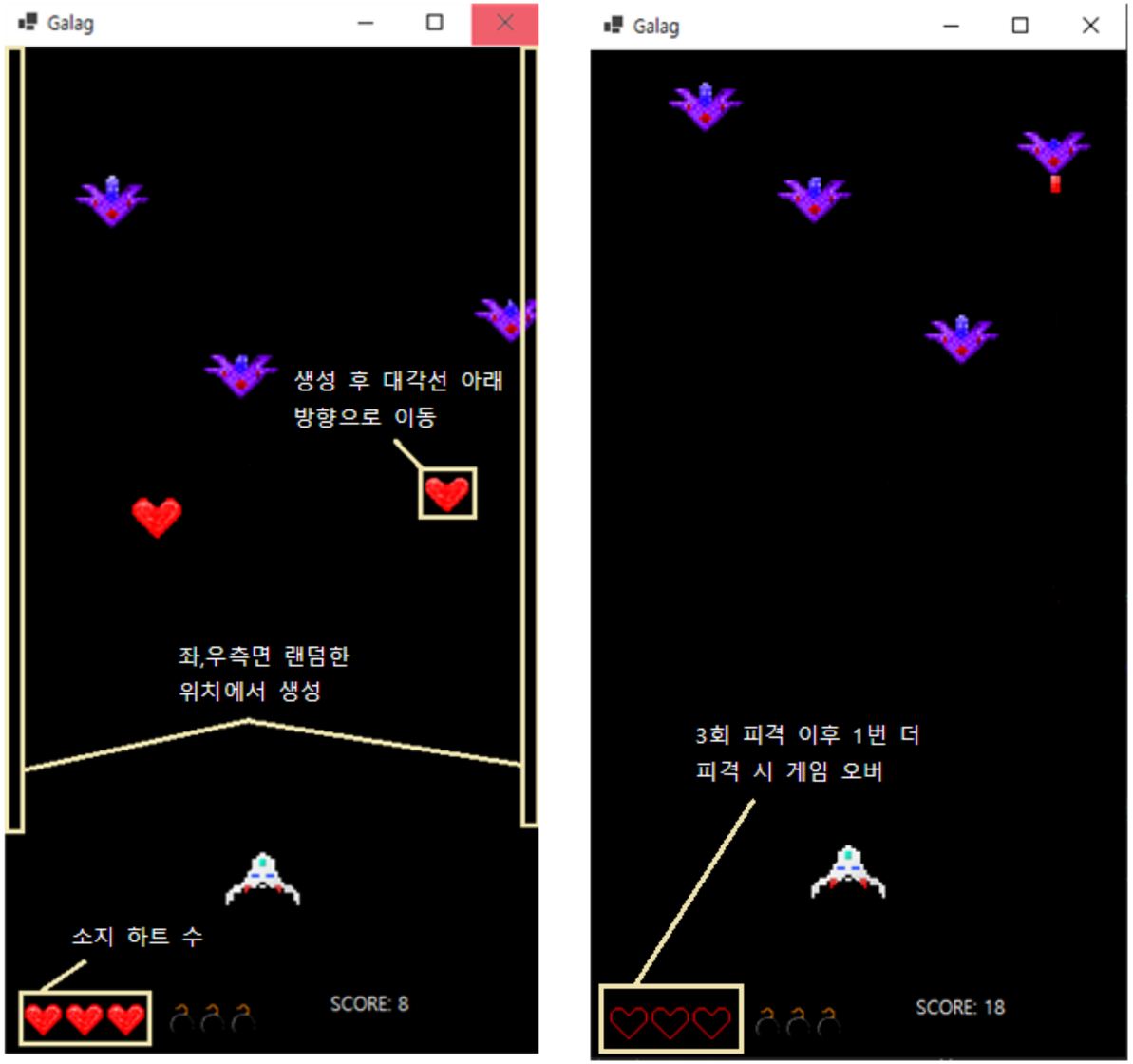
[더 빠른 방사형 공격 패턴]

- 공격 시 4가지의 공격 패턴을 가지고 있어서 일정 확률에 따라 공격 패턴을 선택해 공격합니다. 우측 공격, 좌측 공격, 방사형 공격, 더 빠른 방사형 공격으로 공격 패턴이 구성되어 있습니다.

폭탄 생성 및 획득

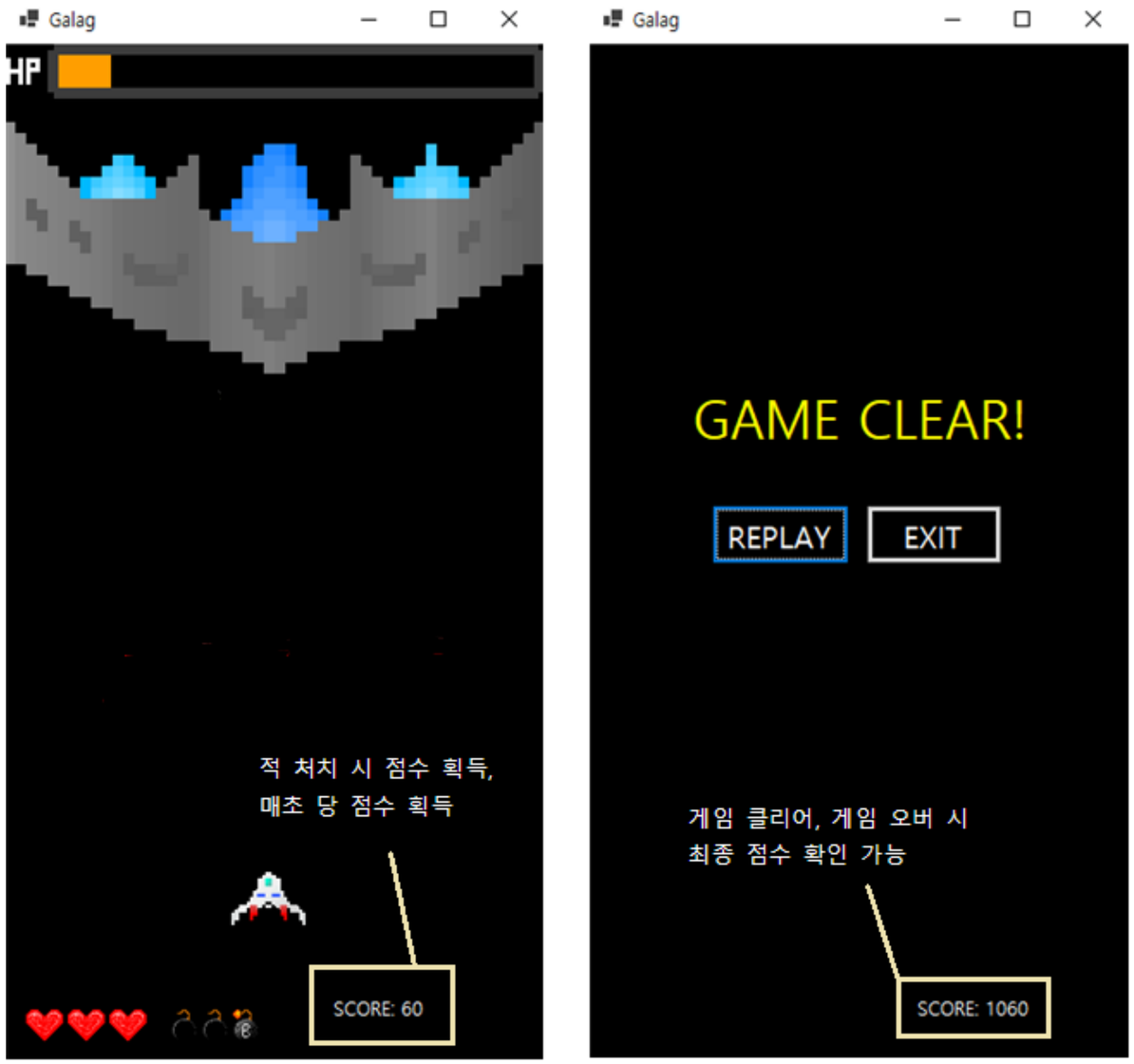
- 화면상의 좌측면, 우측면 중 한 곳에서 랜덤하게 생성되며 대각선 아래 방향으로 움직입니다.
- 갤러그와 닿으면 갤러그는 폭탄 1개를 획득할 수 있습니다. 폭탄은 최대 3개까지 소지 가능합니다.
- 폭탄 사용 시 화면상에 등장해 있는 모든 적 객체와 적 탄알을 제거합니다. 보스 객체에게는 경우 5의 데미지를 주고 적 탄알을 모두 제거합니다.

하트 생성 및 획득



- 화면상의 좌측면, 우측면 둘 중 한 곳에서 랜덤하게 생성되며 대각선 아래 방향으로 움직입니다.
- 갤러그와 닿으면 갤러그는 하트 1개를 획득할 수 있습니다. 하트는 최대 3개까지 소지 가능합니다.
- 갤러그는 하트가 3개가 모두 소모된 상태에서 한 번 더 피격 시 게임 오버됩니다

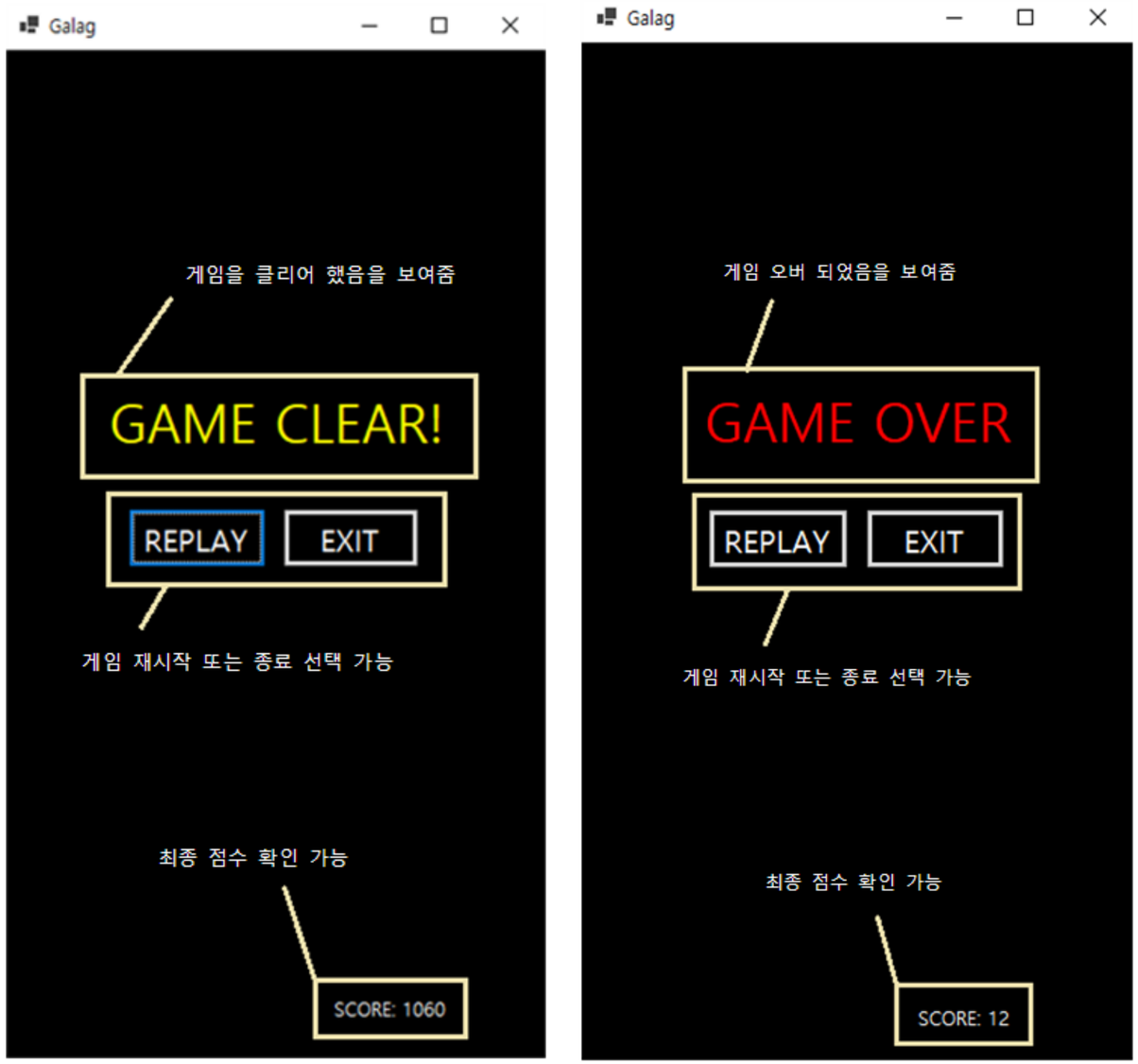
점수 기록



- 매 초당 2점씩 획득하게 되고, 적 객체 처치 시 10점, 보스 객체 처치 시 1000점을 획득 할 수 있습니다.

- 게임 오버 또는 게임 클리어 시 화면 우측 하단에서 게임이 종료된 시점의 점수를 확인할 수 있습니다.

게임 클리어 / 게임 오버



- 게임 클리어 / 게임 오버를 보여주고 Replay 버튼을 누르면 게임을 다시 시작할 수 있습니다

- Exit 버튼을 누르면 게임을 종료합니다

03

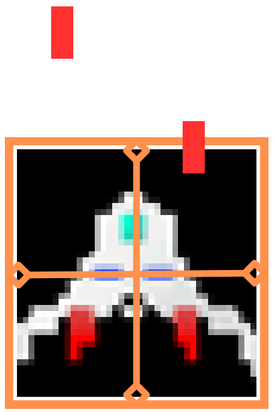
기능 구현

피격 구현

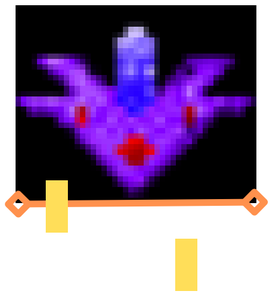
```
ArrayList galagArmoArray = new ArrayList(); // 모든 갤러그 탄알을 관리하는 리스트
ArrayList enemyArmoArray = new ArrayList(); // 모든 적 탄알을 관리하는 리스트
```

```
// 갤러그의 히트박스에 적 객체의 탄알이 만나면 갤러그가 피격된 것으로 판단
if ((tmp.enemy_armoX >= (galagX - galag.Width / 2) + 30 && tmp.enemy_armoX <= (galagX + galag.Width / 2) + 22) && (tmp.enemy_armoY > galagY - galag.Height / 2) && (tmp.enemy_armoY < galagY + galag.Height / 2))
```

```
// 적 객체의 히트박스에 갤러그 탄알이 만나면 적 객체를 피격된 것으로 판단
if (tmp.armoY < enemy.enemyY && (enemy.enemyX - (enemy.getWidth() / 2) + 30 <= tmp.armoX && tmp.armoX <= (enemy.enemyX + enemy.getWidth() / 2) + 22))
```



[갤러그 피격 범위]



[적 피격 범위]

- 생성되는 적 탄알과 갤러그 탄알은 배열에 저장됩니다
- 객체와 탄알 간의 거리를 계산하여, 서로 오버랩 되었는지 판단하고 피격을 구현합니다.
- 플레이어인 갤러그의 경우 모든 적 탄알과의 거리를 계산하는 방식으로 피격이 구현됩니다
- 일반 적과 보스의 경우 모든 갤러그 탄알과의 거리를 계산하는 방식으로 피격이 구현됩니다

아이템 생성 및 획득 구현

```
ArrayList boomArray = new ArrayList(); // 모든 폭탄 객체를 관리하는 리스트
ArrayList heartArray = new ArrayList(); // 모든 하트 객체를 관리하는 리스트
```

- **모든 아이템은 생성된 각 아이템을 담는 배열을 통해 관리합니다**

```
if (boomArray.Count != 0) // 생성된 폭탄이 있는 경우
{
    for (int i = 0; i < boomArray.Count; i++) // 생성된 폭탄을 갤러그가 획득했는지를 판단하는 반복문
    {
        Boom tmp = boomArray[i] as Boom;
        if (tmp.LeftOrRight == 0)
            tmp.boomX += 7;// 생성된 폭탄의 시작 x의 위치가 좌측면인 경우, 오른쪽으로 폭탄의 이동을 위한 x좌표값을 설정
        else
            tmp.boomX -= 7;// 생성된 폭탄의 시작 x의 위치가 우측면인 경우, 왼쪽으로 폭탄의 이동을 위한 x좌표값을 설정
        tmp.boomY += 7; // 폭탄의 이동을 위한 y좌표값을 설정
        tmp.setLocation(); // 실제 폭탄의 이동을 위한 좌표값 설정

        if (tmp.boomY > ClientSize.Height - 100) // 폭탄이 일정 범위 아래로 내려가는 경우 폭탄이 사라지게 한다
        {
            tmp.setInvisible();
            boomArray.RemoveAt(i);
        }

        // 폭탄의 히트박스와 갤러그가 겹치게 되면 폭탄을 먹은 것으로 판단한다
        else if (tmp.boomX - tmp.getWidth() / 2 <= galagX && galagX <= tmp.boomX + tmp.getWidth() / 2 && tmp.boomY - tmp.getHeight() / 2 <= galagY && galagY <= tmp.boomY + tmp.getHeight() / 2)
        {
            if (galagBoom < 3) // 폭탄 획득 시 갤러그의 폭탄 카운트를 증가시킨다. 폭탄은 최대 3번까지만 획득할 수 있도록 한다
                galagBoom++;
            boomStatus.Image = Image.FromFile("C# 리소스파일\\boombar" + galagBoom.ToString() + ".png"); // 획득한 폭탄의 갯수에 따라 폭탄 획득 상태 표시를 업데이트
            tmp.setInvisible();
            boomArray.RemoveAt(i); // 갤러그와 닿은 폭탄은 획득된 것이므로 화면에서 삭제한다
        }
    }
}
```

- 타이머를 활용하여, 일정한 간격으로 아이템이 생성되도록 구현했습니다
- 생성된 아이템은, 대각선 아래로 이동하며, 아이템이 획득되지 않고 일정 구간 아래로 내려가면 아이템은 사라집니다
- 피격 기능과 마찬가지로 갤러그와 아이템 간의 거리 계산을 통해, 아이템 획득을 구현했습니다.
- 아이템 갯수에 따라 이미지를 전환하여 아이템 위젯을 업데이트합니다

[폭탄과 하트 생성 및 획득 코드 구성 동일]

일반 적 생성

```
ArrayList enemyArray = new ArrayList(); // 모든 적 객체를 관리하는 리스트
```

```
private void Timer_Tick3(object sender, EventArgs e) // 적 객체를 생성하는 타이머3
{
    Random enemyAppear = new Random();
    int n = enemyAppear.Next(0, 100); // 랜덤변수를 통해서 일정 확률로 적 객체 생성을 수행한다

    if (n < 100) // 적 객체 생성 시
    {
        // 새로운 적 객체 생성을 위한 랜덤한 x,y 좌표를 얻는다
        Random enemyX = new Random();
        Random enemyY = new Random();
        int x = enemyX.Next(10, ClientSize.Width - 50);
        int y = enemyY.Next(0, 200);

        if (enemyArray.Count == 0) // 현재 적 객체가 하나도 없다면 얻은 x,y 좌표를 가진 적 객체를 바로 생성한다
            enemyArray.Add(new Enemy(x, y, this));

        else if (enemyArray.Count < 6) // 현재 적 객체가 이미 있는 경우
        {
            // 이미 생성되어 있는 적 객체들을 확인하여 새로 생성하는 적 객체가 기존 적 객체들과 x축으로 겹치지 않는 경우만 새로운 적 객체를 생성하도록 한다
            foreach (Enemy tmp in enemyArray)
            {
                int x1 = tmp.enemyX - tmp.getWidth() / 2;
                int x2 = tmp.enemyX + tmp.getWidth() / 2;
                if (((x - tmp.getWidth() / 2) >= x1 && (x - tmp.getWidth() / 2) <= x2) || ((x + tmp.getWidth() / 2) >= x1 && (x + tmp.getWidth() / 2) <= x2))
                    return; // 기존 적 객체의 x좌표와 새로운 적 객체의 x좌표가 겹치는 경우 새로운 적 객체를 생성하지 않고 return
            }
            enemyArray.Add(new Enemy(x, y, this)); // 기존 적 객체의 x좌표와 새로운 적 객체의 x좌표가 겹치지 않는 경우 새로운 적 객체를 생성
        }
    }
}
```

- 생성된 모든 일반 적 객체는 배열을 통해 관리됩니다
- 타이머를 활용하여, 일정 시간마다 적이 생성되도록 구현했습니다
- 적 생성 시, 적 탄알 발사를 위한 타이머를 활성화하여, 일정 간격으로 탄알을 발사합니다
- 적 이미지가 겹치는 문제를 해결하기 위해서 적들이 랜덤한 위치에 등장하되, x축 상으로는 겹치지 않도록 구현했습니다.

일반 적 생성

```
Random rand = new Random();
int enemy_idx;
if (enemyArray.Count != 0) // 적 객체가 존재하는 경우
{
    int enemyArmo_appear = rand.Next(0, 1000); // 일정 확률로 적 탄알을 생성한다
    if (enemyArmo_appear < 300) // 적 탄알을 생성되는 경우
    {
        enemy_idx = rand.Next(0, enemyArray.Count); // 현재 존재하는 적 객체들 중에서 선택하여
        Enemy enemy = enemyArray[enemy_idx] as Enemy;
        int enemyX = enemy.enemyX;
        int enemyY = enemy.enemyY;
        enemyArmoArray.Add(new EnemyArmo(enemyX, enemyY, this)); // 선택된 적 객체가 적 탄알을 발사하도록 한다.
    }
}

for (int i = 0; i < enemyArmoArray.Count; i++) // 모든 적 객체의 탄알을 확인해 갤러그가 적 객체의 탄알에 피격되었는지 안되었는지를 판단하는 반복문
{
    int armo_speed;
    EnemyArmo tmp = enemyArmoArray[i] as EnemyArmo;

    armo_speed = rand.Next(5, 20);
    tmp.enemy_armoY += armo_speed; // 적 객체의 탄알의 이동을 위한 좌표값 설정. 이동속도는 일정 속도 랜덤값을 이용하여 랜덤한 속도를 갖는다

    if (tmp.enemy_armoY > ClientSize.Height-70) // 적 객체의 탄알이 화면 아래 부분 일정 구간을 넘어가면 삭제하도록 한다
    {
        tmp.setInvisible();
        enemyArmoArray.RemoveAt(i);
    }
    else
    {
        // 갤러그의 히트박스에 적 객체의 탄알이 만나면 갤러그가 피격된 것으로 판단
        if ((tmp.enemy_armoX >= (galagX - galag.Width / 2) + 30 && tmp.enemy_armoX <= (galagX + galag.Width / 2) + 22) && (tmp.enemy_armoY > galagY - galag.Height / 2) && (tmp.enemy_armoY < galagY + galag.Height / 2))
```

- 적 생성 시, 적 탄알 발사를 위한 타이머를 활성화합니다.
- 현재 존재하는 일반 적 중에서 랜덤하게 하나를 선택하고, 선택한 적이 랜덤한 탄알 속도를 갖는 적 탄알을 발사하게 한다.

일반 적 생성

```
        {
            tmp.setInvisible();
            enemyArmoArray.RemoveAt(i); // 갤러그와 피격된 적 객체의 탄알은 삭제
            galag.Image = Image.FromFile("C# 리소스파일\\galagHit.gif"); // 갤러그의 피격 시 이펙트 발생
            galagFlag = 1; // 현재 피격당한 이펙트를 보여주었음을 알려줌

            galagLife--;
            if (galagLife == -1) // 모든 하트가 소모된 경우 게임오버에 대해서 처리한다
            {
                /* 게임 오버되었기 때문에 실행하던 모든 타이머들을 중단시키고 화면 상에 남아있던 모든 요소들을 제거한다 */
                boomMakingTimer.Stop();
                enemyShootingTimer.Stop();
                galagShootingTimer.Stop();
                enemyMakingTimer.Stop();
                heartMakingTimer.Stop();
                scoreTimer.Stop();
                this.Controls.Clear();
                gameOver(); //게임 오버 시 화면을 구성한다
            }
            else if (galagLife >= 0) //갤러그 체력이 남아있는 경우 피격 시 하트가 1개씩 감소
                heartStatus.Image = Image.FromFile("C# 리소스파일\\HeartBar" + galagLife.ToString() + ".png"); // 피격 시 감소한 하트 수에 따라 현재 하트 상태 업데이트
        }
    }
    tmp.moveEnemyArmo(); // 실질적 적 객체의 탄알 이동을 수행
}
```

보스 : 공격 패턴 구현

```
Random rand = new Random();
int bossArmoAppear= rand.Next(0, 1000); // 랜덤변수를 통해 보스 객체의 공격 확률을 설정
if (bossArmoAppear < 120) // 보스 객체가 공격을 수행하는 경우
{
    Random pattern = new Random();
    int bossPattern = rand.Next(0, 100); // 랜덤변수를 통해 보스 객체의 공격 패턴을 선택하여 공격을 수행한다.
    if (bossPattern < 25) // 25% 확률로는 방사형으로 움직이는 탄알을 발사하는 패턴
    {
        enemyArmoArray.Add(new EnemyArmo(150, 200, this, -10, 17));
        enemyArmoArray.Add(new EnemyArmo(150, 200, this, -5, 17));
        enemyArmoArray.Add(new EnemyArmo(150, 200, this, 0, 17));
        enemyArmoArray.Add(new EnemyArmo(150, 200, this, 5, 17));
        enemyArmoArray.Add(new EnemyArmo(150, 200, this, 10, 17));
    }
    else if (bossPattern < 60) // 35% 확률로는 우측에서 생성되고 직진하며 움직이는 탄알을 발사하는 패턴
    {
        enemyArmoArray.Add(new EnemyArmo(240, 200, this, 0, 15));
        enemyArmoArray.Add(new EnemyArmo(180, 200, this, 0, 15));
        enemyArmoArray.Add(new EnemyArmo(120, 200, this, 0, 15));
        enemyArmoArray.Add(new EnemyArmo(60, 200, this, 0, 15));
        enemyArmoArray.Add(new EnemyArmo(0, 200, this, 0, 15));
    }
    else if(bossPattern<85) // 35% 확률로는 좌측에서 생성되고 직진하며 움직이는 탄알을 발사하는 패턴
    {
        enemyArmoArray.Add(new EnemyArmo(60, 200, this, 0, 15));
        enemyArmoArray.Add(new EnemyArmo(120, 200, this, 0, 15));
        enemyArmoArray.Add(new EnemyArmo(180, 200, this, 0, 15));
        enemyArmoArray.Add(new EnemyArmo(240, 200, this, 0, 15));
        enemyArmoArray.Add(new EnemyArmo(300, 200, this, 0, 15));
    }
    else // 15% 확률로는 더 빠르게 방사형으로 움직이는 탄알을 발사하는 패턴
    {
        enemyArmoArray.Add(new EnemyArmo(150, 200, this, -15, 20));
        enemyArmoArray.Add(new EnemyArmo(150, 200, this, -10, 20));
        enemyArmoArray.Add(new EnemyArmo(150, 200, this, -5, 20));
        enemyArmoArray.Add(new EnemyArmo(150, 200, this, 0, 20));
        enemyArmoArray.Add(new EnemyArmo(150, 200, this, 5, 20));
        enemyArmoArray.Add(new EnemyArmo(150, 200, this, 10, 20));
        enemyArmoArray.Add(new EnemyArmo(150, 200, this, 15, 20));
```

- 보스의 경우 추가적으로 moveX,Y 값을 갖는 탄알을 생성하여, 단순히 직선으로만 발사되는 탄알이 아닌 다양한 패턴으로 공격이 구현되도록 했습니다.

- 타이머를 활용하여, 일정 시간마다 정해진 확률에 따라, 4가지 패턴 중 한 가지 공격 패턴을 선택하여 공격합니다.

보스 : HP 바 구현

```
if(bossAppear == true) // 보스 객체가 등장해 있다면 보스 이미지를 화면에 띄워주고 보스 객체의 피격횟수에 따른 보스 객체의 체력바를 업데이트
{
    bossHP.Parent = this;
    if (bossHitCnt <=5)
        bossHP.Image = Image.FromFile("C# 리소스파일\\bossHP5.png");
    else if(bossHitCnt<=10)
        bossHP.Image = Image.FromFile("C# 리소스파일\\bossHP10.png");
    else if (bossHitCnt <= 15)
        bossHP.Image = Image.FromFile("C# 리소스파일\\bossHP15.png");
    else if (bossHitCnt <= 20)
        bossHP.Image = Image.FromFile("C# 리소스파일\\bossHP20.png");
    else if (bossHitCnt <= 25)
        bossHP.Image = Image.FromFile("C# 리소스파일\\bossHP25.png");
    else if (bossHitCnt <= 30)
        bossHP.Image = Image.FromFile("C# 리소스파일\\bossHP30.png");
    else if (bossHitCnt <= 35)
        bossHP.Image = Image.FromFile("C# 리소스파일\\bossHP35.png");
    else if (bossHitCnt <= 40)
        bossHP.Image = Image.FromFile("C# 리소스파일\\bossHP40.png");
    else if (bossHitCnt <= 45)
        bossHP.Image = Image.FromFile("C# 리소스파일\\bossHP45.png");
    else if (bossHitCnt <= 50)
        bossHP.Image = Image.FromFile("C# 리소스파일\\bossHP50.png");
}
```

- 보스의 체력인 bossHitCnt 를 체크하여 적절한 HP바 이미지로 전환하여 구현했습니다.

04

리소스 파일 제작

리소스 파일 제작

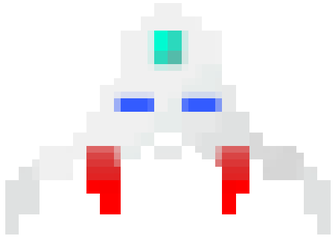
- 필요한 리소스 파일들은 도트 이미지 파일을 제작할 수 있는 Piksel-0.14.0 프로그램을 통해 직접 제작하였습니다.

- 단순히 정지된 이미지보다는 프레임 별로 이미지를 만들고 이를 연결하여 움직이는 gif 이미지 파일을 제작해 사용함으로서 조금은 더 역동적인 게임이 될 수 있도록 하였습니다.

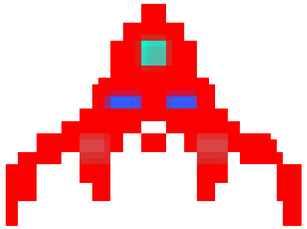
- 프로젝트 파일 내부에 ‘C# 리소스 파일’ 폴더에서 제작한 gif파일 및 gif파일의 프레임 이미지를 담고 있는 png파일을 확인하실 수 있습니다.

리소스 파일 제작

[이미지 이름을 클릭하여 GIF 파일을 확인할 수 있습니다]



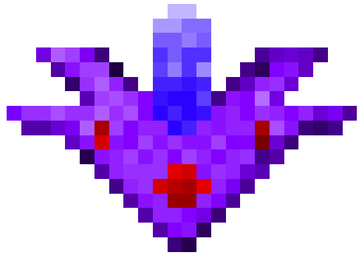
[galag]



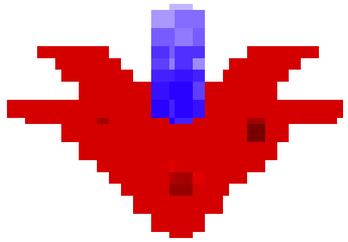
[galagHit]

[boom]

[heart]



[enemy]



[enemyHit]

[boss]

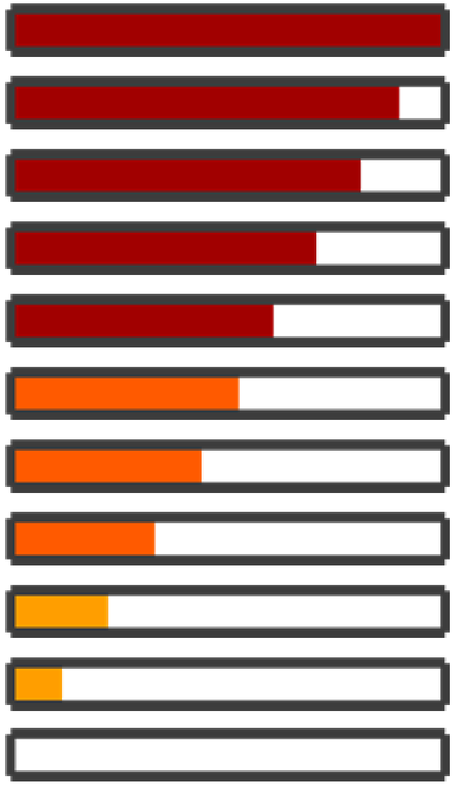
리소스 파일 제작

[boombar0~3]

[HeartBar0~3]

[galagArmo]

[enemyArmo]



[bossHP0~50]