

[깃허브] <https://github.com/neulsesron/Portfolio>

[유튜브] https://www.youtube.com/playlist?list=PLhvPX7WLxufJ7Bt0JAKL8vCtbvQk_XBjy

깃발잡기		3D 던전 RPG	
클라이언트	유니티	클라이언트	유니티
서버	포톤2		
			
<ul style="list-style-type: none">- 포톤2를 사용하여 4인 멀티플레이 게임을 만들었습니다.- 서버 접속, 방 생성 및 참가, 채팅, 아이템 사용, 깃발 잡기, 점수·체력바 등의 유저 정보 동기화 등을 구현하였습니다.		<ul style="list-style-type: none">- 3D 던전 맵을 구성하여 RPG 게임을 만들었습니다.- 기본 이동, 퀘스트, 아이템 사용, HP등의 스탯, 몬스터 공격, 인벤토리, 저장·불러오기 기능 등을 구현하였습니다.	

(1) 4인 멀티 깃발잡기

: 포톤 서버를 사용하여 로그인, 로비, 방 생성, 채팅 등을 구현하였습니다. 게임 내의 모든 공유 변수(채팅, 점수, 체력바 등)는 master 클라이언트가 일괄적으로 관리하도록 했습니다. 깃발이 생성되는 쿨타임 동안 맵에 드랍되는 공격 아이템을 획득하여 상대를 공격할 수 있습니다. 피격 판정은 공격자가 아닌 피격 대상이 맞았을 때를 기준으로 했습니다.

(2) 던전 RPG

: 몬스터의 AI를 유한상태머신을 사용하여 구현하였습니다. 또한, 몬스터의 종류를 상속을 통해 다양하게 구현하였습니다. 피격/공격/상호작용 가능 여부를 인터페이스를 사용하여 구현하였습니다. 길 찾기에는 NavMeshAgent 컴포넌트를 사용하였습니다. 플레이어의 체력은 현재 HP만 저장하여 관리하고, Max HP는 게임을 불러왔을 때 장착한 장비를 검사하여 계산하도록 했습니다.

- 공격 타겟 검사

```

Collider[] targetsInViewRadius = Physics.OverlapSphere(transform.position, viewRadius, targetMask);
for (int i = 0; i < targetsInViewRadius.Length; i++) {
    Transform target = targetsInViewRadius[i].transform;

    Vector3 dirToTarget = (target.position - transform.position).normalized;
    if (Vector3.Angle(transform.forward, dirToTarget) < viewAngle / 2) {
        float distToTarget = Vector3.Distance(transform.position, target.position);
        if (!Physics.Raycast(transform.position, dirToTarget, distToTarget, obstacleMask)) {
            if (target.GetComponent<IDamageable>()?.IsAlive ?? false) {
                visibleTargets.Add(target);

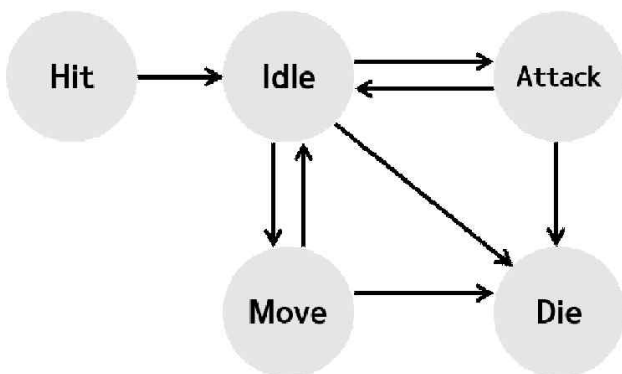
                if (nearestTarget == null || (distanceToTarget > distToTarget)) {
                    nearestTarget = target;
                }

                distanceToTarget = distToTarget;
            }
        }
    }
}
}

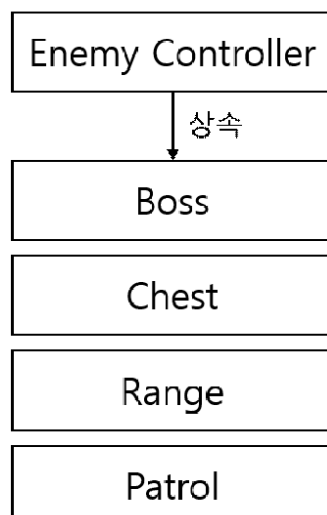
```

범위 내의 콜라이더를 가진 대상을 검사하고, LayerMask를 사용하여 장애물 뒤에 있는 타겟을 제외한 뒤, 살아있는 대상이면 타겟 리스트에 추가하도록 했습니다.

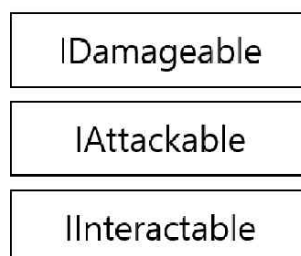
- 유한상태머신



- 몬스터 종류



- 인터페이스


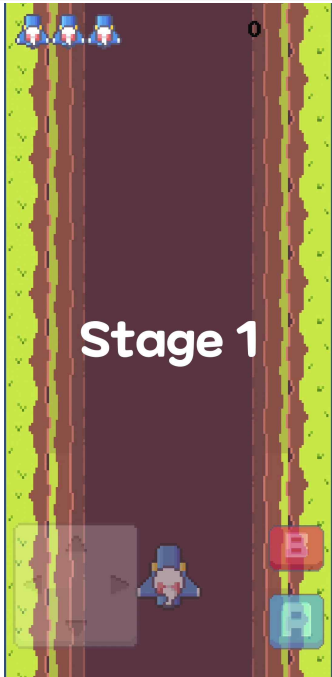


```

13 references
public interface IInteractable
{
    3 references
    bool IsInteractable { get; }
    2 references
    float Distance { get; }

    1 reference
    void Interact(GameObject other);
    5 references
    void StopInteract(GameObject other);
}

```

퍼즐 어드벤처		슈팅게임	
클라이언트	유니티	클라이언트	유니티
			
<ul style="list-style-type: none"> - 스토리 중심의 퍼즐 어드벤처 장르 게임을 만들었습니다. - 오브젝트 상호작용, NPC와 같은 각 맵의 이벤트, 미니맵, 인벤토리, 퍼즐 등을 구현하였습니다. 		<ul style="list-style-type: none"> - 간단한 모바일 슈팅게임을 만들었습니다. - 기본 조작, 레벨 디자인, 보스 공격 패턴 다양화 등을 구현하였습니다. 	

(3) 퍼즐 어드벤처

: 다양한 오브젝트와의 상호작용을 통해 게임을 진행합니다. 맵 별 이벤트를 저장하기 위해 딕셔너리 형을 사용하였고, 저장/불러오기 시에는 리스트로 변환했습니다.

(4) 슈팅게임

: 모바일 게임이며, 레벨/무한 두 가지 모드를 플레이 할 수 있습니다. 텍스트 파일로 레벨을 디자인하였습니다. 배열을 사용한 오브젝트 풀링 기법으로 몬스터와 총알 등을 관리하였습니다. 보스의 공격패턴을 4가지로 구현하여 2초마다 다음 패턴으로 넘어가도록 하였습니다.

```

void FireArc()
{
    if (health <= 0)
        return;

    currPatternCount++;

    Vector2 dir = new Vector2(Mathf.Sin(15 * Mathf.PI * currPatternCount/maxPatternCount[patternIndex]), -1);
    BossFire("EnemyBulletA", Vector3.zero, dir.normalized);

    if (currPatternCount < maxPatternCount[patternIndex])
        Invoke("FireArc", 0.15f);
    else
        Invoke("Think", 2);
}

```

```
public GameObject MakeObj(string type)
{
    switch (type) {
        case "Boss":
            targetPool = boss;
            break;
        case "EnemyL":
            targetPool = enemyL;
            break;

        ...

        case "BossBulletA":
            targetPool = bossBulletA;
            break;
        case "BossBulletB":
            targetPool = bossBulletB;
            break;
    }

    for (int i = 0; i < targetPool.Length; i++) {
        if (!targetPool[i].activeSelf) {
            targetPool[i].SetActive(true);
            return targetPool[i];
        }
    }

    return null;
}
```

테트리스			
클라이언트	콘솔(C++)		
			
<ul style="list-style-type: none"> - 콘솔에서 동작하는 테트리스 게임을 만들었습니다. - 블록의 이동, 회전, 충돌 및 점수 시스템을 구현하였습니다. 			

(5) 콘솔 테트리스

: C++을 사용하여 개발한 콘솔 테트리스 게임입니다. board, block 두 가지 클래스를 생성하여 구현하였습니다. 블록의 이동, 회전, 강제낙하, 스코어, 게임 종료 등의 기능을 구현하였습니다.