

# The Quest

# 계획

---

## ◎ 1일차

- 기획 / 구성 / 설계

## ◎ 2일차

- 기본 시스템
- 캐릭터, 몬스터 이동 및 공격

## ◎ 3일차

- 추가기능

# 1일차

---

## ◎ 기획

- HeadFirst C# - The Quest

## ◎ 구상

- 클래스별 해야할 작업들

## ◎ 설계

# 구상 - 클래스별 해야할 작업들

---

## ◎ Form

- **Game** 객체 생성
- 게임 화면 업데이트
- 이동 버튼
- 공격 버튼
- 아이템 장착

# 구상 - 클래스별 해야할 작업들

---

## ● Game

- 플레이어 객체 생성
- 적 목록 저장
- 레벨별 드랍되는 장비 저장(1개)
- 플레이어와 몬스터의 이동
- 플레이어의 공격과 몬스터의 이동
- 레벨 생성
  - 몬스터와 무기들의 위치를 랜덤지점에 위치
  - 가지고 있는 장비라면 필드에 생성하지 않기

# 구상 - 클래스별 해야할 작업들

---

- 인벤토리에 무엇이 있는지 확인
- 폼에서 클릭한 아이템을 플레이어에게 장착

# 구상 - 클래스별 해야할 작업들

---

## ◎ Mover ( 추상 )

- 각 객체의 현재 위치 저장
- 객체 사이의 거리 계산
- 이동
  - 던전의 크기 만큼 이동 제한 필요
  - 플레이어 혹은 몬스터는 이동함수의 재정의가 필요
    - 플레이어는 버튼 입력으로 이동
    - 몬스터는 설정 **AI**에 의한 이동

# 구상 - 클래스별 해야할 작업들

---

## ● Player

- 현재 **HP** 정보
- 장착한 아이템 정보
- 인벤토리에 무엇이 있는가
- 포션을 사용했을 때의 효과
- 장비 장착 메소드



# 구상 - 클래스별 해야할 작업들

---

## ◎ Enemy ( 추상 )

- 현재 **HP** 정보
- 이동 **AI**
  - 각 몬스터별 재정의

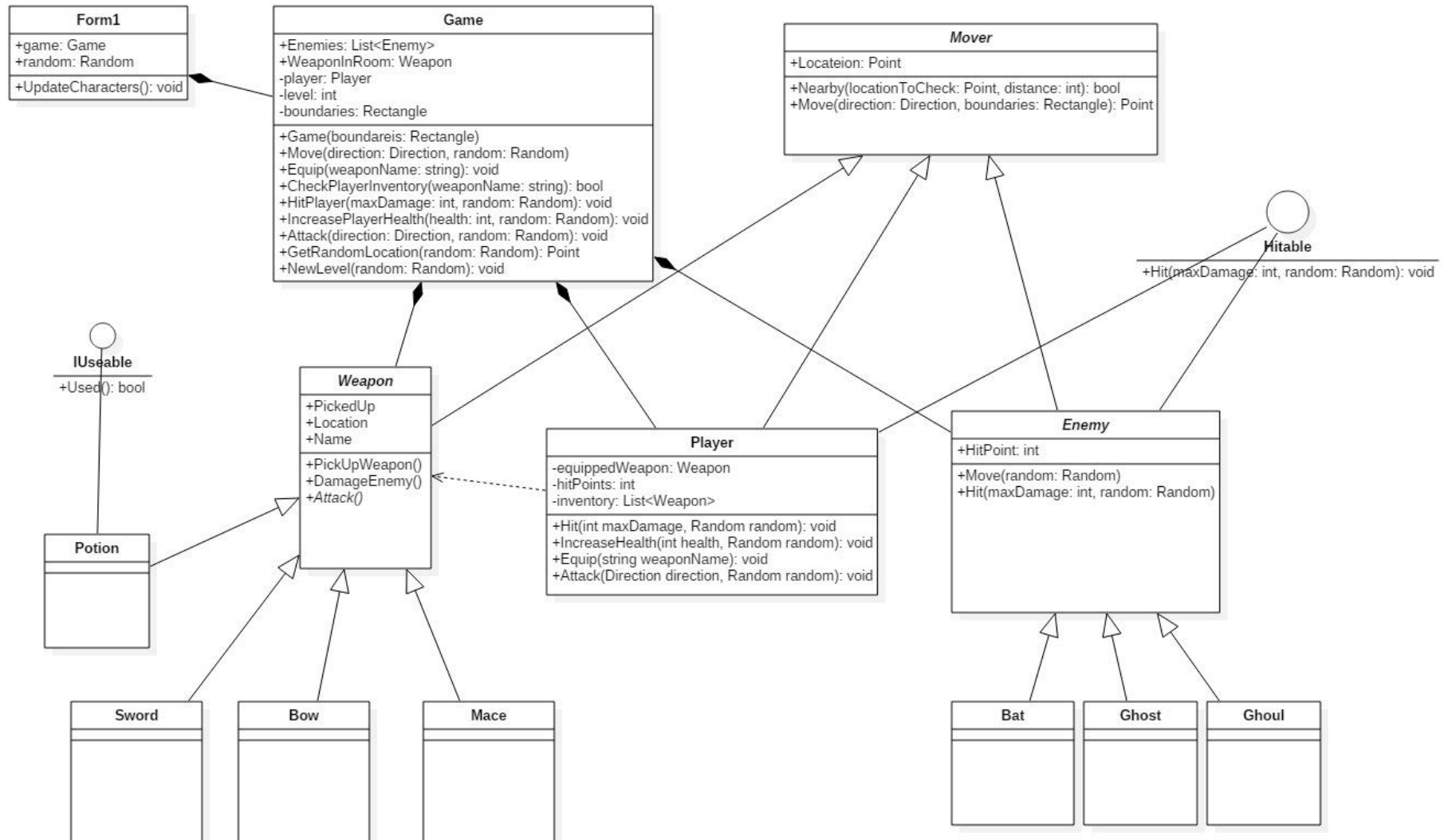
# 구상 - 클래스별 해야할 작업들

---

## ◎ Weapon ( 추상 )

- 드랍된 아이템들의 위치 저장
- 플레이어가 주운 아이템인지 판별
- 각 무기들의 이름
- 몬스터를 공격
  - 무기별 범위와 공격력 재정의

# 설계



## 2일차

---

- 기본 시스템
- 플레이어와 몬스터의 이동
- 플레이어와 몬스터의 공격

# 기본 시스템

---

- 방향을 저장하는 enum – Direction

```
public enum Direction
{
    Up = 0,
    Down = 2,
    Left = 3,
    Right = 1,
    Stop = 4,

}
```

# 기본 시스템

## ● 게임 화면 업데이트

- 플레이어 위치 갱신
- 몬스터 위치 갱신
- 몬스터, 아이템의 PictureBox 제어

```
private void enemyCheck()
{
    int enemiesShown = 0;
    foreach (Enemy enemy in game.Enemies)
    {
        if (enemy is Bat)
        {
            Bat_Pic.Location = enemy.Location;
            BatHP.Text = enemy.HitPoints.ToString();
            if (enemy.HitPoints > 0)
            {
                Bat_Pic.Visible = true;
                enemiesShown++;
            }
            else
            {
                enemiesShown--;
            }
        }
    }
}
```

생략

```
private void imageInvisible()
{
    //Enemy
    Bat_Pic.Visible = false;
    Ghost_Pic.Visible = false;
    Ghoul_Pic.Visible = false;
    Skeleton_Pic.Visible = false;

    //DropItem
    Sword_Drop_Pic.Visible = false;
    Mace_Drop_Pic.Visible = false;
    Bow_Drop_Pic.Visible = false;

    BluePotion_Drop_Pic.Visible = false;
    RedPotion_Drop_Pic.Visible = false;

    //Inventory
    Sword_Inven_Pic.Visible = false;
    Mace_Inven_Pic.Visible = false;
    Bow_Inven_Pic.Visible = false;

    RedPotion_Inven_Pic.Visible = false;
    BluePotion_Inven_Pic.Visible = false;
}
```

```
private void pickUpItem()
{
    Control weaponControl = null;
    switch (game.WeaponInRoom.Name)
    {
        case "Sword":
            weaponControl = Sword_Drop_Pic;
            break;
        case "Bow":
            weaponControl = Bow_Drop_Pic;
            break;
        case "Mace":
            weaponControl = Mace_Drop_Pic;
            break;
        case "RedPotion":
            weaponControl = RedPotion_Drop_Pic;
            break;
        case "BluePotion":
            weaponControl = BluePotion_Drop_Pic;
            break;
    }

    weaponControl.Location = game.WeaponInRoom.Location;
    if (game.WeaponInRoom.PickedUp)
    {
        weaponControl.Visible = false;
    }
    else
        weaponControl.Visible = true;
}
```

# 기본 시스템

## ● Game 클래스

- 게임의 모든것을 제어하는 클래스로  
플레이어, 몬스터, 아이템의 객체들은 이 클래스에서  
생성되고 사용된다.

```
public void Move(Direction direction, Random random)
{
    Player.Move(direction);
    foreach (Enemy enemy in Enemies)
    {
        enemy.Move(random);
    }
}
```

```
public void Attack(Direction direction, Random random)
{
    Player.Attack(direction, random);
    foreach (Enemy enemy in Enemies)
    {
        enemy.Move(random);
    }
}
```

```
case 1:
    Enemies.Add(new Bat(this, GetRandomLocation(random)));
    WeaponInRoom = new Sword(this, GetRandomLocation(random));

    break;
case 2:
    Enemies.Clear();
    Enemies.Add(new Ghost(this, GetRandomLocation(random)));
    WeaponInRoom = new BluePotion(this, GetRandomLocation(random));

    break;
```

# 플레이어의 이동

## ● Mover Class

- 플레이어의 위치를 **newLocation**으로 이동

```
public Point Move(Direction direction, Rectangle boundaries)
{
    Point newLocation = location;

    switch(direction)
    {
        case Direction.Up:
            if(newLocation.Y - MoveInterval >= boundaries.Top)
            {
                newLocation.Y -= MoveInterval;
            }
            break;
        case Direction.Down:
            if (newLocation.Y + MoveInterval <= boundaries.Bottom)
            {
                newLocation.Y += MoveInterval;
            }
            break;
        case Direction.Left:
            if(newLocation.X - MoveInterval >= boundaries.Left)
            {
                newLocation.X -= MoveInterval;
            }
            break;
        case Direction.Right:
            if (newLocation.X + MoveInterval <= boundaries.Right)
            {
                newLocation.X += MoveInterval;
            }
            break;
        default:
            break;
    }

    return newLocation;
}
```



# 플레이어의 이동

**Game** 클래스로부터  
이동명령이 오면  
**Mover** 클래스의  
**Move**를 실행하고  
근처에 가지고 있지  
않은 장비가 있으면  
인벤토리에 추가한다

```
// 이동-----  
public void Move(Direction direction)  
{  
  
    location = Move(direction, game.Boundaries);  
  
    //장비 줍기  
    if(!game.WeaponInRoom.PickedUp)  
    {  
        if(Nearby(game.WeaponInRoom.Location, 30))  
        {  
            if (inventory.Contains(game.WeaponInRoom) == false)  
            {  
                inventory.Add(game.WeaponInRoom);  
                game.WeaponInRoom.PickUpWeapon();  
            }  
        }  
    }  
}
```

# 몬스터의 이동

## ● Enemy Class

각 몬스터들의 Move  
메소드에서는 AI를  
그리고

MoveControl 메소드로  
몬스터 이동

```
protected virtual void MoveControl(Direction NewDirection, Random random)
{
    switch (NewDirection)
    {
        case Direction.Up:
            if (location.Y - MoveInterval >= boundaries.Top)
            {
                location.Y -= MoveInterval;
            }

            break;
        case Direction.Down:
            if (location.Y + MoveInterval <= boundaries.Bottom)
            {
                location.Y += MoveInterval;
            }

            break;
        case Direction.Left:
            if (location.X - MoveInterval >= boundaries.Left)
            {
                location.X -= MoveInterval;
            }

            break;
        case Direction.Right:
            if (location.X + MoveInterval <= boundaries.Right)
            {
                location.X += MoveInterval;
            }

            break;
        default:
            break;
    }
}
```

# 몬스터의 공격

## ● 플레이어가 사정거리 안에 있는지 체크

```
public bool Nearby(Point locationToCheck, int distance)
{
    if (Math.Abs(location.X - locationToCheck.X) < distance && Math.Abs(location.Y - locationToCheck.Y) < distance)
        return true;
    else
        return false;
}
```

## ● 사정거리 안에 있다면 플레이어에게 데미지

```
if (NearPlayer())
{
    game.GiveDamageToPlayer(maxDamage, random);
}
```

- 몬스터는 이동과 공격이 동시에 이루어지기 때문에
- If문은 **Enemy** 클래스의 **MoveControl**메소드에 포함

# Enemy를 상속받은 클래스의 예

이동은 **Enemy**에서  
구현해두었기에 **AI**만  
구현하면 된다.

```
class Bat : Enemy
{
    public Bat(Game game, Point location) : base(game, location, 6)
    {
        maxDamage = 2;
    }

    public override void Move(Random random)
    {
        int AI = random.Next(4);

        if (AI % 2 == 0)
        {
            MoveControl(FindPlayerDirection(), random);
        }
        else
        {
            MoveControl((Direction)random.Next(4), random);
        }
    }
}
```

# 플레이어의 공격

업캐스팅을 사용하여  
장착 장비가 **Attack**  
메소드를 호출하면  
해당 클래스의 **Attack**이  
호출된다.

```
public void Attack(Direction direction, Random random)
{

    if(equippedWeapon is RedPotion)
    {
        RedPotion item = equippedWeapon as RedPotion;
        if(item.Used == false)
        {
            IncreaseHealth(10, random);
            item.Attack(direction, random);
            inventory.Remove(item);
            Weapons.Remove(item.Name);
        }
    }

    else if (equippedWeapon is BluePotion)
    {
        BluePotion item = equippedWeapon as BluePotion;
        if (item.Used == false)
        {
            IncreaseHealth(10, random);
            item.Attack(direction, random);
            inventory.Remove(item);
            Weapons.Remove(item.Name);
        }
    }

    else if(equippedWeapon !=null)
        equippedWeapon.Attack(direction, random);
}
```

# 플레이어의 공격

- 공격방향, 공격 사정거리, 무기 데미지를 계산하여 몬스터에게 데미지를 준다.
- 여기서의 **Nearby**는 오버로딩된 메소드로 몬스터의 **Nearby**와는 다른 행동을 한다.

```
protected bool DamageEnemy(Direction direction, int radius, int damage, Random random)
{
    Point target = game.PlayerLocation;
    foreach(Enemy enemy in game.Enemies)
    {
        if(Nearby(enemy.Location, direction, radius))
        {
            enemy.Hit(damage, random);
            return true;
        }
    }

    return false;
}
```

# 플레이어의 공격

## 예시

- 아래는 **Sword** 클래스의 **Attack** 메소드이며 플레이어로부터 공격 명령이 오면 아래 메소드가 실행된다

```
public override void Attack(Direction direction, Random random)
{
    bool SuccessAttack = false;
    int IntegerDirection = (int)direction;
    if (SuccessAttack == false)
    {
        SuccessAttack = DamageEnemy((Direction)(IntegerDirection % 4), radius, 3, random);
        SuccessAttack = DamageEnemy((Direction)((IntegerDirection + 1) % 4), radius, 3, random);
        SuccessAttack = DamageEnemy((Direction)((IntegerDirection + 3) % 4), radius, 3, random);
    }
}
```

# 3일차

---

## ◎ 추가기능

- 몬스터의 추가
- 강력(?)한 AI



# Boss Moster

---

- 마지막 라운드에 등장하는 보스몬스터
- 일반 몬스터들보다 약 2배가량 크며
- 간단하지만 강력한 **AI**탐제



# Boss Moster

## ◎ 체력 20 이상일 때

- 일반 몬스터와 동일하며 약 30% 확률로 이동

## ◎ 체력 10 이상일 때

- 약 50% 확률로 이동 한다.
- 공격 사거리가 2배 증가한다
- 기본 사정거리 25

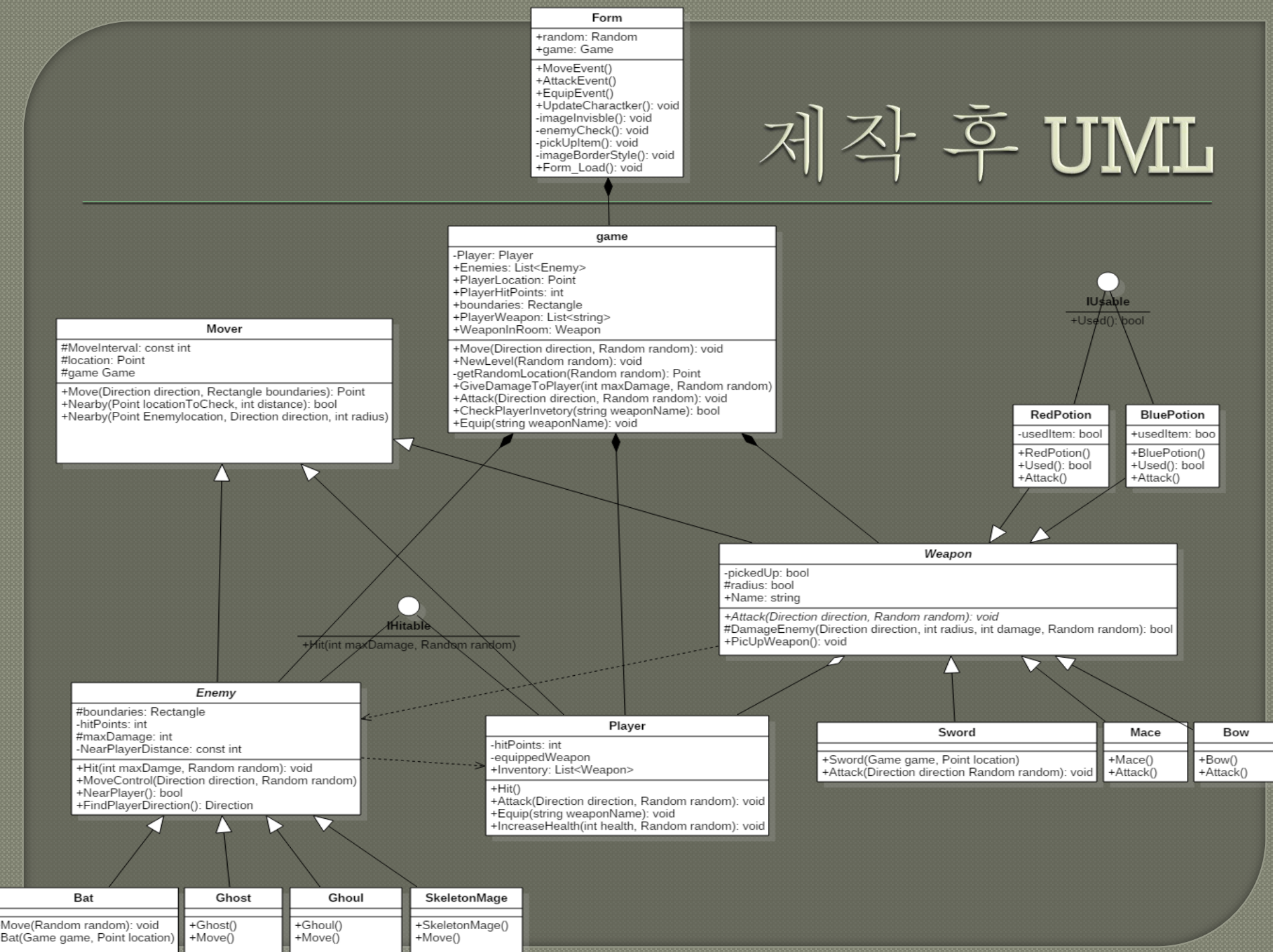
```
if (base.Nearby(game.PlayerLocation, 50))  
{  
    game.GiveDamageToPlayer(maxDamage, random);  
}
```

# Boss Muster

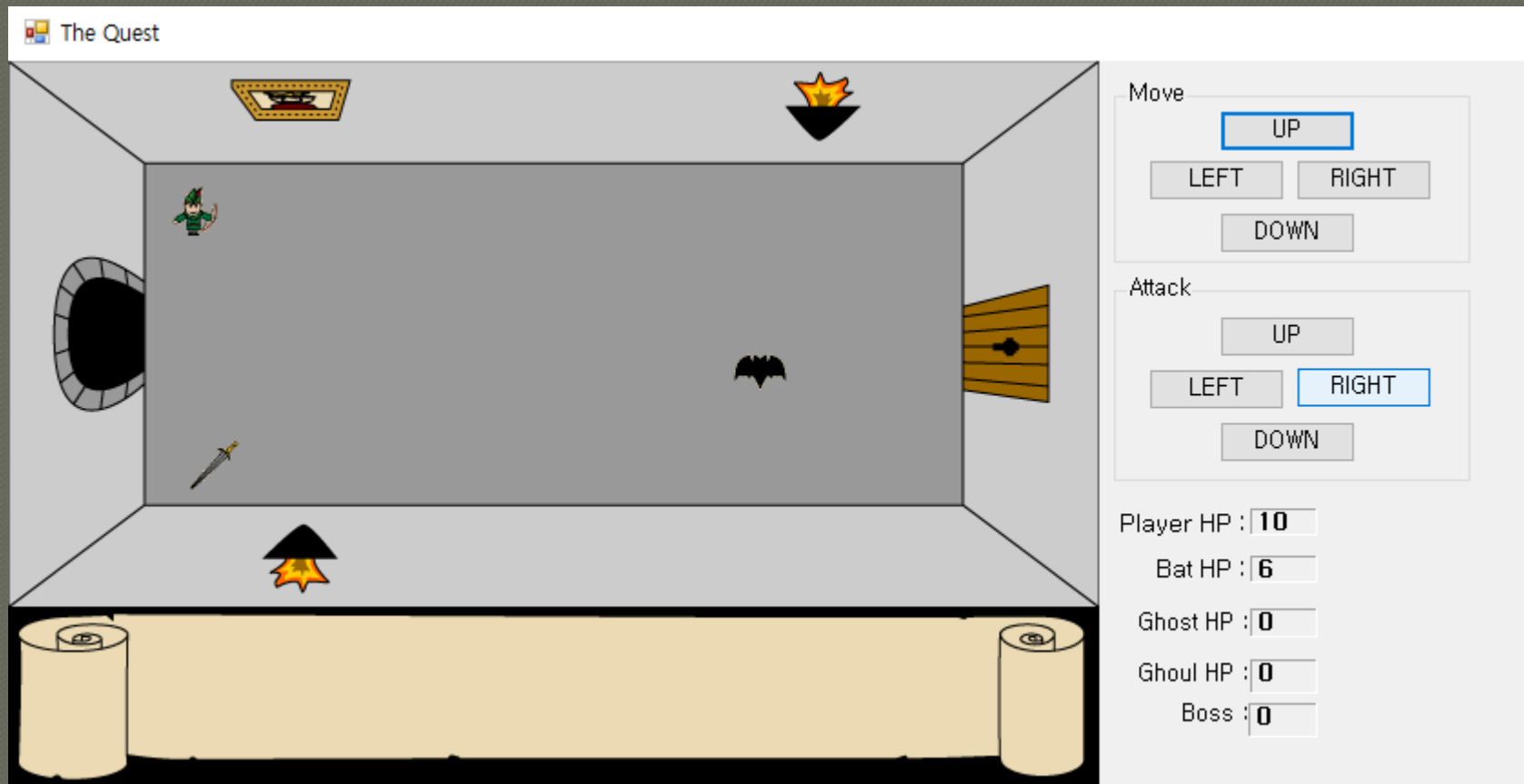
---

- 체력 이 10 미만일 때
  - 공격 사정거리가 줄어들고
  - 플레이어의 현재 체력의 절반만큼의 데미지를 가지게 된다

# 제작 후 UML

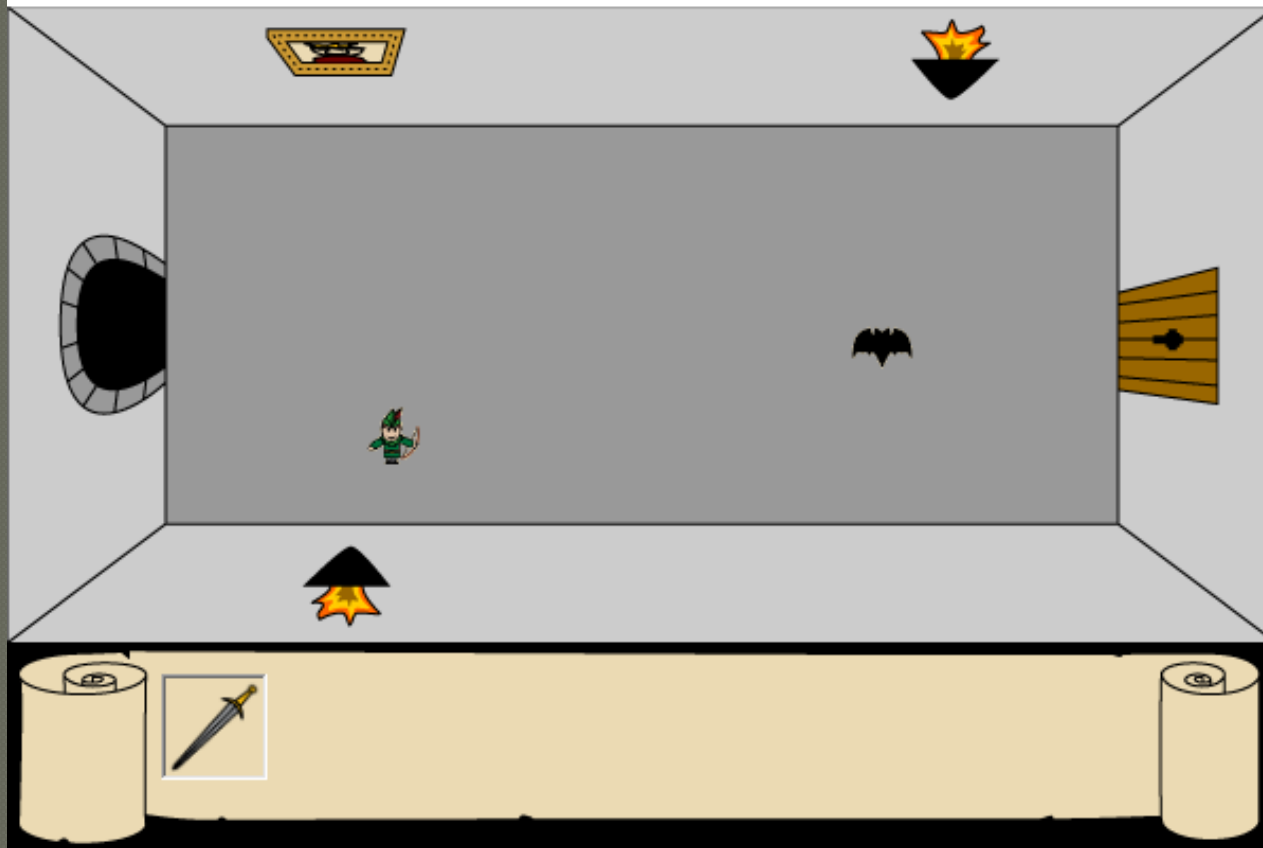


# 실행 화면



# 실행화면

The Quest



Move

UP

LEFT

RIGHT

DOWN

Attack

UP

LEFT

RIGHT

DOWN

Player HP : 10

Bat HP : 6

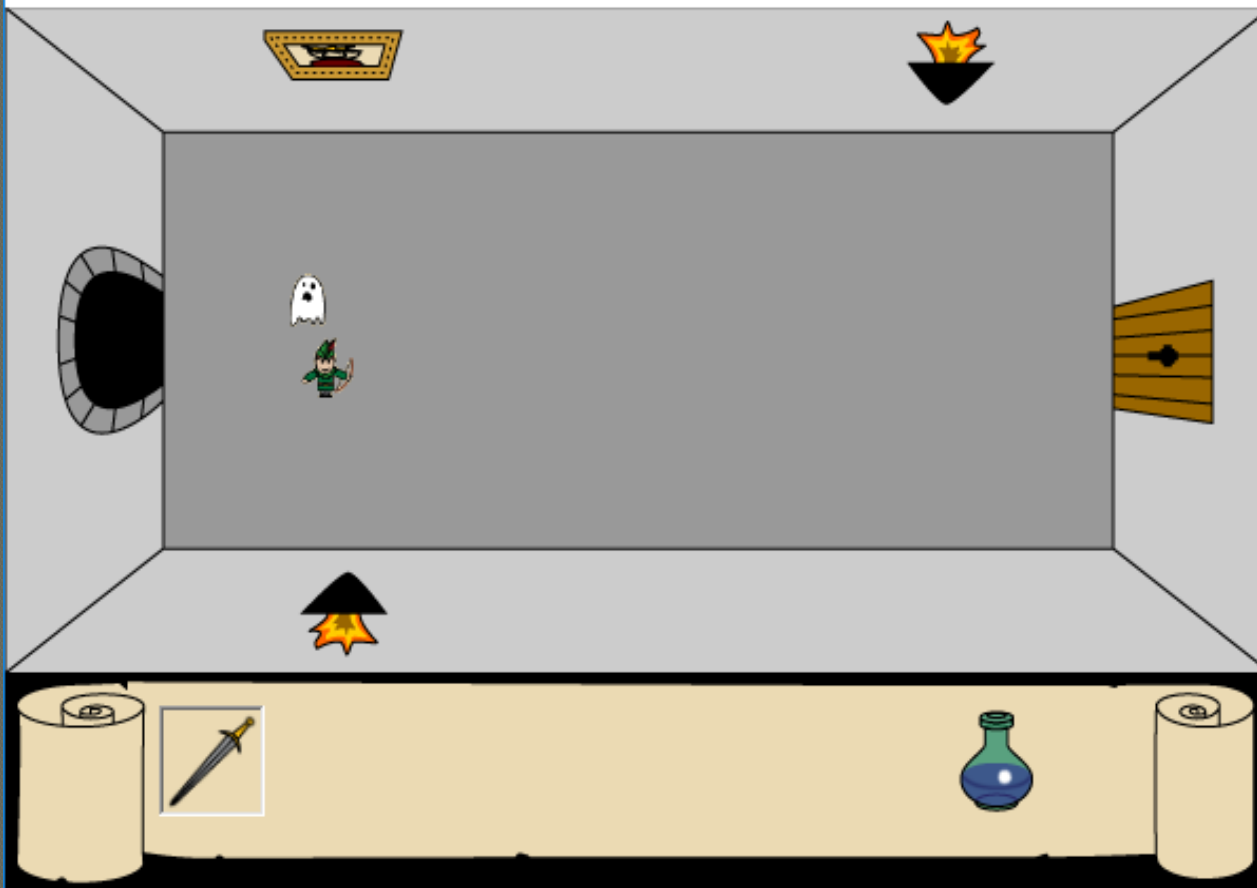
Ghost HP : 0

Ghoul HP : 0

Boss : 0

# 실행화면

The Quest



Move

UP

LEFT

RIGHT

DOWN

Attack

UP

LEFT

RIGHT

DOWN

Player HP : 8

Bat HP : 0

Ghost HP : 5

Ghoul HP : 0

Boss : 0

# 마치며

---

- ◎ 의도치 않은 간단한 버그( 몬스터와 캐릭터가 겹치면 공격어느 방향을 공격하던 데미지가 안들어간다)의 존재
  - 조금더 시간이 있었다면?
- ◎ 몬스터가 너무 싸다
  - 모든 데미지가 **random**일 텐데...