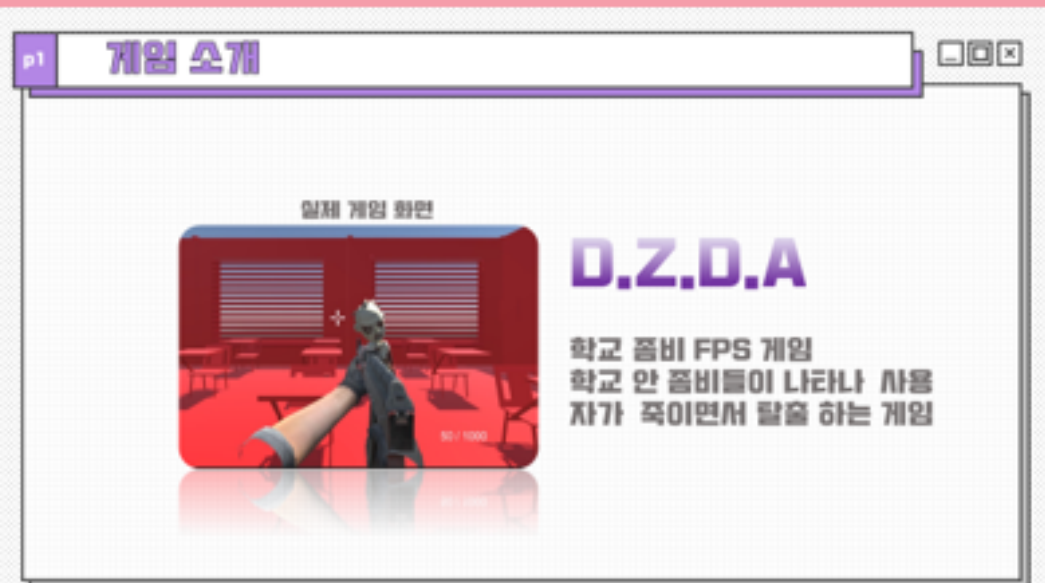

포트폴리오

김주연

1. 게임소프트웨어



처음 UNITY를 접하게 된 경험

총알 객체와 좀비 객체가 닿으면 사라지는 기능

(Raycast)

카메라 1인칭 구현 경험 (camera)

2. 현장실습인턴

현장 실습



인식 !



캐릭터 뵈 !



첫 경험 이후 흥미가 생겨 현장 실습을 함
뷰포리아를 이용하여 AR 게임 구현

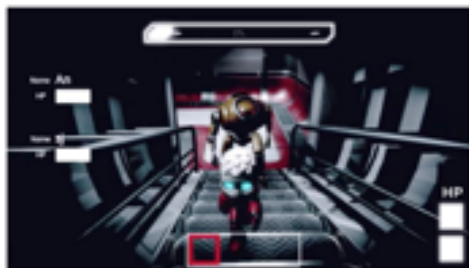
(인턴 3명과 함께 인터넷 보고 사용해봄)

초등학교 대상 교육용 게임

->이순신 역사에 관해서 배움 & 전쟁 게임 구현
<캐릭터 능력 설정 및 저장, 캐릭터 리깅 & 애니메이션, 오디오 (효과음), 외주 업무, 설정 창, 상점 구현, npc 구현, 썬 마다 캐릭터 능력 저장 >

3. 졸업작품

게임 소개



장르 | 공포 / 퍼즐 / 탈출 / SF / 멀티 / 3인칭 / 3D

플랫폼 | PC

사용 엔진 | Unity

게임 규칙 | 적을 피해가며 주어진 미션을 완수해 탈출한다.

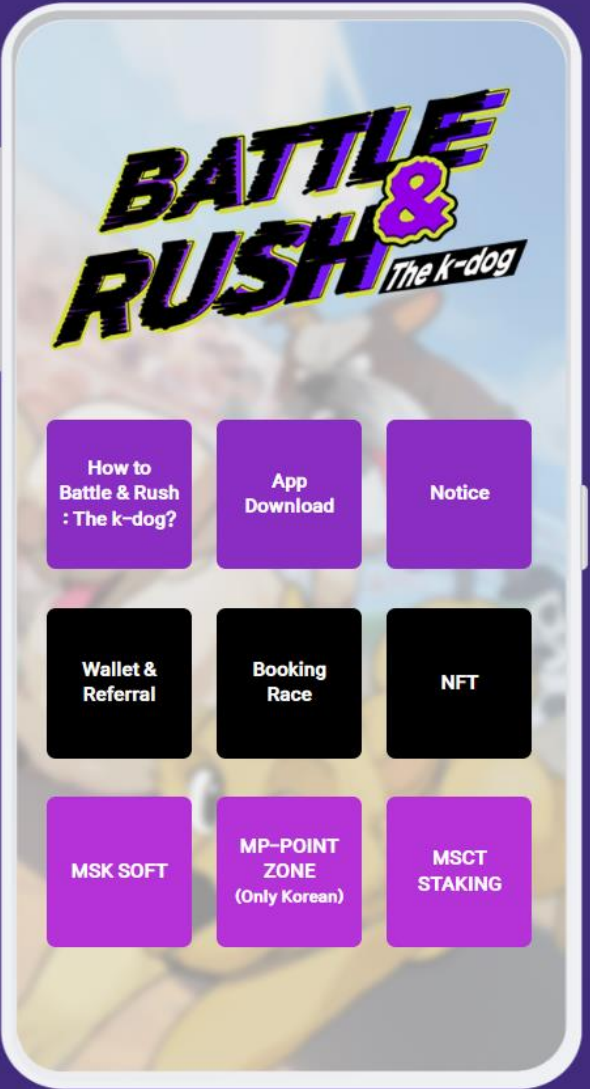





<https://kookmin-sw.github.io/capstone-2021-17/>


게임 기획 & 캐릭터 애니메이션 및 여러 능력
& 카메라 구현 (벽 충돌 방지) & 인벤토리
& 캐릭터 오디오 & 아이템 사용 시 로봇과 상호
작용 & 쉐이더 사용 (물체 넘어 상대가 보임)

4. 회사활동

<https://play.google.com/store/apps/details?id=com.app.battlenrush>








Battle&Rush(배틀앤러쉬)



MSKsoft INC
인앱 구매


4.7★
리뷰 20개

1천+
다운로드


전체이용가 ㉸

설치





Unity 게임 클라이언트 담당
- UI, 그래픽, 애니메이션, 오디오 등
전체 적인 부분 담당

4. DirectX (기술위주)

게임 소개



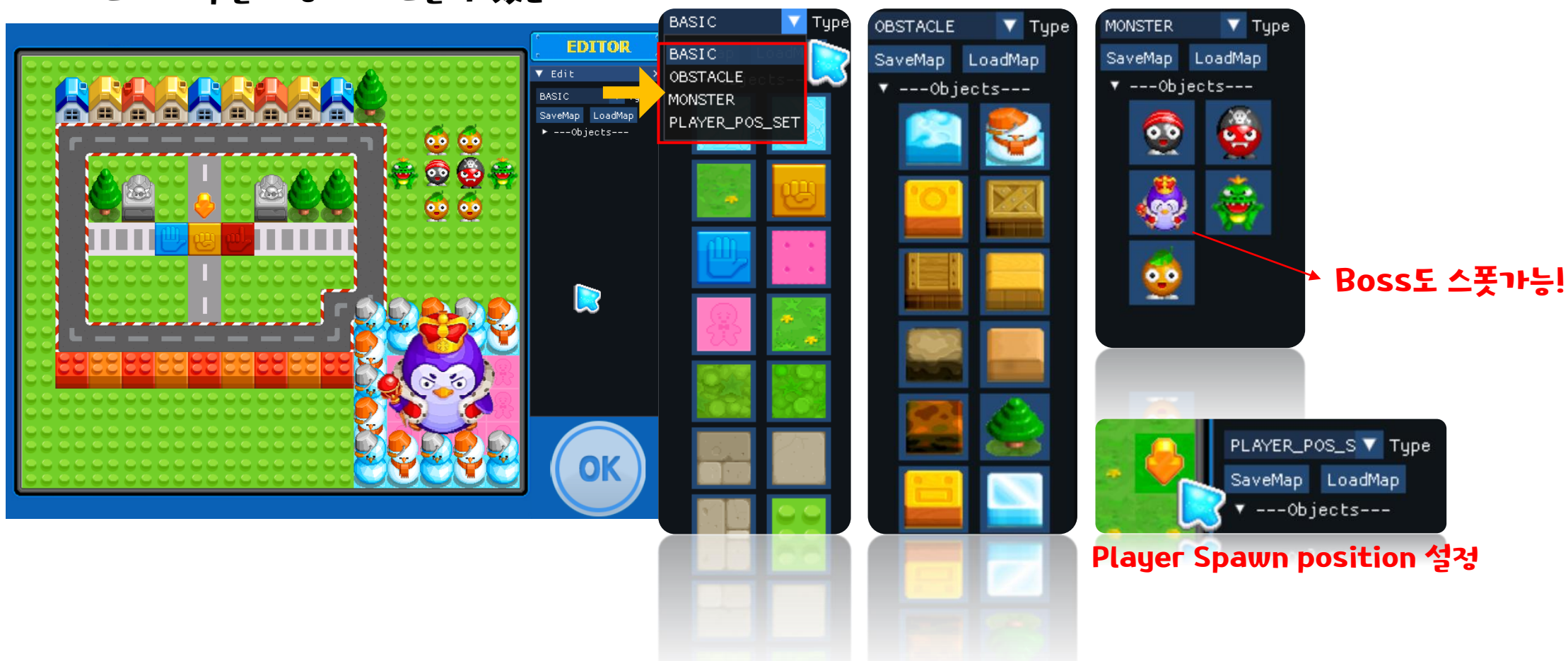
제목	Crazy Arcade (모작)
장르	액션 게임, 전략 비디오 게임
플랫폼	PC
개발환경	Visual Studio 2022
개발언어	C++, XML
라이브러리	DirectX
개발기간	2024.04.08~ 2024.05.03 (4주)

<https://cooljyum.tistory.com/37>

Map Editor

Map Editor

- Tile, Monster, PlayerPos 설정
- 나만의 Map을 사용자가 만들 수 있음



Map - Tile

Pos를 비교하여 가장 가까운 Tile 찾기

: 2중 배열의 (0,0)을 기준점을 정하고
현재의 점 거리와 계산하여
가장 가까운 거리에 있는 Tile을 바로 알 수 있다 !

`vector<vector<Tile*>> bgTiles;`



TileManager

Tile을 2중배열로 관리
: 타일 찾기 쉬워짐

```
Tile* TileManager::GetNearPosTileState(Vector2 pos)
{
    // 만약 위치(pos)가 지도 범위를 벗어난다면 nullptr 반환
    if (pos.x < mapSize["Left"] || pos.x + Tile::TILE_SIZE > mapSize["Right"]
        || pos.y > mapSize["Up"] || pos.y - Tile::TILE_SIZE < mapSize["Down"])
        return nullptr;

    // 첫 번째 타일의 위치를 가져옴 //기준이 되는 Tile
    Vector2 firstTilePos = bgTiles[0][0]->GetGlobalPosition();

    // 타일 좌표계로 변환
    Vector2 calPos = { pos.x - firstTilePos.x, firstTilePos.y - pos.y };
    calPos /= Tile::TILE_SIZE;
    calPos = { round(calPos.x), round(calPos.y) };

    // 변환된 타일 좌표에 해당하는 타일 반환
    return bgTiles[calPos.x][calPos.y];
}
```



타일 사이즈 : 100 이라 치자
현재 ☆의 위치는 (90,90)
->100으로 나누면 (0.9,0.9)
->올림처리 (1,1)

결과



Map - Tile

```
enum Type
{
    BASIC, OBSTACLE, ATTACK
};
```



Tile Type

Bubble이 Pop()하면서 Wave 스폰

→ 해당 Tile의 Type을 체크하여 BASIC을 ATTACK 변환

→ OBSTACLE 경우 ObstacleTile을 Deactivate()

→ Character는 Tile의 Type을 Check하여
ATTACK일 경우 Hit()

```
//물풍선 파도를 생성할때 상하좌우 확인하는 for문
//파도가 생성할 해당 pos의 Type이 OBSTACLE일 경우
if (tile->GetType() == Tile::OBSTACLE)
{
    BasicTile* basicTile = (BasicTile*)tile;
    ObstacleTile* obstacleTile = (ObstacleTile*)basicTile->GetObstacleTile();

    // 장애물 타일이 존재하고 활성화되어 있다면
    if (obstacleTile != nullptr && obstacleTile->IsActive())
    {
        // 장애물 타일을 비활성화 상태로 설정
        obstacleTile->Deactivate();
    }

    // 기본 타일의 장애물 타일을 제거하고 기본 타일로 설정
    basicTile->SetObstacleTile(nullptr);
    basicTile->SetType(Tile::BASIC);
    break;
}
```

```
void Monster::CheckTileHit()
{
    if (curState == DIE) return;

    //몬스터와 제일 가까운 타일을 받아온다.
    Tile* tile = TileManager::Get()->GetNearPosTileState(GetCollider()->GetGlobalPosition());

    if (!tile) return; //타일이 없으면 return

    //해당 타일이 ATTACK이면 Hit
    if (tile->GetType() == Tile::ATTACK)
        Hit(tile->GetCollider());
}
```

StageManager

StageTable

	A	B	C	D	E	F
1	key	num	type	stage1	stage2	stage3
2		0	1	1 stage1-1	stage1-2	stage1-3
3		1	2	1 stage2-1	stage2-2	stage2-3
4		2	3	1 stage3-1	stage3-2	stage3-3

StageManager

- StageTable을 받아 맵 정보를 띄움
선택하면 해당 맵 정보를 받아 게임 스테이지 관리



StageManager.h

```
void LoadStage();  
void NextStage();  
void SpawnPlayer(bool isSpawn = false);  
  
void Start();  
void Clear();  
void Gameover();
```

Render

Render Manager

타일, 몬스터, 플레이어
Depth 대로 Sort하여 화면 출력

RenderManager X



RenderManager O



최적화

RenderTarget - bgTiles

```
void TileManager::CreateRenderTarget()  
{  
    //최적화를 위한 RenderTarget  
    bgTileTarget = new RenderTarget();  
  
    Texture* targetTexture = Texture::Add(L"BGTileMap", bgTileTarget->GetSRV());  
  
    //RenderTarget을 띄울 이미지  
    bgTileMap = new Quad(Vector2{ SCREEN_WIDTH, SCREEN_HEIGHT });  
    bgTileMap->GetMaterial()->SetTexture(targetTexture);  
    bgTileMap->SetLocalPosition(CENTER);  
    bgTileMap->UpdateWorld();  
  
    RenderManager::Get()->Add("BGTileRender", bgTileMap);  
}  
  
void TileManager::SetRenderTarget()  
{  
    //렌더할 이미지 Set()  
    bgTileTarget->Set();  
    RenderManager::Get()->Render("BG");  
    RenderManager::Get()->Render("BGTile");  
}
```

bgTileMap



FPS : 41 → FPS : 116

화면에 그리는걸 최소화 시켜서 **Frame 상향**

몬스터 및 보스

Basic Monster



Hit Monster



Boss - Bubble Monster



Angry모드 - Attack



MonsterTable

key	name	speed	isBubble	hp	isBoss	type
0	Jjoljjol-i	20.0f	0	1	0	0
1	Beoleog-i	30.0f	0	1	0	0
2	Pudadag	30.0f	0	2	0	0
3	Gamgyul-i	20.0f	0	1	0	0
101	Penguin	40.0f	1	4	1	1

Monster

- 각자 고유 key값을 주어져 데이터를 불러와 생성

소감

충돌 처리와 최적화의 중요성

크레이지 아케이드 게임은 빠른 속도와 고밀도의 오브젝트들이 서로 충돌하는 것이 흔하기 때문에, 정확하고 효율적인 충돌 감지 및 처리가 게임에 버그 없이 들어가야 한다고 생각했습니다. 또한, 게임이 화면에 표시하는 내용이 많지 않을 것으로 예상했지만, 실제로는 객체 풀링이나 렌더 타겟과 같은 최적화 기술이 중요하다는 것을 깨달았습니다. 이를 통해 게임의 성능을 향상시키고 원활한 플레이에 경험을 제공할 수 있습니다.

타일 관리와 객체 상호작용

타일을 처음에 일반 배열로 했다가 타일의 배치 및 위치를 동적으로 조작해야 하는 경우에는 복잡성이 증가하는데 효율적인 타일 관리 및 검색 알고리즘이 필요할 때 이중 배열로 관리하는 것이 편하다는 걸 알고 변경했습니다. 이중 배열로 변경함으로써, 게임의 성능을 향상시키고 유지보수성을 향상시킬 수 있었습니다.

데이터의 중요성

데이터의 저장과 불러오기는 게임의 핵심 부분으로, 이를 효과적으로 관리함으로써 게임의 진행 상황을 영구적으로 보존할 수 있습니다. 이를 통해 게임을 종료한 후에도 자신의 맵 데이터를 유지할 수 있어 사용자, 개발자 입장에서든 편리함을 느낄 수 있습니다.

5. 기타

이외에....

앱인벤터를 이용한 어플 만들기

라즈베리 파이로 rc카 트랙 제작 및 실습

구글 코랩 머신러닝 실습

자바 스크립트로 웹 페이지 제작 실습

과천 과학관 동영상 공모전 스태프

과천 과학관 로보메이션 햄스터 축구

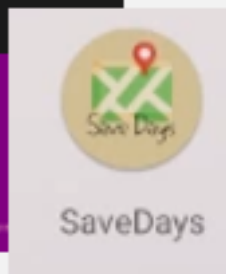
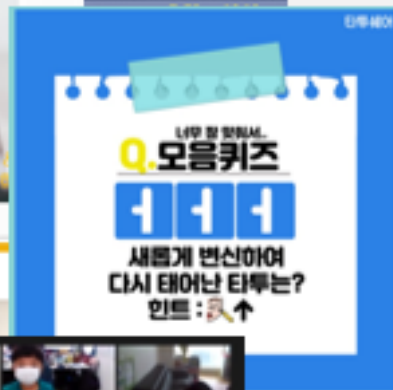
학교 과 동아리(SW 교육 동아리) 활동으로 초중고 SW캠프 진행

타투 어플 회사 현장실습으로 마케팅 진행

안드로이드 지도 어플 제작 경험

사용 해본 언어: python, java, javascript, c, c++, c#

그 외 등등 ... 여러 아르바이트 경험 해봤습니다.



감사합니다
