Game Development Journey

PORTFOLIO

by Unity

MY VALUES

VISION

프로젝트 팀 경험 및 **인게임 기능 제작** 다수 경험 보유

MISSION

팀의 진행 스타일을 분석하고, 그에따라 소통하며 맞추어 나갑니다. **팀원들과의 협력을** 통해 **효율적으로 원활하게 작업할 수 있습니다.**

PROJECT

1

(개인)골드메탈 3d Copy Practice

목표 – 개인적인 드론 보스 추가해보기 및 레이저 구현

Quad Action

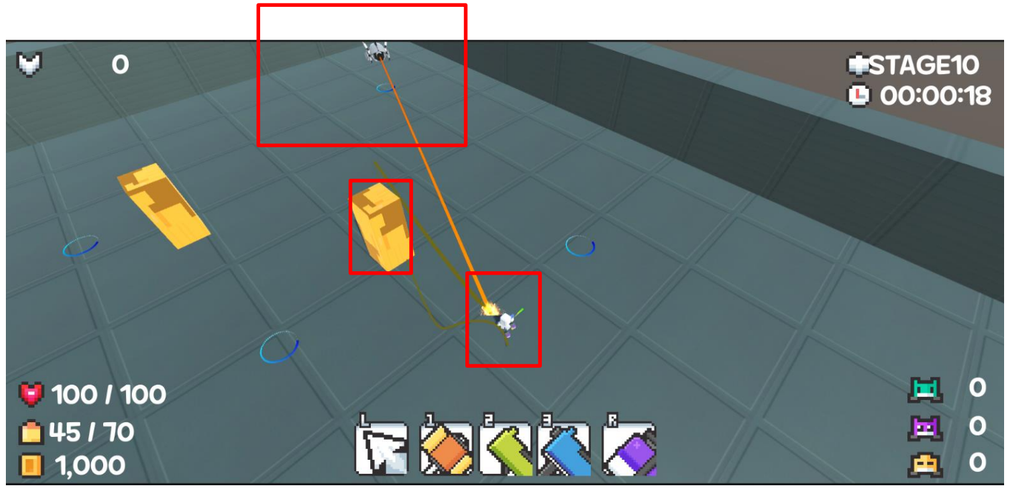
드론 에셋 출처

https://assetstore.unity.com/account/asse
Low poly combat drone

그 외 출처

https://www.goldmetal.co.kr/

```
RaycastHit hit;
transform.LookAt(PlayerTransform.position); //바라보기

Vector3 targetPosition = PlayerTransform.position; // 대상 게임 오브젝트의 위치
Vector3 currentPosition = this.gameObject.transform.position; // 현재 게임 오브젝트의 위치
targetPosition = new Vector3(targetPosition.x, 0, targetPosition.z);
currentPosition = new Vector3(currentPosition.x, 0, currentPosition.z);

float distance = Vector3.Distance(targetPosition, currentPosition); // 대상과의 거리 계산
dsahu = distance;
if (Physics.Raycast(transform.position, transform.forward, out hit) && !hit.collider.gameObject.CompareTag("Player")) //충돌 객체 레이캐스트로 정보 얻기
{

    LookPlayer = PlayerTransform.transform.localPosition;
    laserBall.isLaserball = false;
    laserBall.gameObject.SetActive(false);
    laserBall.gameObject.transform.localPosition = new Vector3(0, 0, 96.1f);

    secondboss.lr.enabled = false;
    StartCoroutine(Rest());
}
else if (distance <= 150f)
{
    laserBall.isLaserball = true;
    laserBall.gameObject.SetActive(true);

    secondboss.lr.enabled = true;
    Vector3 dronePosition = new Vector3(SecondBoss.position.x, 20, SecondBoss.position.z);
    SecondBoss.position = Vector3.Lerp(transform.position, dronePosition, 1f * Time.deltaTime);
}
else
{
    laserBall.isLaserball = false;
```



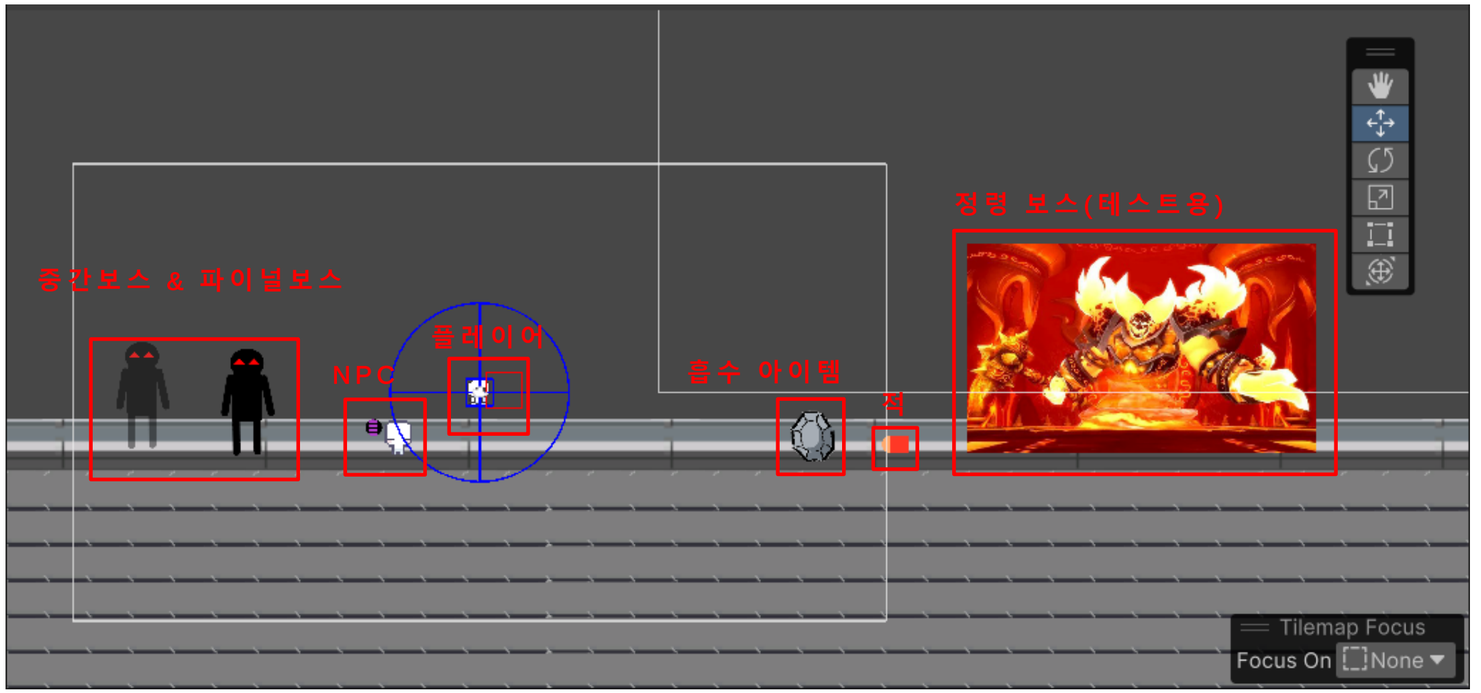
#기능 설명
해당 사각지대에 들어오면
드론이 플레이어를 따라가는 **레이저를 발사합니다.**
플레이어는 **벽에 숨거나 거리 조절하며** 싸워야 합니다.

#코드 설명
1.**레이캐스트의 hit**을 선언합니다.
2.현재 오브젝트가 **PlayerTransform**의
위치를 바라보도록 설정합니다.
3.게임 오브젝트의 위치를 **targetPosition**에 저장합니다.
4.현재 게임 오브젝트의 위치를 **currentPosition**에 저장합니다.
5.각 위치의 **y 좌표를 0**으로 만듭니다.
6.대상과의 거리를 계산하여 변수 **distance**에 저장합니다.

* 중간보스 & 파이널보스는 다른 팀원이 작업하였습니다. *

PROJECT

2



(팀)동아리 액션 플랫포머 프로젝트

목표 – 동아리에서의 협업 경험, 소통

DeathFire

2d 에셋 출처

라그나로스 - 나무위키

https://www.goldmetal.co.kr/

키를 눌러 변경


```
foreach (KeyCode Key in number.Keys)
{
    if (Input.GetKeyDown(Key))
    {
        SkillTransformChange(Key);
    }
}
```

```
void SkillTransformChange(KeyCode Key)
{
    ingameSkillsImage[gaugeSequence].transform.position = ingameSkillsImage[number[Key]].transform.position;
    ingameSkillsImage[gaugeSequence].rectTransform.sizeDelta = new Vector2(50, 50);
    gaugeCountText[gaugeSequence].transform.position = gaugeCountText[number[Key]].transform.position;
    gaugeNumberText[gaugeSequence].transform.position = gaugeNumberText[number[Key]].transform.position;

    ingameSkillsImage[number[Key]].transform.position = skill0ImageTransform.transform.position;
    ingameSkillsImage[number[Key]].rectTransform.sizeDelta = new Vector2(100, 100);
    gaugeCountText[number[Key]].transform.position = skill0TextTransform.transform.position;
    gaugeNumberText[number[Key]].transform.position = skill0NumberTextTransform.transform.position;
    gaugeSequence = (int)Key - (int)KeyCode.Alpha1;
}
```

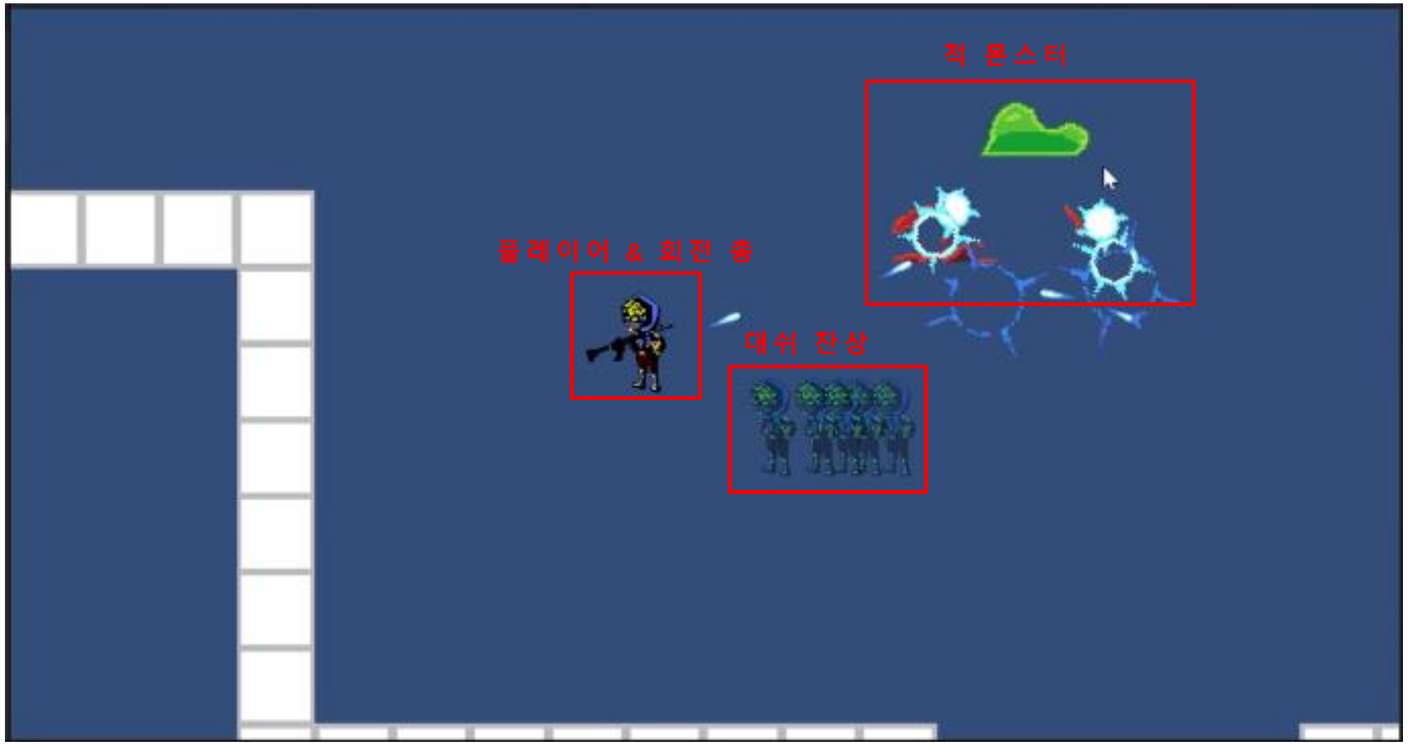
#기능 설명
키를 눌러 게임 중 사용 원소를 바꿀 수 있으며,
탭 키를 눌러 **스킬 업그레이드창을** 사용할 수 있습니다.
속도, 스킬 능력 등이 달라집니다.

#코드 설명
누르는 키에 따라

1. **이미지의 위치를** 변경합니다.
2. **이미지 크기를** 변경합니다.
3. **텍스트 위치를** 변경합니다.
4. **숫자 텍스트 위치를** 변경합니다.
5. **gaugeSequence**에 **현재 눌린 키에 대한 순서를** 저장합니다.

PROJECT

3



(개인)엔터 더 건전 플레이 방식 Copy Practice

목표 – 타 게임의 기능 구현, 기능 추가

Warrior_In_The_Box

총알 이미지 에셋 출처

https://assetstore.unity.com/account/assets
Sprite muzzle flashes

몬스터 이미지 에셋 출처

https://assetstore.unity.com/account/assets
Slime Enemy - Pixel Art

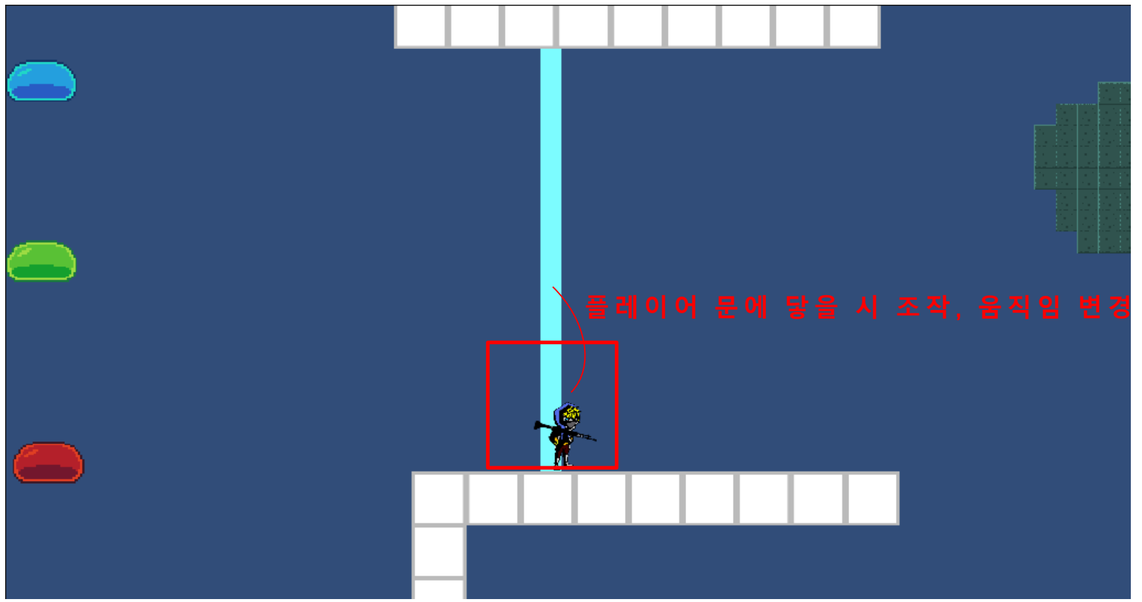
```
public PlayerType CurrentPlayerType { get; set; }
```

```
Vector2 nextVec = inputVec.normalized * playerSpeed * Time.fixedDeltaTime;
rigid.MovePosition(rigid.position + nextVec);
```

```
if (horizontalInput > 0)//오른쪽 방향키
{
    transform.rotation = Quaternion.Euler(0, 0, 0); //오른쪽방향
    transform.Translate(transform.right * playerSpeed * Time.fixedDeltaTime);
}
else if (horizontalInput < 0)//왼쪽 방향키
{
    transform.rotation = Quaternion.Euler(0, 180, 0); //왼쪽방향
    transform.Translate(transform.right * -1 * playerSpeed * Time.fixedDeltaTime);
}
```

```
target = transform.position;
// 카메라의 마우스포인터 받기
mouse = Camera.main.ScreenToWorldPoint(Input.mousePosition);
// target에서 mouse까지의 라디안 값을 계산하고, Mathf.Rad2Deg를 사용하여 해당 값을 각도로 변환
angle = Mathf.Atan2(mouse.y - target.y, mouse.x - target.x) * Mathf.Rad2Deg;
// 현재 객체를 해당 각도 angle만큼 회전
this.transform.rotation = Quaternion.AngleAxis(angle, Vector3.forward);

//좌우 반전
if (angle > 90 || angle < -90)
{
    spriteRenderer.flipY = true;
}
else
{
    spriteRenderer.flipY = false;
}
```

Quaternion.Euler
해당 함수 인자에 오일러각을 넣으면 쿼터니언으로 변환된 값을 반환시켜준다.
Quaternion.AngleAxis
AngleAxis() 메서드는 axis를 중심으로 angle(°)만큼 회전시킨 Quaternion을 반환한다.



#기능 설명
플레이어가 문에 닿을 시 중력이 0인 **쯔꾸르 액션**에서
중력이 있는 **플랫포머 액션**으로 **폼 체인지**가 일어납니다.

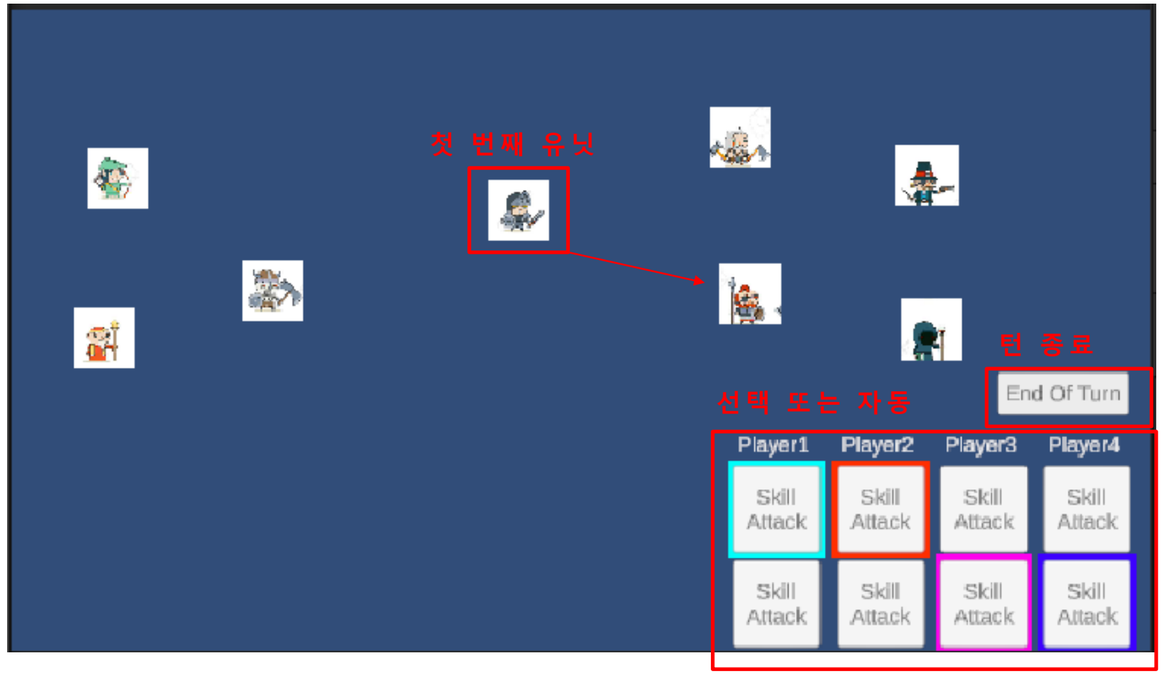
#코드 설명
1.**Enum**값으로 플레이어 타입과 총 타입을 구별하였습니다.

2.입력된 벡터를 **정규화**하고, 플레이어의 속도와 고정된 시간 간격을 **곱하여** 다음 프레임에서 **이동할 벡터를** 계산합니다.

3.현재 위치에 **nextVec**를 더한 새로운 위치로 플레이어를 이동시킵니다.

PROJECT

4



Project Graden

(팀)동아리 모바일 턴제 RPG

목표 – 팀 경험, 소통, 턴제 순서 구현

캐릭터 임시 이미지 출처

https://kr.123rf.com/photo_90922699_lineart-%EB%82%A8%EC%84%B1-%EC%97%AC%EC%84%B1-%ED%8C%90%ED%83%80%EC%A7%80-rpg-%EA%B2%8C%EC%9E%84-%EC%BA%90%EB%A6%AD%ED%84%B0-%EB%B2%A1%ED%84%B0-%EC%95%84%EC%9D%B4%EC%BD%98-%EC%84%B8%ED%8A%B8-%EB%B2%A1%ED%84%B0-%EC%9D%BC%EB%9F%AC%EC%8A%A4%ED%8A%B8.html

2. 스킬

팀원의 전투 시스템 기획 참고
*치유 제외

규칙 - 스킬 예약

- 스킬 슬롯의 스킬들을 클릭하여 각 아군의 스킬 시전을 예약할 수 있다.
- 예약한 스킬을 다시 클릭하여 예약취소 할 수 있다.
- 각 아군당 한 턴에 최대 한 개의 스킬만을 예약할 수 있다.
- 공격 스킬의 경우 전방의 적을 우선으로 공격한다.
- 치유 스킬의 경우 현재 체력 비율이 가장 낮은 아군을 우선으로 치유한다.

< 스킬 예약 표시 예시 >

```
public void ClickBtn()
{
    GameObject clickObject = EventSystem.current.currentSelectedGameObject;

    for (int i = 0;i < 8; i += 2)
    {
        int first = i;
        int second = i + 1;

        int nothing = 0;
        int plusOne = 1;

        if (sonButtons[first] == clickObject)
        {
            TwiceChoiceButton(i, nothing, plusOne);
        }
        else if (sonButtons[second] == clickObject)
        {
            TwiceChoiceButton(i, plusOne, nothing);
        }
    }
}
```

```
void TwiceChoiceButton(int i, int num1, int num2)
{
    if (isSetButton[i + num1] == false)
    {
        if (isSetButton[i + num2] == true)
        {
            isSetButton[i + num2] = false;
            isSetButton[i + num1] = true;

            for (int j = 0; j < sequenceValue.Count; j++)
            {
                if (sequenceValue[j] == i + num2)
                {
                    sequenceValue[j] = i + num1;
                    break;
                }
            }

            UpdateSequenceImages(btnRectTransforms[i + num2], i + num2, false);
            UpdateSequenceImages(btnRectTransforms[i + num1], i + num1, true);
        }
        else
        {

            isSetButton[i + num1] = true;
            sequenceValue.Add(i + num1);
            UpdateSequenceImages(btnRectTransforms[i + num1], i + num1, true);
        }
    }
```

```
void UpdateSequenceImages(RectTransform rectTransform, int imageNumber, bool isOk)
{
    sequenceImages[imageNumber].GetComponent<RectTransform>().anchoredPosition = rectTransform.anchoredPosition;
    sequenceImages[imageNumber].SetActive(isOk);
}
```

```
void RandomChoiceButtons(int activeButtons)
{
    bool halfATK = Dods_ChanceMaker.GetThisChanceResult_Percentage(50);
    if (halfATK)
    {
        isSetButton[activeButtons] = true;
        sequenceValue.Add(activeButtons);
        UpdateSequenceImages(btnRectTransforms[activeButtons], activeButtons, true);
    }
    else
    {
        isSetButton[activeButtons + 1] = true;
        sequenceValue.Add(activeButtons + 1);
        UpdateSequenceImages(btnRectTransforms[activeButtons + 1], activeButtons + 1, true);
    }
}
```

#구현 기능 설명

ClickBtn:

버튼이 클릭되었을 때 호출됩니다.

TwiceChoiceButton 메서드를 호출합니다.

TwiceChoiceButton:

각각 선택 가능한 버튼을 처리하며, 중복 선택을 방지하고 순서를 업데이트합니다.

UpdateSequenceImages:

순서 값을 업데이트하고, 해당하는 이미지를 업데이트합니다.

RandomChoiceButton:

50%의 확률로 버튼을 랜덤하게 선택하고, 선택된 버튼을 활성화하며 순서를 업데이트합니다.

4. 기본공격

팀원의 전투 시스템 기획 참고

*치명타 제외

규칙 - 기본 공격

- 스킬 사용이 끝난 후, 속도 능력치가 높은 순으로 아군 유닛과 적군 유닛이 번갈아가며 기본 공격을 한다.
- 기본 공격은 기본적으로 (최종 공격력)(100% + (기본 공격 증폭)%)의 피해를 입힌다.
- 기본 공격에는 치명타가 적용될 수 있으며, 치명타는 (최종 치명타 확률)%의 확률로 발생한다.
- 치명타 공격은 (기본공격)x(최종 치명타 피해%)의 피해를 입힌다.
- 전방의 적을 우선으로 공격한다.

규칙 - 기본 공격 - 연속 공격

- 공격 유닛과 피격 유닛의 최종 속도 능력치가 2배 이상 차이날 경우 (공격 유닛 최종 속도)/(피격 유닛 최종 속도)를 내림한 수 만큼 기본공격을 시전한다.
- 연속 공격으로 인해 발생한 추가 공격 또한 모두 기본 공격으로 간주한다.

```
void StartBasicATK() //속도 능력치가 높은 순으로 기본공격 실행
{
    while (p1AtkSpeed.Any() || p2AtkSpeed.Any())
    {
        float maxSpeedP1 = p1AtkSpeed.Any() ? p1AtkSpeed.Max() : float.MinValue;
        float maxSpeedP2 = p2AtkSpeed.Any() ? p2AtkSpeed.Max() : float.MinValue;

        int indexP1 = p1AtkSpeed.IndexOf(maxSpeedP1);
        int indexP2 = p2AtkSpeed.IndexOf(maxSpeedP2);

        if (maxSpeedP1 > maxSpeedP2)
        {
            P1ATKTest(indexP1);
            p1AtkSpeed.Remove(maxSpeedP1);
        }
        else if (maxSpeedP2 > maxSpeedP1)
        {
            P2ATKTest(indexP2);
            p2AtkSpeed.Remove(maxSpeedP2);
        }
        else
        {
            float maxSpeed = Math.Max(maxSpeedP1, maxSpeedP2);

            if (maxSpeedP1 == maxSpeed && maxSpeedP2 == maxSpeed)
            {
                //50% 처리
                bool halfATK = Dods_ChanceMaker.GetThisChanceResult_Percentage(50);
                if (halfATK)
                {
```

```
bool ArePositionsEqual(List<int> list1, List<int> list2, int valueToCompare) //리스트의 인덱스 같은 값 체크
{
    int indexInList1 = list1.IndexOf(valueToCompare);
    int indexInList2 = list2.IndexOf(valueToCompare);

    return indexInList1 == indexInList2 && indexInList1 != -1;
}
```

```
IEnumerator StartSkillATK()
{
    for (int i = 0; i < 4; i++)
    {
        bool result = ArePositionsEqual(p1InputValue, p2InputValue, i + 1); //1부터 같은지 체크

        if (result) //스킬순서가 같다면 속도 비교 실행
        {
            yield return StartCoroutine(SameSkillSequence(i));
        }
        else //스킬순서가 다르다면 둘 중 랜덤 실행
        {
            yield return StartCoroutine(RandomSkillSequence(i));
        }
        //Debug.Log("i의 위치가 두 리스트에서 똑같은가? " + result);
    }
}
```

#구현 기능 설명

ArePositionsEqual:

두 리스트에서 주어진 값의 인덱스가 같은 값이 존재하는지 확인합니다.

StartSkillATK:

두 리스트에서 값의 위치를 비교하여 같으면 순서대로 스킬 실행, 다르면 랜덤하게 실행합니다.

StartBasicATK:

두 플레이어의 공격 속도를 비교하여 높은 속도 순서로 공격을 실행하고, 속도가 같으면 50% 확률로 랜덤하게 선택하여 실행합니다.

5. 예외처리 - 동속

팀원의 전투 시스템 기획 참고

규칙 - 동속(스킬)

- 스킬 사용 순서가 같은 적군 유닛과 아군 유닛의 속도 능력치 격차가 5% 이내일 때 발생한다.
- 동속전 발동 조건 - 격차 계산식 :
  (속도가 더 높은 유닛의 속도 능력치의 5%) > (속도가 더 높은 유닛의 속도 능력치) - 속도가 낮은 유닛의 속도 능력치)
- 동속 조건이 충족되어 스킬 동속전 이벤트가 발생하면, 화면에 스킬 동속전 이펙트가 출력되며, 동속전 대상인 양 팀의 유닛 중 하나가 50%의 확률로 먼저 스킬을 사용한다.

규칙 - 동속(스킬)

- 기본 공격 사용 순서가 같은 적군 유닛과 아군 유닛의 속도 능력치 격차가 5% 이내일 때 발생한다.
- 동속전 발동 조건 - 격차 계산식 : (속도가 더 높은 유닛의 속도 능력치의 5%) > (속도가 더 높은 유닛의 속도 능력치) - 속도가 낮은 유닛의 속도 능력치)
- 동속 조건이 충족되어 기본 공격 동속전 이벤트가 발생하면, 화면에 기본공격 동속전 이펙트가 출력되며, 동속전 대상인 양 팀의 유닛 중 하나가 50%의 확률로 먼저 기본공격을 사용한다.

50% 랜덤전

```
IEnumerator RandomSkillSequence(int order)
{
    //50% 처리
    bool halfATK = Dods_ChanceMaker.GetThisChanceResult_Percentage(50);
    if (halfATK)
    {
        yield return StartCoroutine(P1SkillTest(order));
        yield return StartCoroutine(P2SkillTest(order));
    }
    else
    {
        yield return StartCoroutine(P2SkillTest(order));
        yield return StartCoroutine(P1SkillTest(order));
    }
}
```

동속전

```
IEnumerator SameSkillSequence(int order) //스킬 사용 순서가 같으면 속도가 높은 것 실행 후 2번째 것 실행
{
    if (p1AtkSpeed[order] > p2AtkSpeed[order])
    {
        float threshold = p1AtkSpeed[order] * 0.05f;
        if (threshold > p1AtkSpeed[order] - p2AtkSpeed[order]) //동속전
        {
            //동속전 이펙트 실행 포함
            yield return StartCoroutine(RandomSkillSequence(order));
        }
        else //그냥 실행
        {
            yield return StartCoroutine(P1SkillTest(order));
        }
    }
    else
    {
        float threshold = p2AtkSpeed[order] * 0.05f;
        if (threshold > p2AtkSpeed[order] - p1AtkSpeed[order]) //동속전
        {
            //동속전 이펙트 실행 포함
            yield return StartCoroutine(RandomSkillSequence(order));
        }
        else //그냥 실행
        {
            yield return StartCoroutine(P2SkillTest(order));
            yield return StartCoroutine(P1SkillTest(order));
        }
    }
}
```

#구현 기능 설명

SameSkillSequence:
주어진 순서에 따라 두 플레이어의 스킬 시퀀스를 실행하는 메소드입니다.
두 플레이어 중 속도가 높은 플레이어의 스킬을 먼저 실행하며,
동일한 속도의 경우 5% 이내의 차이가 있으면 랜덤 스킬 순서를 실행합니다.

RandomSkillSequence:
50% 확률로 두 플레이어 간의 랜덤 스킬 순서를 실행하는 메소드입니다.
랜덤 결과에 따라 두 플레이어 중 하나가 먼저 스킬을 사용하고,
그에 따라 다른 플레이어도 스킬을 사용합니다.

출처 확인

#50% 확률은 블로그 **Dods_ChanceMaker**를 사용하였습니다.

https://dodnet.tistory.com/4484

#템플릿 출처

Modern and Minimal Business Proposal Presentation

https://www.canva.com/ko_kr/presentations/templates/

#PPT Font 출처

소제목 : (Montserrat Classic, 62pt)

본문 (맑은 고딕, 25pt)

THANK YOU