

Code Review: Invaders

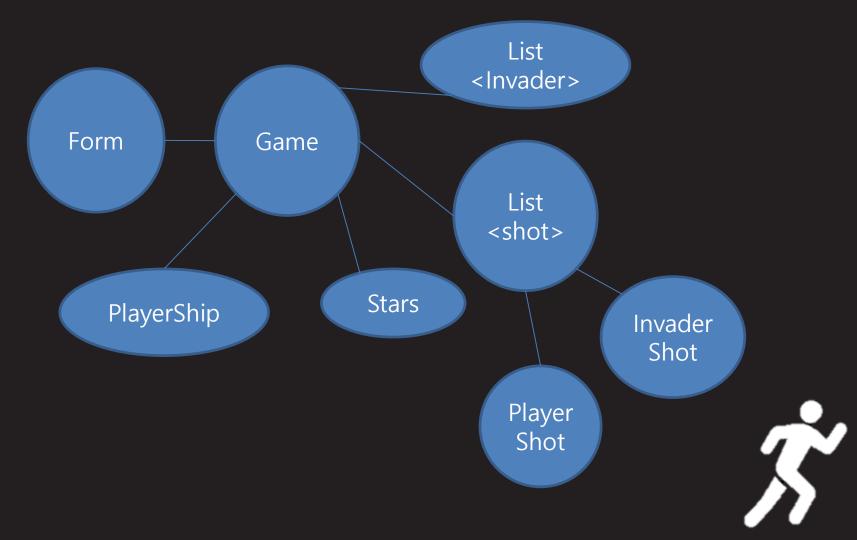
김승환

Contents

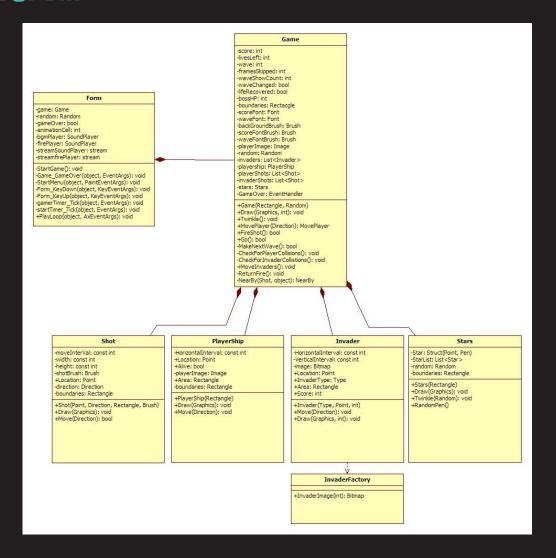
- 설계 구조 및 클래스 다이어그램
- 목표 및 구현
- 코드 분석
- 개선할 점



설계 구조



Class Diagram





Class Diagram

Form

-game: Game
-random: Random
-gameOver: bool
-animationCell: int
-bgmPlayer: SoundPlayer
-firePlayer: SoundPlayer
-streamSoundPlayer: stream
-streamfirePlayer: stream

-StartGame(): void -Game_GameOver(object, EventArgs): void -StartMenu(object, PaintEventArgs): void -Form_KeyDown(object, KeyEventArgs): void -Form_KeyUp(object, KeyEventArgs): void -gamerTimer_Tick(object, EventArgs): void -startTimer_Tick(object, EventArgs): void +PlayLoop(object, AxEventArgs): void

Game

-score: int -livesLeft: int -wave: int -framesSkipped: int -waveShowCount: int -waveChanged: bool -lifeRecovered: bool -bossHP: int -boundaries: Rectacgle -scoreFont: Font -waveFont: Font

-backGroundBrush: Brush -scoreFontBrush: Brush -waveFontBrush: Brush -playerImage: Image -random: Random -invaders: List<Invader>

-playership: PlayerShip -playerShots: List<Shot> -invaderShots: List<Shot>

-stars: Stars

-GameOver: EventHandler

+Game(Rectangle, Random) +Draw(Graphics, Int): void

+Twinkle(): void

+MovePlayer(Direction): MovePlayer

+FireShot(): bool +Go(): bool

-MakeNextWave(): bool

-CheckForPlayerCollisions(): void -CheckForInvaderCollistions(): void

+MoveInvaders(): void -ReturnFire(): void

-NearBy(Shot, object): NearBy



Class Diagram

Form

-game: Game
-random: Random
-gameOver: bool
-animationCell: int
-bgmPlayer: SoundPlayer
-firePlayer: SoundPlayer
-streamSoundPlayer: stream
-streamfirePlayer: stream

-StartGame(): void -Game_GameOver(object, EventArgs): void -StartMenu(object, PaintEventArgs): void -Form_KeyDown(object, KeyEventArgs): void -Form_KeyUp(object, KeyEventArgs): void -gamerTimer_Tick(object, EventArgs): void -startTimer_Tick(object, EventArgs): void +PlayLoop(object, AxEventArgs): void

Game

-score: int -livesLeft: int -wave: int -framesSkipped: int -waveShowCount: int -waveChanged: bool -lifeRecovered: bool -bossHP: int -boundaries: Rectacgle -scoreFont: Font -waveFont: Font

-backGroundBrush: Brush -scoreFontBrush: Brush -waveFontBrush: Brush -playerImage: Image -random: Random -invaders: List<Invader>

-playership: PlayerShip -playerShots: List<Shot> -invaderShots: List<Shot>

-stars: Stars

-GameOver: EventHandler

+Game(Rectangle, Random) +Draw(Graphics, Int): void

+Twinkle(): void

+MovePlayer(Direction): MovePlayer

+FireShot(): bool +Go(): bool

-MakeNextWave(): bool

-CheckForPlayerCollisions(): void -CheckForInvaderCollistions(): void

+MoveInvaders(): void -ReturnFire(): void

-NearBy(Shot, object): NearBy



Class Diagram

Shot Player5hip Invader Stars -moveInterVal: const int -HorizontalInterval: const int -HorizontalInterval: const int -Star: Struct(Point, Pen) -width: const int +Location: Point -VerticalInterval: const int -StarList: List<Star> +Alive: bool -image: Bitmap -height: const int -random: Random -shotBrush: Brush -playerImage: Image +Location: Point -boundaries: Rectangle +Location: Point +InvaderType: Type +Area: Rectangle +Stars(Rectangle) -direction: Direction -boundaries: Rectangle +Area: Rectangle +Draw(Graphics): void -boundaries: Rectangle +Score: int +PlayerShip(Rectangle) +Twinkle(Random): void +Draw(Graphics): void +Invader(Type, Point, int) +Shot(Point, Direction, Rectangle, Brush) +RandomPen() +Draw(Graphics): void +Move(Direction): void +Move(Direction): void +Move(Direction): bool +Draw(Graphics, int): void InvaderFactory +InvaderImage(int): Bitmap



목표 및 구현

- 함수 모듈화 및 단순화
- 스토리텔링
- 변수, 메소드 작명법 준수
- 버그 최소화
- 커리큘럼 내용 리마인드(파일 입출력, 예외 처리, LINQ, 디자인 패턴)



목표 및 구현

- 게임 시작 화면
- 메인 게임 화면
- 키입력
- 게임 구성(총 9스테이지)
- 보스 스테이지 제작
- BGM, Effect Sound



코드 분석

- 게임세팅
- 게임 화면 그리기
- 배경음악, 효과음
- 충돌 체크, Invader 이동
- 총알 발사



코드 분석

■ Invader 객체 리스트에 집어넣기



코드 분석

■ Flyweight의 Factory Method

```
public static Bitmap getInvaderImage(Type InvaderType, int animationCell)
   switch (InvaderType)
       case Type, Bug:
            if (animationCell == 0) return Properties.Resources.bug1;
           else if (animationCell == 1) return Properties.Resources.bug2;
            else if (animationCell == 2) return Properties.Resources.bug3;
            else return Properties.Resources.bug4;
       case Type.Satellite:
            if (animationCell == 0) return Properties.Resources.satellite1
            else if (animationCell == 1) return Properties.Resources.satel
            else if (animationCell == 2) return Properties.Resources.satel
            else return Properties.Resources.satellite4;
        case Type.Saucer:
            if (animationCell == 0) return Properties.Resources.flyingsauc
           else if (animationCell == 1) return Properties.Resources.flyin
            else if (animationCell == 2) return Properties.Resources.flvin
            else return Properties.Resources.flyingsaucer4;
        case Type, Spaceship:
            if (animationCell == 0) return Properties.Resources.spaceship1
            else if (animationCell == 1) return Properties.Resources.space
            else if (animationCell == 2) return Properties.Resources.space
            else return Properties.Resources.spaceship4;
        case Type.Star:
            if (animationCell == 0) return Properties.Resources.star1;
            else if (animationCell == 1) return Properties.Resources.star2
            else if (animationCell == 2) return Properties.Resources.star3
            else return Properties.Resources.star4;
        case Type.Boss:
            return Properties.Resources.Boss;
```



개선할 점 및 아쉬운 점

- State Pattern을 사용하지 않아 조건문이 너무 많다
- 스토리텔링 기법이 남에게 와 닿지 않을 수 있다
- 시간을 너무 조금 투자했다(파일 입출력 미구현)
- 추상화를 사용하지 않았다(물체들의 기본 베이스가 되는 클래스)

