Lab#3 Invader

최대원

배경

- 작은 별이 반짝임, 임의의 위치와 알파값을 가짐
- 별들이 아래로 떨어짐(유저가 우주공간을 나아가는 착시 제공)
- 별의 알파값이 낮을수록 속도가 느림

Form1

- 매 타이머마다 Game.Update() 실행
- 적 애니메이션은 실제 타이머보다 느리다(Frame Skip)
- Paint Eventhandler 발생
- 키 입력 받음

Game

- Paint 받아서 모든 객체 그려줌
  +별 반짝이게 해줌
- 키 입력에 따라 플레이어 이동시켜줌
- 적의 이동과 사격 계산하기
- 충돌 처리하기
- 플레이어 사망 판단하기
- 화면 밖으로 나간 총알 삭제하기
- 적이 전멸 한 경우 다음 스테이지 세팅
- 스테이지가 진행되고 적이 적어질수록 적이 빨리 움직임
- 일정 간격으로 적이 총을 쏘게 만듦
- 목숨을 다 쓰거나 적이 최하단에 도착하면 Game Over
- Game Over 상태에서 Q를 누르면 게임 종료 S를 누르면 처음부터 다시 시작
- 게임 화면 경계값(Ractangle) 보유

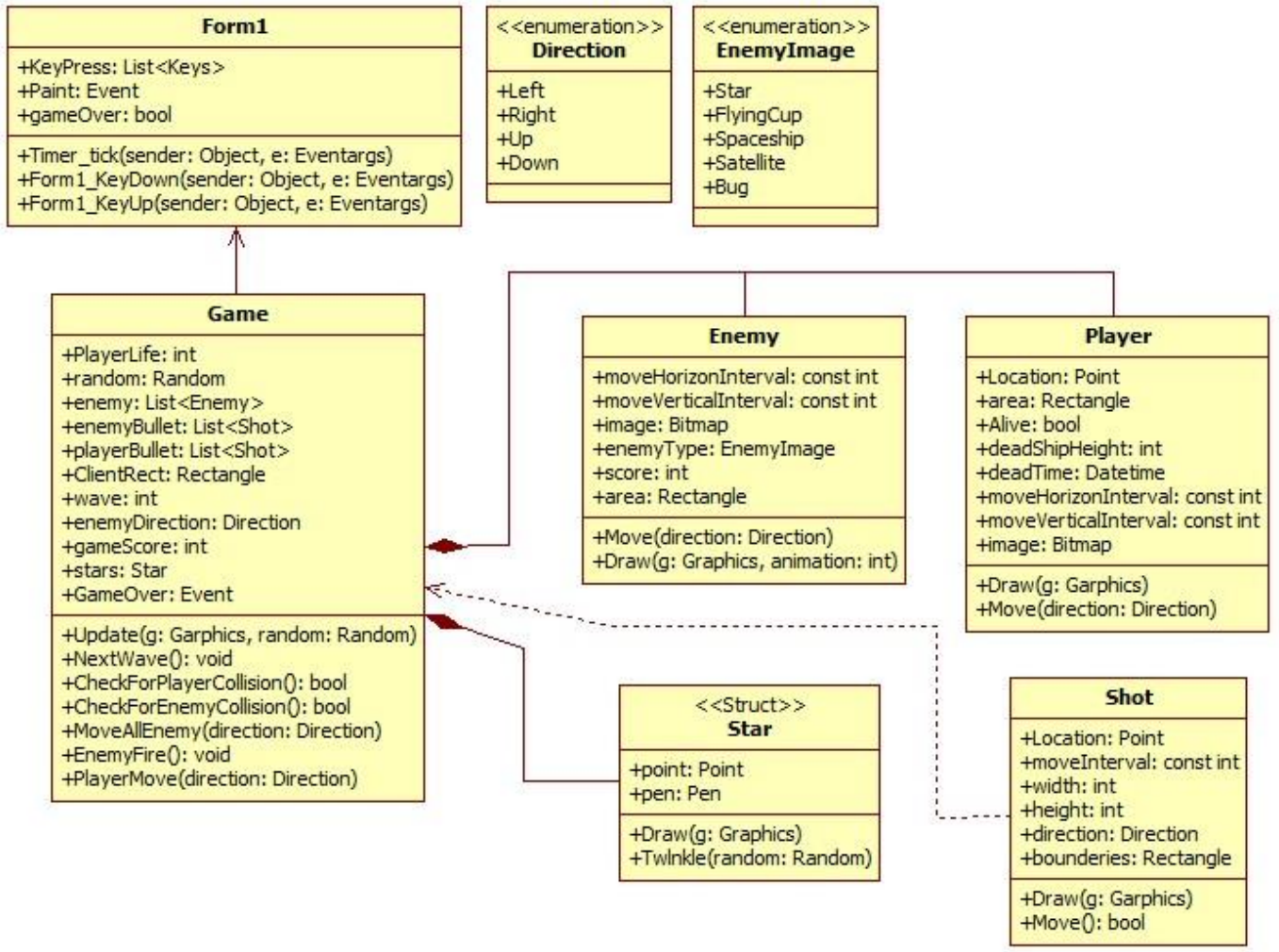
Enemy

- 30명의 적
- 맨 앞줄의 적만 사격을 함
- 적의 ㄹ자 이동, 한쪽 벽에 다다르면 다같이 아래로 한번 이동 후 반대편 벽으로 이동
- 화면 하단까지 적이 도착하면 게임 오버
- 앞의 적 한 줄이 사라지면 이동 속도가 증가
- 파괴하는 적의 생김새에 따라 얻는 점수가 다름
- 적을 다 파괴하면 다음 스테이지로 이동
- 적의 총알은 한 화면에 2개 까지만 있을 수 있다.
- 그러나 스테이지가 넘어갈 때 마다 총알 제한이 1개씩 늘어난다

Player

- 좌우 키로 이동, z키로 사격
  누르고 있으면 계속 이동하거나 사격함
- 플레이어의 총알은 한 화면에 5개까지 있을 수 있고
- 1초에 3발을 사격 가능(쿨타임)
- 목숨은 총 3개
- 죽으면 부활할 때 까지 적이 일시정지(총 3초)
- 죽을 때 애니메이션(점점 얇아지게)

UML Diagram


Form1
+KeyPress: List<Keys>
+Paint: Event
+gameOver: bool
+Timer_tick(sender: Object, e: Eventargs)
+Form1_KeyDown(sender: Object, e: Eventargs)
+Form1_KeyUp(sender: Object, e: Eventargs)
<<enumeration>>
Direction
+Left
+Right
+Up
+Down
<<enumeration>>
EnemyImage
+Star
+FlyingCup
+Spaceship
+Satellite
+Bug
Game
+PlayerLife: int
+random: Random
+enemy: List<Enemy>
+enemyBullet: List<Shot>
+playerBullet: List<Shot>
+ClientRect: Rectangle
+wave: int
+enemyDirection: Direction
+gameScore: int
+stars: Star
+GameOver: Event
+Update(g: Garphics, random: Random)
+NextWave(): void
+CheckForPlayerCollision(): bool
+CheckForEnemyCollision(): bool
+MoveAllEnemy(direction: Direction)
+EnemyFire(): void
+PlayerMove(direction: Direction)
Enemy
+moveHorizonInterval: const int
+moveVerticalInterval: const int
+image: Bitmap
+enemyType: EnemyImage
+score: int
+area: Rectangle
+Move(direction: Direction)
+Draw(g: Graphics, animation: int)
Player
+Location: Point
+area: Rectangle
+Alive: bool
+deadShipHeight: int
+deadTime: Datetime
+moveHorizonInterval: const int
+moveVerticalInterval: const int
+image: Bitmap
+Draw(g: Garphics)
+Move(direction: Direction)
<<Struct>>
Star
+point: Point
+pen: Pen
+Draw(g: Graphics)
+Twlnkle(random: Random)
Shot
+Location: Point
+moveInterval: const int
+width: int
+height: int
+direction: Direction
+bounderies: Rectangle
+Draw(g: Garphics)
+Move(): bool

달라진 점

- 한 화면에 존재 가능한 플레이어의 총알 최대치 = 6 + (wave * 2);
- 한 화면에 존재 가능한 적의 총알 최대치 = 5 + wave;
- 적이 죽으면 그 자리에 폭파 이펙트와 점수 글자를 남김
- 레벨이 올라갈 수록 난이도 증가

```
//매 타이머마다 실행==========================================================
public void Update(Random random)...

//이하 나머지 메소드==========================================================
private void Game_GameOver()...//게임 오버 이벤트 발생시 호출 할 함수
public void DrawDieEffect(Graphics g)...//적 죽을 때 나올 효과
public void CheckForPlayerByEnemyBullet(Graphics g)...
public void TotalScoreDraw(Graphics g)...
public void CheckForEnemyByPlayerBullet(Graphics g)...//적이 죽을 때 각종 처리
public void KillScoreDraw(Graphics g)...
public void PlayerBulletDraw(Graphics g) ...
public void EnemyBulletDraw(Graphics g) ...
public void EnemyDraw(Graphics g)...
public void drawPlayerLife(Graphics g)...
public void NextWave()...
public void MoveAllEnemy(Direction direction)//적을 이동시킴...
private void MovingAllEnemy(int dir)...//적이 움직일 방향지정
public bool CheckForPlayerCollision()...
private void CheckForMoveAllEnemy()...
public void EnemyShoot()...//적이 총을 쏘는 함수
public void EnemyFire()...//누가 총을 쏠지 구하는 함수
public void PlayerFire()...
public Enemy CheckForEnemyCollision(Point point)...//총에 맞은 적이 누구인지 반환
public void playerMove(Direction direction)...
public void Twinkle(Graphics g)...//반짝반짝
```

게임의 핵심 함수들

타이머 없이 카운트다운 사용하기

```
public EnemyDie(Point point, int score)
{
    this.point = point;
    dateTimeNow = DateTime.Now;
    dateTimeAfterSecond = dateTimeNow.AddSeconds(2);
    //Console.WriteLine(dateTimeAfterSecond);
    this.score = score;
}

public bool die()
{
    if (dateTimeAfterSecond <= DateTime.Now)
    {
        return true;
    }
    return false;
}
```

적 이동

1. 방향을 구하고
2. 움직인다.

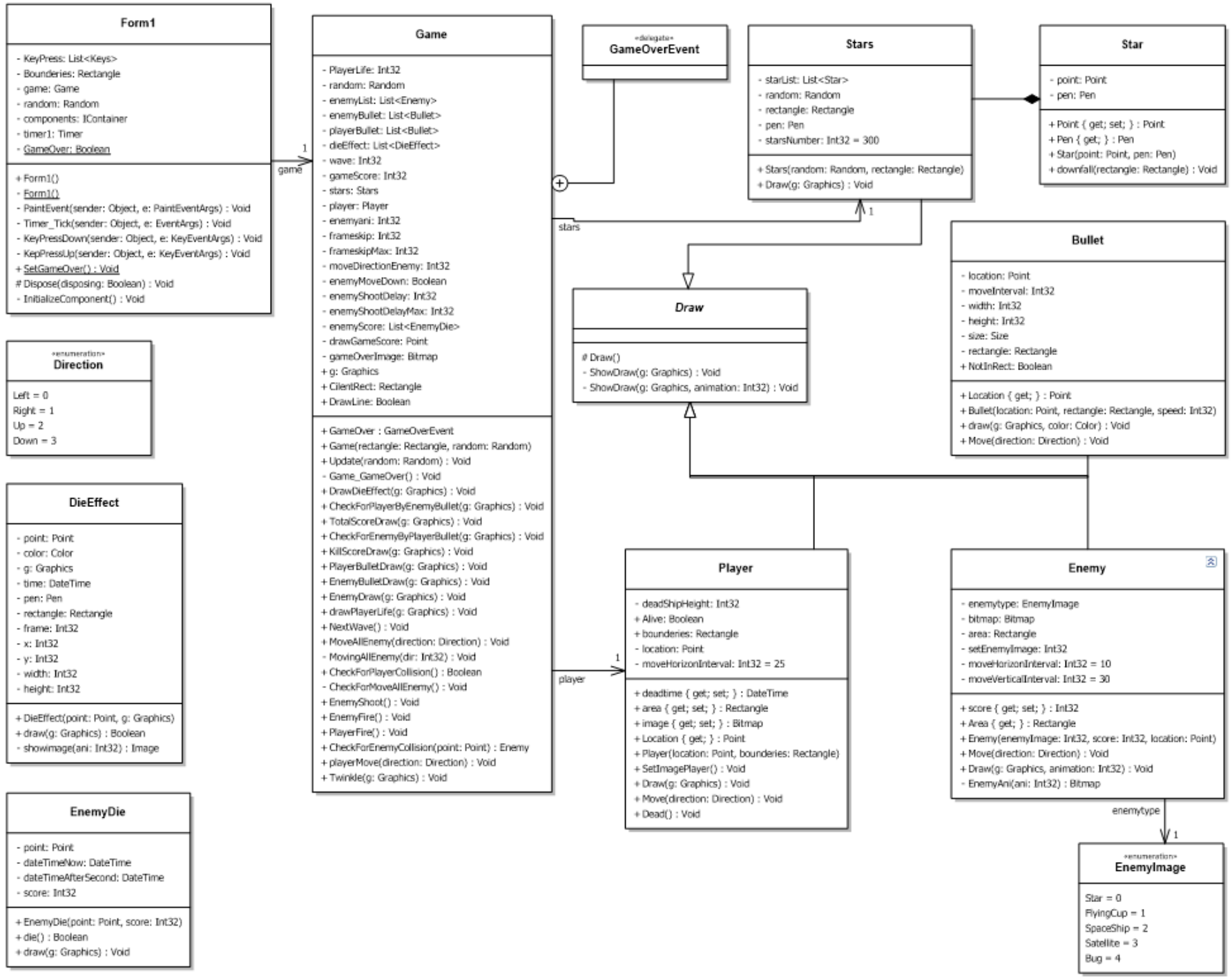
```
private void CheckForMoveAllEnemy()
{
    if (enemyList.Count > 0)
    {
        foreach (Enemy item in enemyList)
        {
            if(item.Area.X < CilentRect.X + 50)
            {
                moveDirectionEnemy = 1;
                enemyMoveDown = true;
            }
            if (item.Area.X > CilentRect.Width - 100)
            {
                moveDirectionEnemy = -1;
                enemyMoveDown = true;
            }
        }
        //Console.WriteLine("moveDirectionEnemy = " +
    }
}//적이 이동 할 방향을 정해주는 함수
```

총알의 이동

```
public Bullet(Point location, Rectangle rectangle, int speed)
{
    this.location = location;
    this.rectangle = rectangle;
    moveInterval = speed;
    NotInRect = false;
    size = new Size(width, height);
}

public void draw(Graphics g, Color color)
{
    using (SolidBrush solidBrush = new SolidBrush(color))
    {
        g.FillRectangle(solidBrush, new Rectangle(Location, size));
    }
}

public void Move(Direction direction)
{
    if (direction == Direction.Up)
    {
        location.Y -= moveInterval;
    }
    if (direction == Direction.Down)
    {
        location.Y += moveInterval;
    }
    if (!rectangle.Contains(location)) NotInRect = true;
}
```


Form1
- KeyPress: List<Keys>
- Bounderies: Rectangle
- game: Game
- random: Random
- components: IContainer
- timer1: Timer
- GameOver: Boolean
+ Form1()
- Form1()
- PaintEvent(sender: Object, e: PaintEventArgs) : Void
- Timer_Tick(sender: Object, e: EventArgs) : Void
- KeyPressDown(sender: Object, e: KeyEventArgs) : Void
- KepPressUp(sender: Object, e: KeyEventArgs) : Void
+ SetGameOver() : Void
# Dispose(disposing: Boolean) : Void
- InitializeComponent() : Void
1
game
Game
- PlayerLife: Int32
- random: Random
- enemyList: List<Enemy>
- enemyBullet: List<Bullet>
- playerBullet: List<Bullet>
- dieEffect: List<DieEffect>
- wave: Int32
- gameScore: Int32
- stars: Stars
- player: Player
- enemyani: Int32
- frameskip: Int32
- frameskipMax: Int32
- moveDirectionEnemy: Int32
- enemyMoveDown: Boolean
- enemyShootDelay: Int32
- enemyShootDelayMax: Int32
- enemyScore: List<EnemyDie>
- drawGameScore: Point
- gameOverImage: Bitmap
+ g: Graphics
+ ClientRect: Rectangle
+ DrawLine: Boolean
+ GameOver : GameOverEvent
+ Game(rectangle: Rectangle, random: Random)
+ Update(random: Random) : Void
- Game_GameOver() : Void
+ DrawDieEffect(g: Graphics) : Void
+ CheckForPlayerByEnemyBullet(g: Graphics) : Void
+ TotalScoreDraw(g: Graphics) : Void
+ CheckForEnemyByPlayerBullet(g: Graphics) : Void
+ KillScoreDraw(g: Graphics) : Void
+ PlayerBulletDraw(g: Graphics) : Void
+ EnemyBulletDraw(g: Graphics) : Void
+ EnemyDraw(g: Graphics) : Void
+ drawPlayerLife(g: Graphics) : Void
+ NextWave() : Void
+ MoveAllEnemy(direction: Direction) : Void
- MovingAllEnemy(dir: Int32) : Void
+ CheckForPlayerCollision() : Boolean
- CheckForMoveAllEnemy() : Void
+ EnemyShoot() : Void
+ EnemyFire() : Void
+ PlayerFire() : Void
+ CheckForEnemyCollision(point: Point) : Enemy
+ playerMove(direction: Direction) : Void
+ Twinkle(g: Graphics) : Void
«delegate»
GameOverEvent
Stars
- starList: List<Star>
- random: Random
- rectangle: Rectangle
- pen: Pen
- starsNumber: Int32 = 300
+ Stars(random: Random, rectangle: Rectangle)
+ Draw(g: Graphics) : Void
Star
- point: Point
- pen: Pen
+ Point { get; set; } : Point
+ Pen { get; } : Pen
+ Star(point: Point, pen: Pen)
+ downfall(rectangle: Rectangle) : Void
stars
1
Draw
Draw()
- ShowDraw(g: Graphics) : Void
- ShowDraw(g: Graphics, animation: Int32) : Void
Bullet
- location: Point
- moveInterval: Int32
- width: Int32
- height: Int32
- size: Size
- rectangle: Rectangle
+ NotInRect: Boolean
+ Location { get; } : Point
+ Bullet(location: Point, rectangle: Rectangle, speed: Int32)
+ draw(g: Graphics, color: Color) : Void
+ Move(direction: Direction) : Void
«enumeration»
Direction
Left = 0
Right = 1
Up = 2
Down = 3
DieEffect
- point: Point
- color: Color
- g: Graphics
- time: DateTime
- pen: Pen
- rectangle: Rectangle
- frame: Int32
- x: Int32
- y: Int32
- width: Int32
- height: Int32
+ DieEffect(point: Point, g: Graphics)
+ draw(g: Graphics) : Boolean
- showimage(ani: Int32) : Image
Player
- deadShipHeight: Int32
+ Alive: Boolean
+ bounderies: Rectangle
- location: Point
- moveHorizonInterval: Int32 = 25
+ deadtime { get; set; } : DateTime
+ area { get; set; } : Rectangle
+ image { get; set; } : Bitmap
+ Location { get; } : Point
+ Player(location: Point, bounderies: Rectangle)
+ SetImagePlayer() : Void
+ Draw(g: Graphics) : Void
+ Move(direction: Direction) : Void
+ Dead() : Void
1
player
Enemy
- enemytype: EnemyImage
- bitmap: Bitmap
- area: Rectangle
- setEnemyImage: Int32
- moveHorizonInterval: Int32 = 10
- moveVerticalInterval: Int32 = 30
+ score { get; set; } : Int32
+ Area { get; } : Rectangle
+ Enemy(enemyImage: Int32, score: Int32, location: Point)
+ Move(direction: Direction) : Void
+ Draw(g: Graphics, animation: Int32) : Void
- EnemyAni(ani: Int32) : Bitmap
enemytype
1
EnemyDie
- point: Point
- dateTimeNow: DateTime
- dateTimeAfterSecond: DateTime
- score: Int32
+ EnemyDie(point: Point, score: Int32)
+ die() : Boolean
+ draw(g: Graphics) : Void
«enumeration»
EnemyImage
Star = 0
FlyingCup = 1
SpaceShip = 2
Satellite = 3
Bug = 4