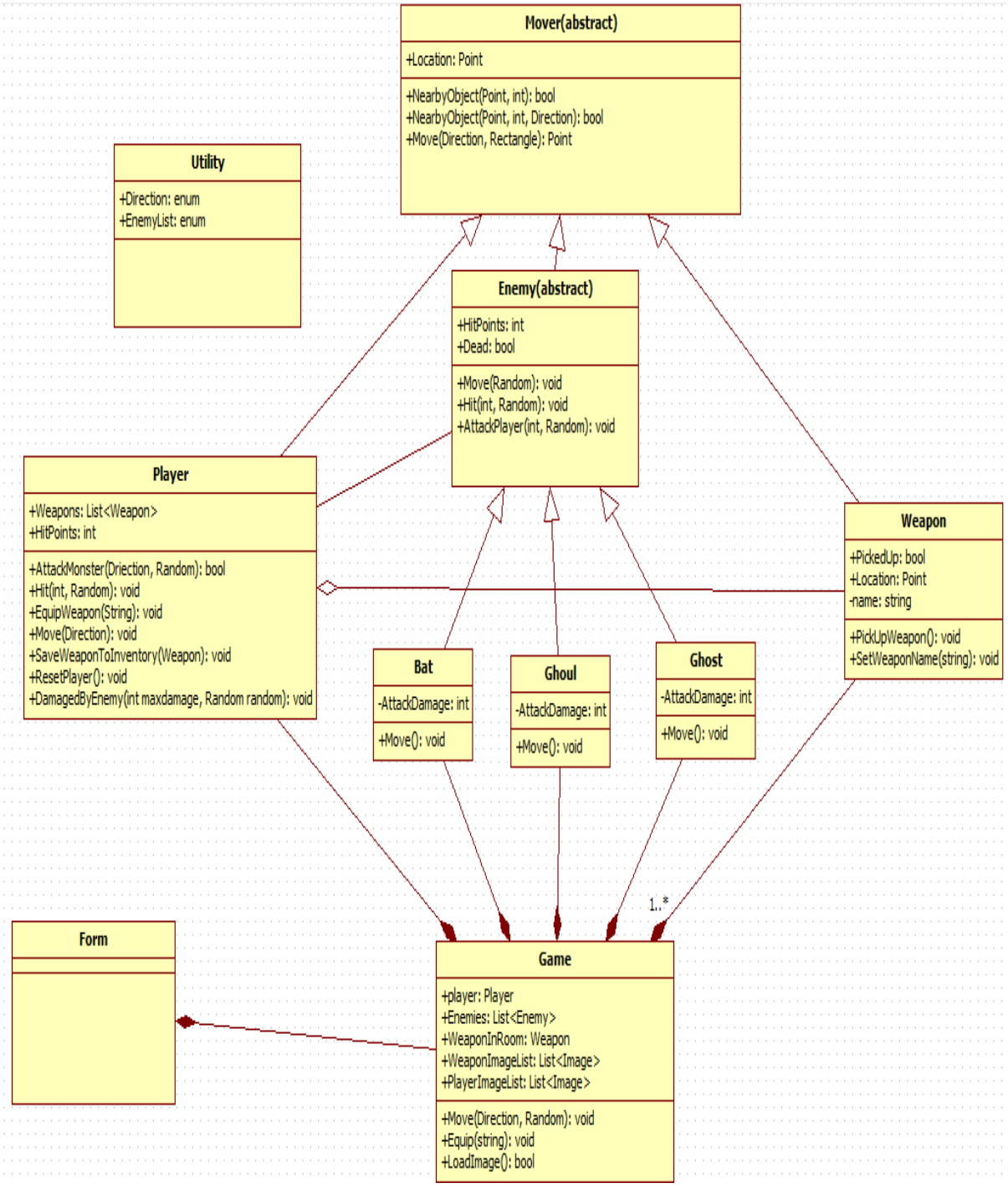
TheQuest-간략한 설명

이재건

1. 클래스 UML



2. Main Function 소개

1) 캐릭터& 몬스터 무브

캐릭터 무브

```
public Point MoveObject(Direction direction, Rectangle boundaries)
{
    Point newLocation = location;
    //Debug.Write(" X :" + newLocation.Y + " Y " + newLocation.Y);
    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;

    }
    // Debug.Write(" X :" + newLocation.Y + " Y " + newLocation.Y);
    return newLocation;
}
```

게임 내 경계 영역보다 작으면 해당 버튼의 방향으로 point값을 변경시킨뒤 return하여 image를 움직여 줍니다.

```
public override void Move(Random random)
{
    if (base.NearPlayer())
    {
        AttackPlayer(attackDamage, random);
    }
    else
    {
        if ((random.Next(1, 4) == 1))  //33%의 확률로 플레이어에게 이동
        {
            Direction direction = FindPlayerDirection(game.PlayerLocation);
            base.location = MoveObject(direction, game.Boundaries);
        }
        else
        {
            Direction direction = (Direction)random.Next(1, 5);
            base.location = MoveObject(direction, game.Boundaries);
        }
    }
}
```

먼저 움직이기 전에 주변에 플레이어가 있는지 체크를 하고 있다면 플레이어를 공격합니다. 없다면 findplayerdirection을 통하여 현재 적과 플레이어의 방향을 계산하여 계산된 방향으로 적을 10만큼 움직여 줍니다.

2)공격 가능한 위치 체크

```
public bool NearbyObject(Point locationToCheck, int distance,Direction direction)
{
    //Console.WriteLine("플레이어 위치 : x" + Location.X + " Y:" + Location.Y);
    //Console.WriteLine("위치: x" + locationToCheck.X + " y: " + locationToCheck.Y);

    if(direction==Direction.Up)
    {
        if (location.Y-locationToCheck.Y  >= 0 && location.Y -distance <= locationToCheck.Y && Math.Abs(location.X - locationToCheck.X) < distance)
            return true;
    }
    else if (direction == Direction.Down)
    {
        if (locationToCheck.Y- location.Y >= 0 && location.Y + distance <= locationToCheck.Y && Math.Abs(location.X- locationToCheck.X) < distance)
            return true;
    }
    else if (direction == Direction.Right)
    {
        if (locationToCheck.X-location.X>=0&& location.X+distance >= locationToCheck.X && Math.Abs(location.Y - locationToCheck.Y) < distance)
            return true;
    }
    else if (direction == Direction.Left)
    {
        if (location.X-locationToCheck.X>= 0 && location.X - distance <= locationToCheck.X && Math.Abs(location.Y - locationToCheck.Y) < distance)
            return true;
    }


    return false;
}
```

공격 버튼 클릭 시 방향에 따라 충돌 범위를 다르게 체크하여 적의 위치 체크 후 공격이벤트

3)이미지 저장 리스트 생성

```
private bool LoadImageIntoList()
{
    EnemyImageList.Add(Image.FromFile(@"image\bat.png"));  //0 is bat
    EnemyImageList.Add(Image.FromFile(@"image\ghost.png"));  //1 is ghost
    EnemyImageList.Add(Image.FromFile(@"image\ghoul.png"));     //2 is ghoul

    WeaponImageList.Add(Image.FromFile(@"image\sword.png"));  //0 is sword
    WeaponImageList.Add(Image.FromFile(@"image\bow.png"));  //1 is bow
    WeaponImageList.Add(Image.FromFile(@"image\mace.png")); //2 is mace
    WeaponImageList.Add(Image.FromFile(@"image\potion_blue.png"));
    WeaponImageList.Add(Image.FromFile(@"image\potion_red.png"));

    PlayerImageList.Add(Image.FromFile(@"image\playerSword.png"));
    PlayerImageList.Add(Image.FromFile(@"image\playerBow.png"));
    PlayerImageList.Add(Image.FromFile(@"image\playerMace.png"));
    PlayerImageList.Add(Image.FromFile(@"image\player.png"));
    for (int i = 0; i < 3; i++)
    {
        if (EnemyImageList[i] == null||WeaponImageList[i]==null||PlayerImageList[i]==null)
            return false;
    }

    return true;

}
```

무기와 플레이어 이미지를 손쉽게 관리하기 위하여 list 에 png 파일을 list화 하여 저장

4)무기 처리 함수

```
public bool MoveToInventory(string name)
{
    if (name == "sword")
    {
        location.X = 89;
        location.Y = 318;
        return true;
    }
    else if (name == "bow")
    {
        location.X = 164;
        location.Y = 318;
        return true;
    }
    else if (name == "mace")
    {
        location.X = 237;
        location.Y = 318;
        return true;
    }
    else if (name == "bluePortion")
    {
        location.X = 317;
        location.Y = 318;
        return true;
    }
    else if(name=="redPortion")
    {
        location.X = 377;
        location.Y = 318;
        return true;
    }

    return false;
```

무기의 sub클래스를 따로 생성하지 않고 이름을 통하여 분류하는 함수. 무기를 획득 시 이미지의 위치를 변경

5)SetImage와 UpdateCharater() 함수를 통하여

매 턴 종료시 캐릭터의 위치와 Image를 업데이트 시켜 줍니다. EnemyList와 Inventory<Weapon>을 for문을 통해 전체적인 검사를 해주어 업데이트 해줍니다.

5)HitImage 추가

```
public void DamageToplayer(int maxdamage, Random random)
{
    if (player.Hitpoints >= 0)
    {

        hitImage.Location = new Point(PlayerLocation.X-10,PlayerLocation.Y+10);
        hitImage.Visible = true;
        player.DamagedByEnemy(maxdamage, random);
    }
}
```

플레이어 또는 Enemy가 공격 당할 시 hitimage를 추가하여 명확한 타격을 제공합니다.

3. 필요한 업데이트들

1) 습득하지 않은 아이템들이 다음 스테이지에서 사라지지 않음

2) 게임 플레이 타임을 증가시킬 수 있는 여러가지 난이도 추가 필요

3) 하드 코딩이 되어진 부분(ex)무기의 클래스화)를 통하여 향후 프로젝트 유지/보수를 손쉽게 할 수 있도록 수정