메이플스토리 모작 개발노트

#깃허브에 올려놓은 내용을 좀 더 구체적으로 기록하기 위해 문서를 제작함.

(https://github.com/seongjune2003/Maplestory-Unity.git)

모작 취지

- 게임개발을 진로로 결정한 후에 게임개발을 위해 준비해야 하는 것을 우선적으로 찾아보았으나 처음부터 1인, 소규모 팀 개발을 하기에는 해당 분야 초심자이기에 실력이 부족하고 아트까지 처음부터 배워서 사용하기에는 너무 많은 시간이 소요됨.
- 그래서 내가 지금 하고 있거나 과거에 했었던 게임을 따라 만들어보면서 개발 실력을 늘려보려고 함.
- 모작 대상으로 메이플스토리를 선정한 이유는 위컴알 툴로 클라이언트를 뜯어서 원하는 리소스를 뽑아낼 수 있다는 이유가 가장 컸음.
- 그리고 기존에 C++과 DirectX로 제작된 메이플스토리의 기능들을 유니티로 구현하려 한다면 필연적으로 구현 로직을 고민하고 연관 코드를 검색하고 그것이 제대로 작동하도록 코드를 수정해야 함. 이 과정이 실력 향상에 도움이 많이 될 거라 판단해 모작을 시작함.

프로젝트 관리

1. 리소스 수집 및 정리
2. 캐릭터 움직임, 이동 구현
3. 카메라 무빙 구현
4. 적 몬스터 제작, 상호작용 (충돌 구현)
5. 일반 공격 및 피격 구현
6. 메인화면 제작
7. 마우스 구현
8. 맵 제작
9. 애니메이션 처리

10. 스킬 구현
11. 몬스터 AI 구현
12. 물약 및 인벤토리 구현

2022년 08월 12일 포트폴리오 제작 시작

8월 12일(금) 작업

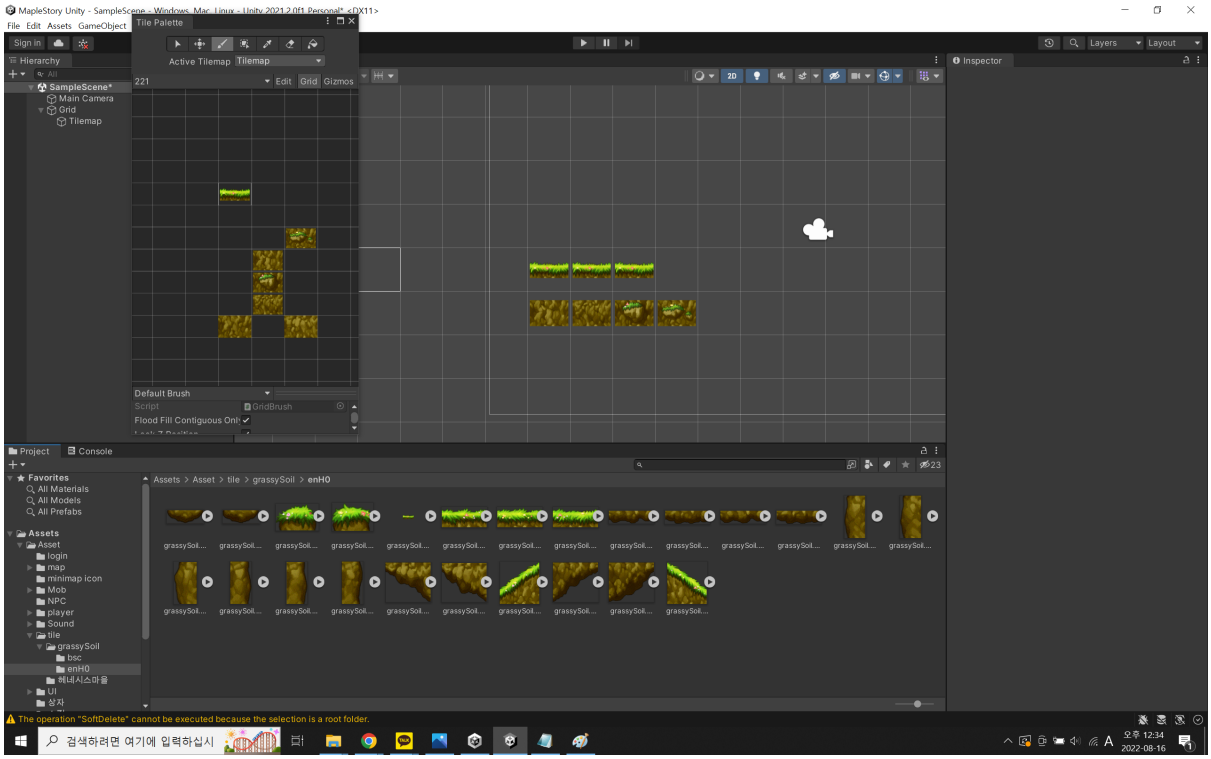
1. 메이플스토리 클라이언트에서 필요 리소스 추출 (캐릭터, 몬스터, 배경 이미지, 사운드)

8월 14일(일) 작업

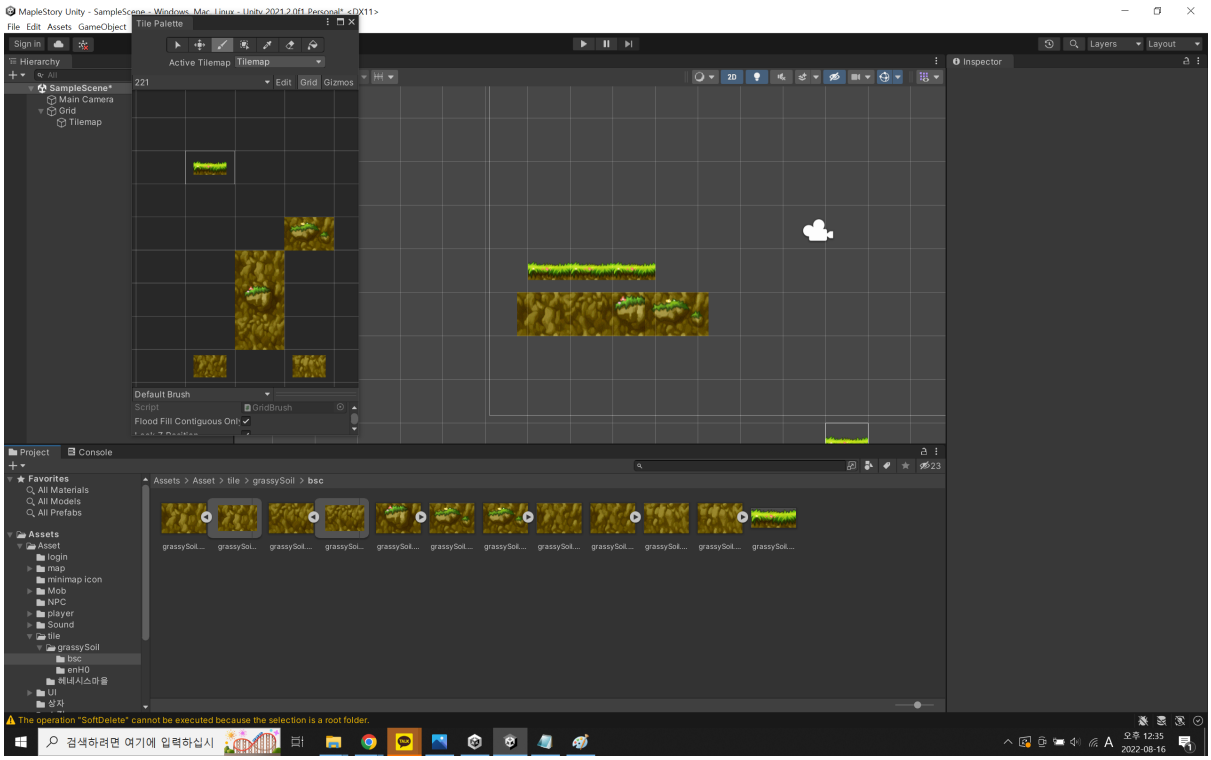
1. 클라이언트에서 리소스 추출 (UI, 스킬 이미지) → 맵 타일 이미지에서 특정 파일(사다리, 로프) 이미지의 위치를 찾지 못해서 시간이 오래 걸림.

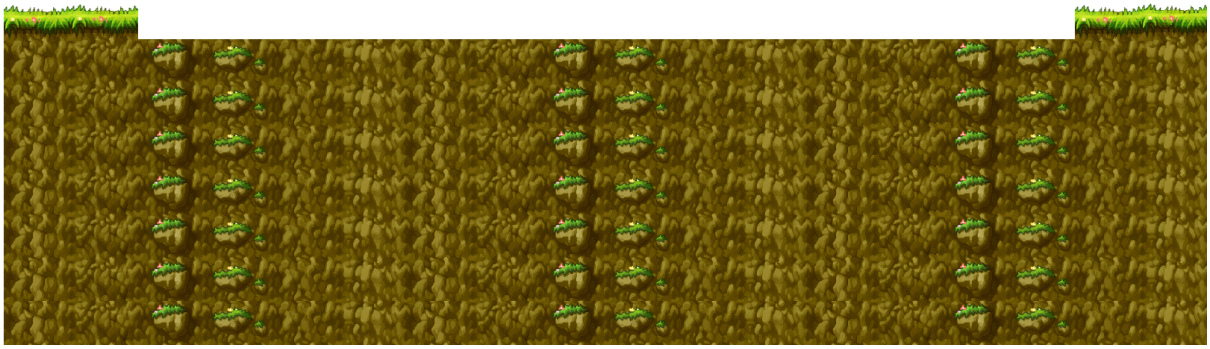
8월 16일(화) 작업

- 추출한 리소스로 유니티 맵 제작을 시작함.
- 추출한 리소스 파일 중에 맵 이미지 파일들은 전부 쪼개져 있었음. → 이유를 찾아보니 하나로 통합된 이미지를 한번에 불러오려고 하면 로딩에 시간이 많이 소요된다고 함. 이미지 파일을 파츠 별로 쪼개고 분할된 이미지를 불러와서 조합하는 것이 드로우콜을 줄이는 방법으로 쓰임
- 이에 맵 제작에 앞서서 분할된 이미지 파일을 스프라이트 아틀라스로 하나로 합치는 과정이 필요했음. 이에 대한 자세한 내용은 '게임 공부 - 그래픽에 대한 이해' 페이지에 기록함.
- 타일맵으로 맵을 제작하려고 하였으나 이미지 파일 크기가 다 달라서 전부 같은 크기로 넣어야하는 타일맵에서는 사용이 불가능했음.

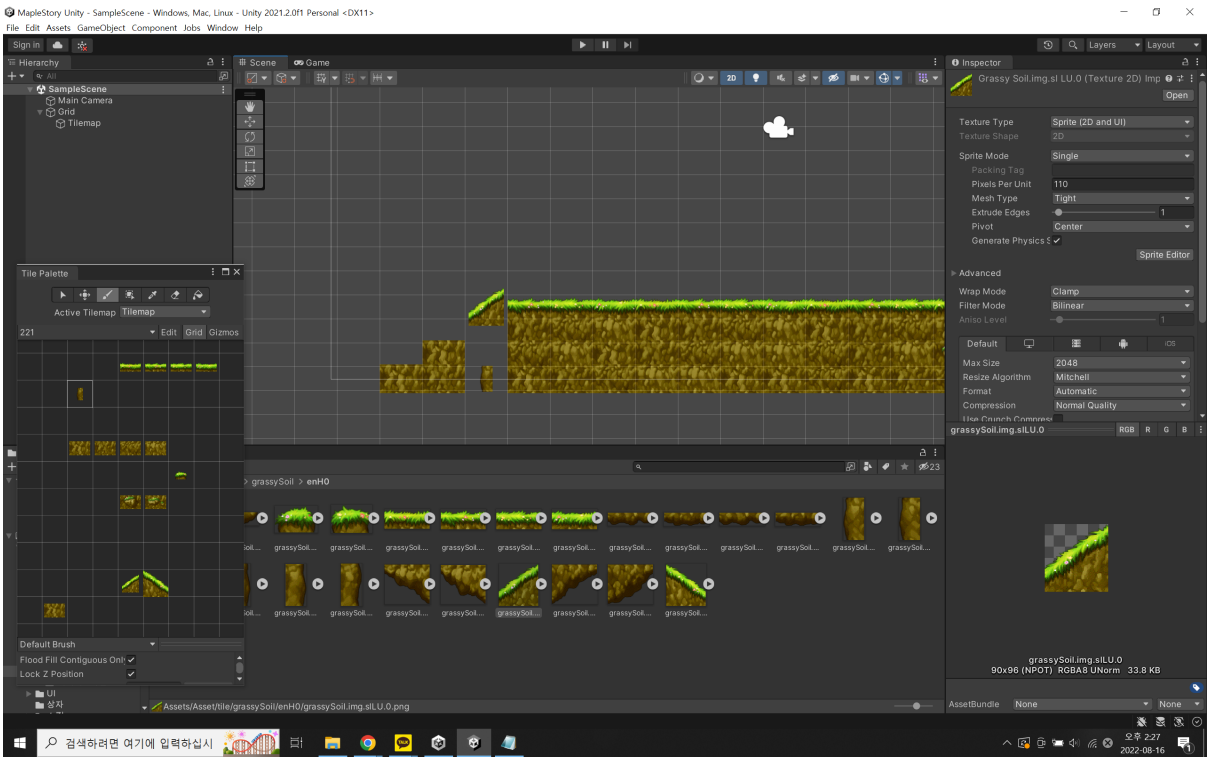


- 처음에 타일 팔레트로 입력했을 때 저런 식으로 이미지 크기의 원본 크기만큼 타일의 정중앙을 중심으로 배치가 되어있었음. 타일 한 칸을 가득 채우는 것도 아니었고 아래 타일과 윗 타일이 접해야 하는데 접하지도 않음.



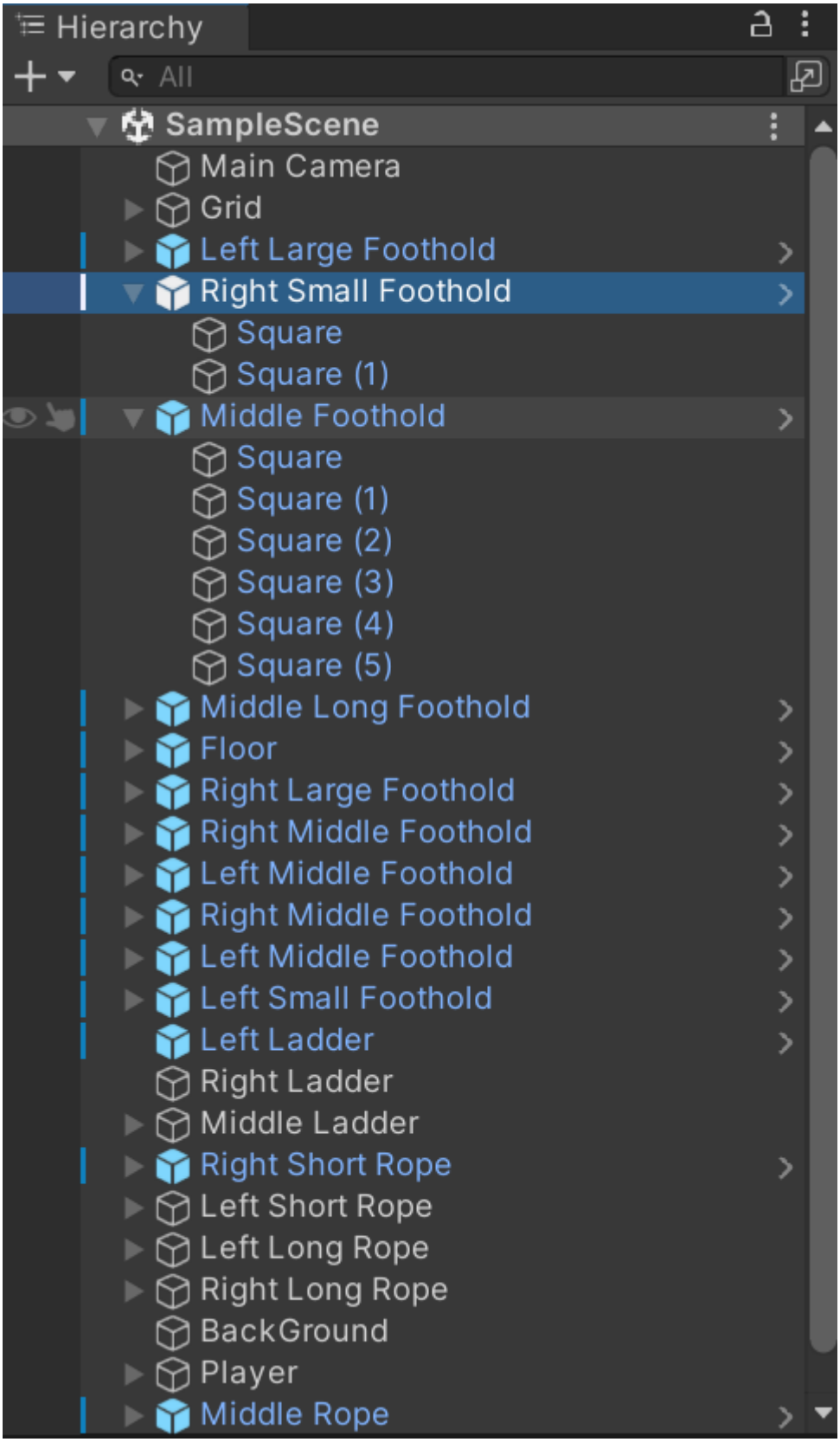


- 이에 타일을 입력하기 전에 타일의 비율로 조정해 원본 크기에 비해 크게 만들어 서로 겹치게 하여 자연스럽게 만들어 보고자 하였음. 그렇게 해 위 이미지와 같은 형태까지 만드는 데는 성공하였으나



- 이미지 파일 크기가 더 큰 언덕 길 이미지를 붙였을 때는 크기 차이 때문에 자연스럽지 않고 연결되지 않았다.
- 게다가 이미지 별로 정사각형, 직사각형 형태 등 이미지의 모양이 서로 다 다르다는 문제가 있었음. 타일맵의 기본형은 정사각형인데 직사각형 형태 이미지가 제대로 들어가지 않아서 기본형을 직사각형 형태로 바꾸는 방법을 찾았으나 그렇게 되면 반대로 정사각형 이미지를 제대로 넣지 못했음.
- 타일맵 각각마다 이미지의 모양을 다르게 줄 수 있는 방법이 없기 때문에 타일맵을 사용하려면 추출한 리소스 이미지를 전부 정사각형 형태로 변형시켜 사용해야 했음. → 하지만 그렇게 되면 시간이 너무 많이 소요되고 깔끔하게 처리할 수도 없었기에 새로운 방법을 고민해봄.

8월 17일 (수)



1. 기존에 리소스를 타일 팔레트에 직접 넣는 방법 대신, 게임 오브젝트를 생성하고 스프라이트를 입혀서 하나씩 붙이는 방식으로 맵을 제작. 발판 하나 당 구성 요소들을 자식 요소로 할당함.

2. (오류) Create Empty로 배경 생성 → 이미지 파일을 배경으로 까니 캐릭터가 이동할 때마다 조합한 타일 경계선에 배경색이 비쳐 보이는 현상이 발생.
3. 충돌 처리를 위해 생성된 타일에 box collider 생성

8월 18일

캐릭터 움직임 구현 시작

1. 애니메이션 제작을 위해 분할된 애니메이션 프레임 이미지들을 스프라이트 아틀라스로 합침 → 드로우콜 감소효과
2. PlayerMovement 스크립트를 생성하고 Input.GetAxis()으로 좌우 방향 이동 구현
3. 점프 구현 → 에러발생. 점프키를 누르고 있을 때 무한히 점프가 가능한 버그가 발생함.

- jumpCount, isJumping, isGrounded 변수를 선언.

```
75
76      // Update is called once per frame
        Event function
77      void Update()
78      {
79          if (isJumping == false && jumpCount > 0)
80          {
81              if (Input.GetButtonDown("Jump"))
82              {
83                  isJumping = true;
84                  jumpCount = 0;
85                  isGrounded = false;
86                  // 점프 애니메이션 실행
87                  ChangeAnimationState(Player_Jump);
88              }
89          }
90      }
91
```

```
179
                Frequently called  1 usage
180         void Jump()
181         {
182             if (isJumping == true)
183             {
184                 rigid.velocity = Vector2.zero;
185
186                 Vector2 jumpVelocity = new Vector2( x: 0,  y: jumpPower);
187
188                 rigid.AddForce(jumpVelocity, ForceMode2D.Impulse);
189
190
191                 isJumping = false;
192             }
193         }
194
```

```
                Event function
233         private void OnCollisionEnter2D(Collision2D col)
234         {
235             if (col.gameObject.CompareTag("Ground"))
236             {
237                 jumpCount = 1;
238
239                 attackCount = 1;
240
241                 isGrounded = true;
242             }
243         }
244
```

- jumpCount가 1 이상, isGrounded == true일 때, isJumping이 false일 때 점프키를 누르면 점프가 가능하도록 조건을 부여.
- 점프를 하면 jumpCount를 0, isGrounded를 false isJumping을 true로 설정해 이중점프가 안되게 구현.
- OnCollisionEnter2D로 Ground 태그를 가진 콜라이더와 충돌 시 세 요소를 리셋함 (문제 해결) → ## 또 다른 버그 발판을 위에서 아래로 떨어져 밟았을 때만 해당 기능이 발생해야 하는데 아래에서 위로 점프하다 발판에 닿은 경우, 발판의 옆면에 닿은 경우에도 점프가 가능한 문제가 존재함. (해결 방법 탐색 필요)

- 캐릭터가 왼쪽으로 이동할 때 왼쪽을 봐야하고 오른쪽으로 이동할 때 오른쪽을 봐야하는데 계속 왼쪽만 보는 문제가 있었음

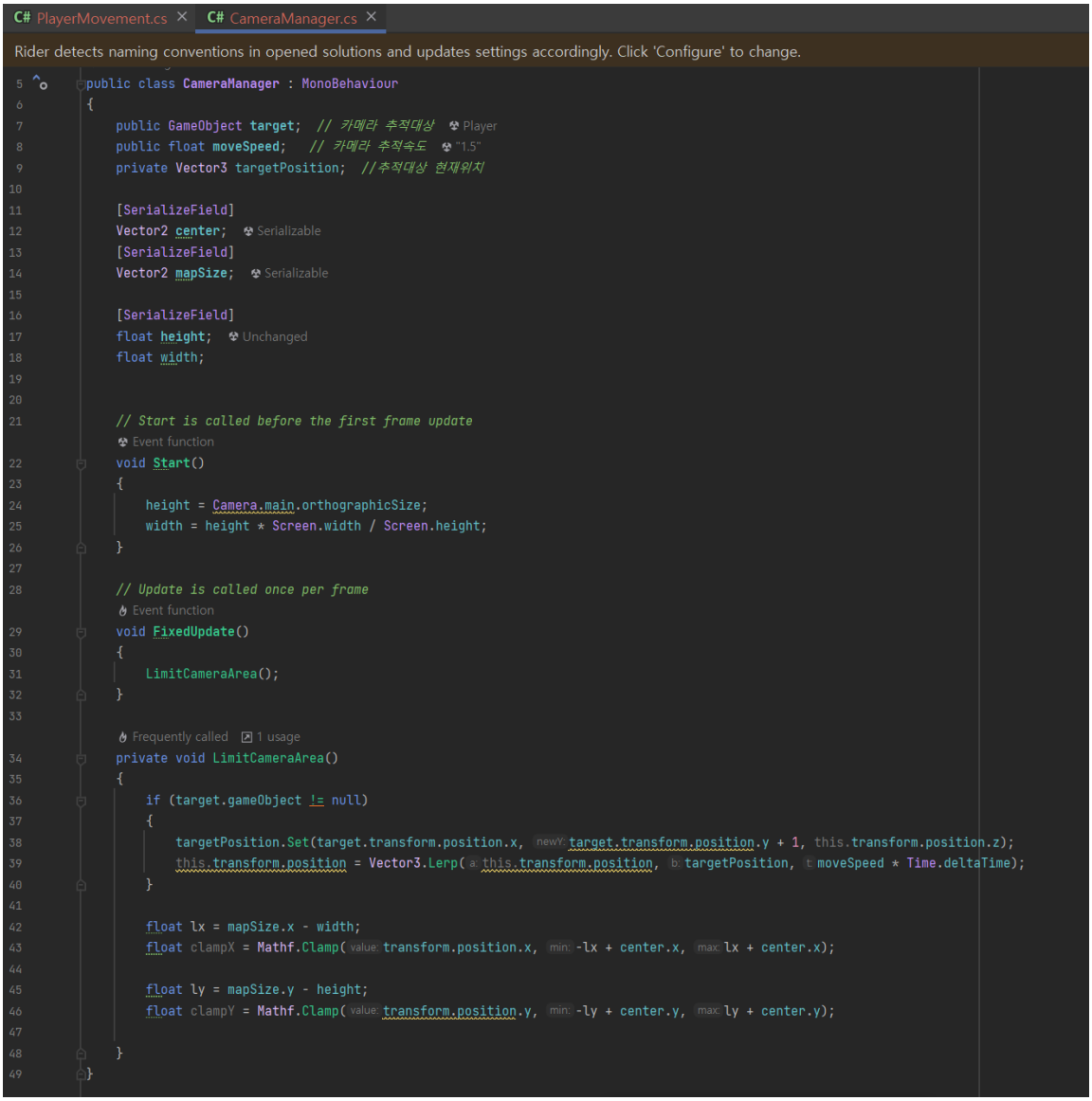
```
148
        Frequently called  1 usage
149     void Move()
150     {
151         Vector3 moveVelocity = Vector3.zero;
152         if (Input.GetAxisRaw("Horizontal") < 0)
153         {
154             rend.flipX = false;
155             moveVelocity = Vector3.left;
156         }
157
158         else if (Input.GetAxisRaw("Horizontal") > 0)
159         {
160             rend.flipX = true;
161             moveVelocity = Vector3.right;
162         }
163         transform.position += moveVelocity * movePower * Time.deltaTime;
164
165         //
166         if (isGrounded && !isAttacking)
167         {
168             if (Input.GetAxisRaw("Horizontal") != 0)
169             {
170                 ChangeAnimationState(Player_Walk);
171             }
172             else
173             {
174                 ChangeAnimationState(Player_Idle);
175             }
176         }
177     }
178
179
```

- Sprite Render의 flip으로 좌우 반전이 가능한 것을 확인. → 코드로 Sprite Render에 접근해서 좌우 방향키를 누를 때 캐릭터 스프라이트 이미지가 뒤집어지도록 구현.
- 정지(Idle)상태와 걷기(walking)상태로 전환 시 애니메이션 전환 구현
- 유니티 엔진 이벤트 함수 업로드 순서에 대한 추가 공부 필요 (어느 부분에 넣어야 하는지 판단하느라 삽질 많이 함)
    - Update와 FixedUpdate의 차이

- Update - 매 프레임마다 호출. 물리효과가 적용되지 않는 오브젝트 관리나 타이머, 키입력을 받을 때 사용됨.
- FixedUpdate - 물리효과가 적용된 오브젝트 관리할 때 쓰임

8월 19일

1. 정지 → 점프, 점프 → 정지, 이동 → 점프, 점프 → 이동 애니메이션 구현
2. 카메라 구현



- 카메라 추적 대상, 추적속도, 추적대상의 현재 위치를 설정하고 카메라가 캐릭터를 추적하도록 설정

- Vector3.Lerp로 부드러운 카메라 이동을 구현

3. 엎드리기 구현

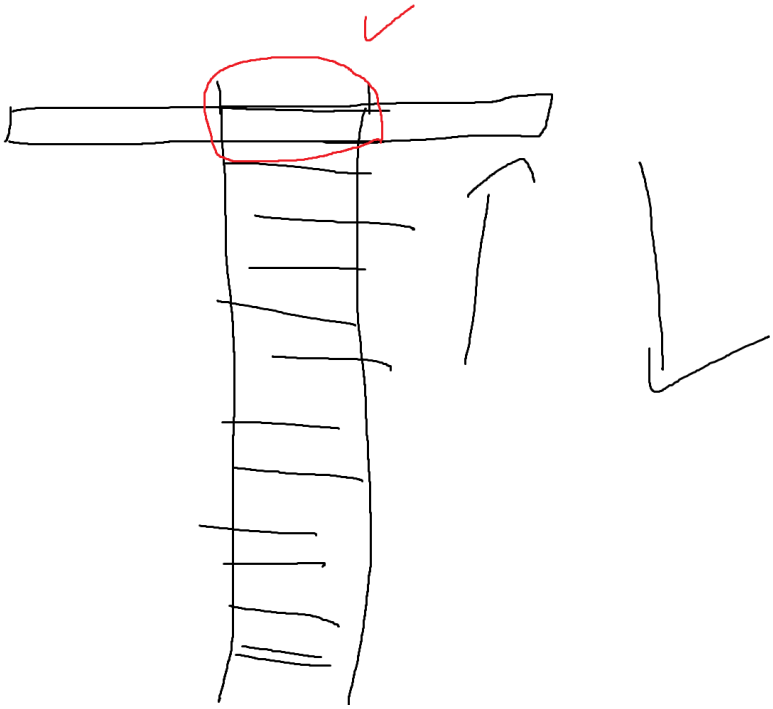
   → 바닥이 아니라 공중에 엎드리는 버그 발생

   → 엎드렸을 때 이동, 점프가 되는 버그가 발생

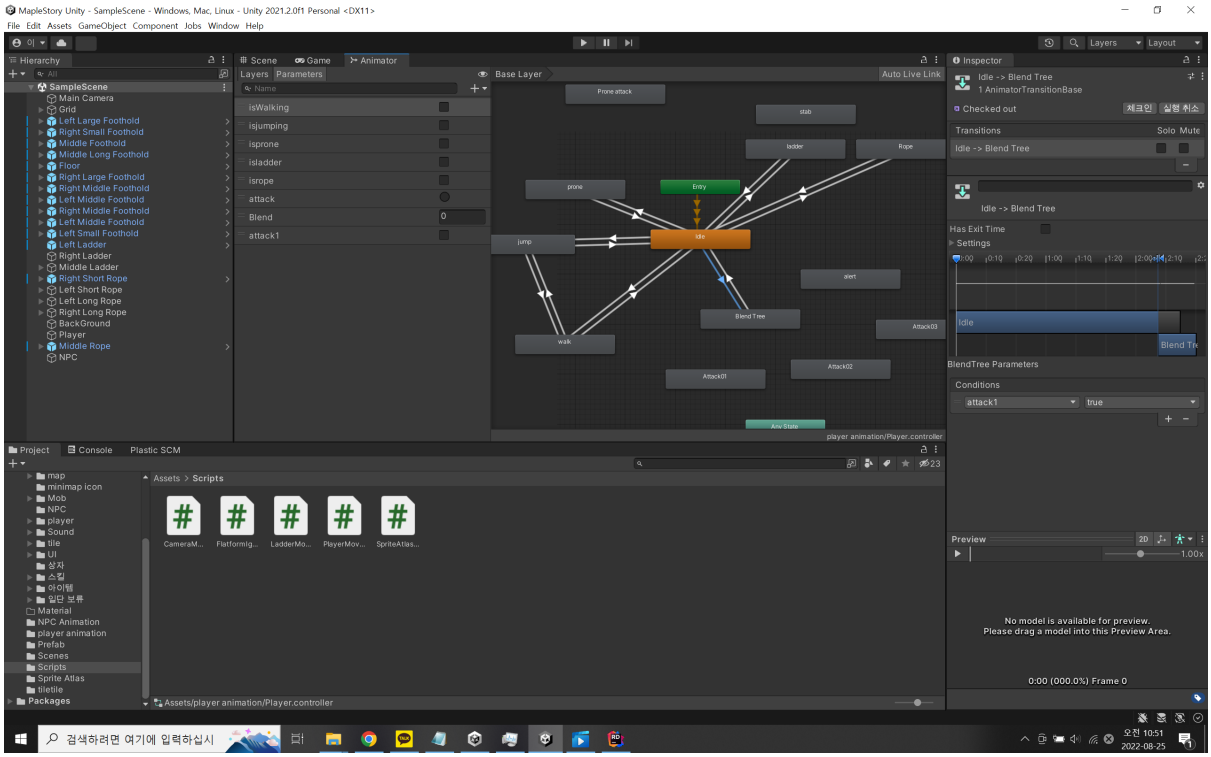
4. 사다리, 밧줄 애니메이션 구현
   - 사다리와 발판이 겹친 부분에서 발판의 Collider에 충돌되서 캐릭터가 더 이상 올라가지 않는 버그가 발생.
   - 바닥에서 밧줄로 오를 때 갑자기 점프를 하는 버그 발생.
   - 사다리, 밧줄에 매달려있을 때 상, 하 방향 키를 누르지 않아도 애니메이션이 계속 실행되는 버그 발생 → 사다리에 매달린 채로 이동을 멈추면 애니메이션도 멈추게 수정이 필요해 보임.

8월 22일

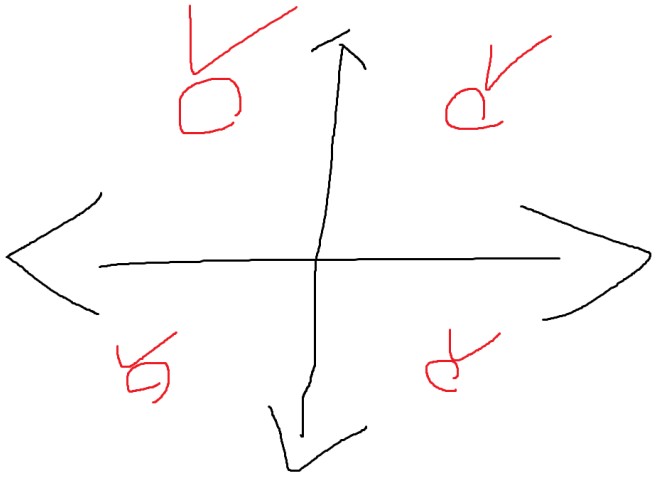
1. 밧줄에 오를 때 오르기가 되지 않고 처음 부분에서 무한 점프하는 버그 해결
2. 사다리를 오를 때 위(발판)의 collider와 충돌하여 올라가지 못하는 현상 해결
   - 바닥에 platform effector 2D를 줘서 아래에서 위로 통과할 수 있도록 하고, ladder에 스크립트를 부여 (OntriggerStay, OntriggerExit로 사다리에 들어갈 때 나갈 때 트리거가 발동하게 하고 발판 콜라이더와 충돌했을 때 ignoreCollision으로 충돌 처리를 무시하게 설계함. → 수정 후 위아래로 이동 가능해짐)
   - 해당 버그는 해결했으나 또 다른 버그가 발생. 사다리를 타고 사다리와 연결되어 있는 발판 콜라이더와 충돌 시 콜라이더를 무시하게 하니 해당 발판부분에 가면 발판의 해당 부분에 가면 밑으로 떨어지는 버그 발생
   - 하단 이미지의 빨간 동그라미 부분에 캐릭터가 가면 아래로 떨어져 버림. (다른 방법으로 구현하던지 아니면 조건을 더 주는 식으로 수정이 필요해 보임)

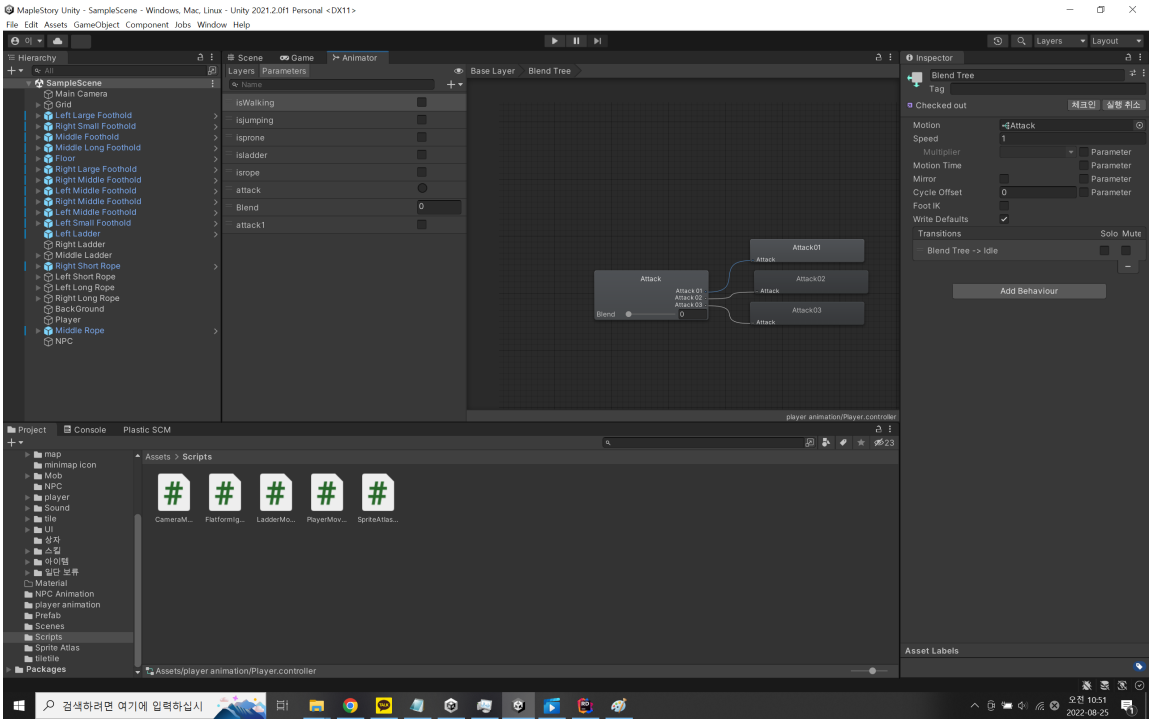


8월 23일


MapleStory Unity - SampleScene - Windows, Mac, Linux - Unity 2021.2.0f1 Personal <DX11>
File Edit Assets GameObject Component Jobs Window Help
Hierarchy
SampleScene
Main Camera
Grid
Left Large Foothold
Right Small Foothold
Middle Foothold
Middle Long Foothold
Floor
Right Large Foothold
Right Middle Foothold
Left Middle Foothold
Right Middle Foothold
Left Middle Foothold
Left Small Foothold
Left Ladder
Right Ladder
Middle Ladder
Right Short Rope
Left Short Rope
Left Long Rope
Right Long Rope
BackGround
Player
Middle Rope
NPC
Scene
Game
Animator
Layers
Parameters
Base Layer
Auto Live Link
isWalking
isjumping
isprone
isladder
isrope
attack
Blend
attack1
Prone attack
stab
ladder
Rope
prone
Entry
jump
Idle
alert
Blend Tree
Attack03
walk
Attack02
Attack01
Any State
player animation/Player.controller
Inspector
Idle -> Blend Tree
1 AnimatorTransitionBase
Checked out
체크인
실행 취소
Transitions
Solo Mute
Idle -> Blend Tree
Has Exit Time
Settings
Idle
BlendTree Parameters
Conditions
attack1
true
Project
Console
Plastic SCM
Assets > Scripts
map
minimap icon
Mob
NPC
player
Sound
tile
UI
상자
스킬
아이템
일단 보류
Material
NPC Animation
player animation
Prefab
Scenes
Scripts
Sprite Atlas
tietile
Packages
Assets/player animation/Player.controller
Preview
No model is available for preview.
Please drag a model into this Preview Area.
0:00 (000.0%) Frame 0
검색하려면 여기에 입력하십시
오전 10:51
2022-08-25

1. 일반 공격모션 애니메이션을 구현하려고 했음. 하지만 내가 사용하고자 하는 일반 공격 모션은 총 3가지로 1/3 확률로 하나의 애니메이션이 나오도록 구현해야 했음.
    - Idle(stand) → attack 1,2,3 attack 1,2,3 → Idle (서 있을 때 공격)
    - walk → stand → attack 1,2,3 attack 1,2,3 → stand → walk (걷기 모션일 때 공격키를 누를 경우)
    - jump → attack 1,2,3, → Idle(땅에 다시 닿았을 때) (점프 공격의 경우)
    - 이런 식으로 공격모션 1, 공격모션 2, 공격모션 3에 연결되는 애니메이션을 메카님으로 연결해주는 것만 해도 변수가 많음. → 애니메이션 변수가 적으면 수동으로 할 수 있으나 변수의 수가 많아지면 그 경우의 수가 엄청나게 많아짐. 이를 일일히 웹으로 연결해서 처리하면 처리하기도 힘들고 유지보수 비용도 높음. 이에 일일이 연결하는 방법이 아닌 다른 방법이 없는지 고민하며 계속 찾아보았음.





- 구글링하는 과정에서 블렌드 트리라는 개념을 참고함. 3D 게임에서 X, Z축 기준으로 좌우상하 방향으로 이동하는 애니메이션은 수동으로 처리한다 하더라도 위 사진의 빨간 동그라미가 있는 방향인 대각선 방면까지 모두 애니메이션을 수동으로 처리해주는 것은 불가능에 가까움. (모든 각도에 맞는 애니메이션을 주는 것은 힘들다)
- 이것을 블렌드 트리로 묶어서 이동키 입력값에 따라 이동 수치를 알아서 조정해주고 그에 맞는 애니메이션이 출력하는 것이 가능해짐. 이 부분에서 아이디어를 받아서 공격모션 구현을 해결함.
- 공격모션 1, 2, 3을 하나의 Blend Tree로 묶고 공격모션 1, 2, 3에 각각 해당하는 수치값 (Threshold) 0, 1, 2를 부여함. → 공격키(x)를 누르고(GetKey) 일반공격 애니메이션이 재생중이 아닐 때 attack 트리거 발생. → Random.range로 0부터 2까지 정수 값을 Blend값에 부여. 해당하는 애니메이션을 출력하도록 설계함.
- (에러발생)

  → 공격모션 3개 중 하나가 랜덤으로 나오는 것은 확인했으나 공격키(x)를 누르고 있을 때 공격 애니메이션 첫 번째 프레임이 무한 반복됨 (애니메이션이 끝까지 재생되지 않고 처음 프레임만 무한 재생)
- (에러발생)

  → 공격버튼을 뗐을 때 공격모션이 끝나고 Idle 상태로 돌아가야 하는데 공격 모션의 마지막 프레임에서 캐릭터가 멈춰버림. → attackend 트리거가 발동하지 않은

것으로 추정됨.

- (에러해결)
  1. attack 함수에서 GetKey를 사용하는 것을 GetKeyDown으로 수정. -> 입력을 한번 받는 걸로 변경함.
  2. attack 함수 내 하단에 animator.GetCurrentAnimatorStateInfo(0).normalizedTime >= 1.0f 으로 애니메이션 재생이 끝났을 때 bool 파라미터가 꺼지도록 설정함 => 이렇게 했더니 원하는 대로 공격하고 Idle 상태로 돌아오긴 했으나 공격키(x)를 눌렀을 때 될 때도 있고 안될 때도 있었음. (유니티 내부 연산 렉?)
  3. 저녁 때 다시 켜서 확인해보니 해당 에러없이 제대로 작동함 → 유니티 오래 키고 있으면 렉이 있다던데 그게 문제의 원인?

== 해결!!!!!!

해결 후 느낀 점 => 데이터 관리에 대해 다른 방법으로 접근할 방법이 필요하다는 것을 느낌 -> 싱글톤, 옵저버(데이터 디자인 방식) 공부 필요 -> 비트연산 등 컴퓨터 구조에 대한 공부 필요 -> 최적화에 대한 이론적 배경으로 쓰기 위해서 -> 스킬 목록을 리스트로 정리하고 필요할 때 추출하는 방법을 사용해야 할 것 같음

- 애니메이션 클립 중 연관된 것들끼리 블렌드 트리로 묶어서 처리하면 좀 더 효율적으로 관리가 가능하다는 것을 체감함.

(남은 에러) 공격 중일 때 공격키를 한번 더 누르면 실행되던 공격 모션이 꺼지고 애니메이션이 공격모션이 처음부터 다시 시작 => 그래도 금방 해결할 수 있는 문제인 듯.

8월 25일

- 8월 23일에 발생했던 오류가 다시 발생함. 공격이 됐다가 안됐다가 함. 되는 조건이나 안되는 조건을 모르겠음. bool parameter에서 true나 false가 일부 상황에서 작동이 안되는 것이 원인? 아니면 bool이나 trigger로 Blend Tree로 이동을 받아 놓고 Blend Tree 내부에서 공격모션 3개 중에 하나를 선택하는 과정이 병목을 발생시킨 것이 원인인가? 아니면 내가 조건을 잘못 주었나?
- 메카님은 변수가 많아지면 그걸 전부 다 연결하기도 힘들다. (거미줄이 생김) 지금은 애니메이션 클립이 10개 남짓이라 경우의 수가 적지만 만약 갯수가 40개에 1대1로 연결해도 천 개가 넘는 애니메이션을 모두 웹으로 연결해주고 코드로 처리해줘

야 한다. 만약 애니메이션 사이에 중간 애니메이션 클립이 더 있다면 그 숫자는 더욱 늘어난다.

- 게다가 유니티 애니메이터에서 any state에 연결하면 자기 자신 스테이트도 any state로 취급되서 exit time 시간 동안 무한 재생되는 버그 애니메이션이 탄생하게 된다. (23일 날 무수히 봤던 버그의 원인인 듯)
- 그래서 결론은? → 메카님 말고 코드로 애니메이션 관리하자. → 처음에 AnimationState 선언하고 ChangeAnimationState 함수 만들어서 애니메이터에 재생할 애니메이션 재생하게 관리 해당 내용 관련 링크 (https://www.youtube.com/watch?v=nBkiSJ5z-hE&t=874s&ab_channel=LostRelicGames)
- 그로 인한 장점은? → 번거로운 트랜지션이나 파라미터 관리를 안 해도 된다.
- 지금 해야 하는 것은? → 트랜지션이랑 파라미터 다 지우고 스크립트에서 코드로 애니메이션을 구현하도록 다시 설계
- 메카님에서 공격 모션 3가지를 블렌드 트리로 효율적으로 처리하려고 했던 것을 코드로 구현하려면 다른 구현 방법을 찾아야 한다.

8월 26일

- 사다리 스크립트 수정한 이후부터 이동모션, 점프모션이 안됨 (애니메이션 재생은 안되는 상태로 캐릭터 좌표 이동만 함) -> 이동 애니메이션이랑 점프 애니메이션을 Idle 애니메이션이 오버라이딩 한 것이 아닌가 추정되는데 정확한 원인은 아직 모르겠음

->해결방법 도출 중

- PlayerMovement 스크립트에서 함수를 하나씩 빼면서 해당 동작 작동 여부 확인하는 방식으로 디버깅 중.
- 점프 애니메이션이 안되는 원인을 찾음 -> prone(엎드리기) 함수에서 아래 방향 키가 안 눌렸을 때 Idle 상태로 변경되게 설정한 것이 걷기나 점프 애니메이션이 작동할 때 오버라이딩 되버린 것이 원인. (커플링 문제) (아래 방향 키를 눌렀을 때 prone상태가 되고 뗐을 때 Idle 상태가 되도록 설계했는데 점프, 이동 상태는 원래 아래 방향 키를 누르지 않으므로 Idle 애니메이션이 오버라이딩 해버린 것)
- 그래서 어떻게 할 것인가? => 기존 코드는 메모장에 옮겨 두고 함수 하나 두개씩 커플링하면서 에러가 안 뜨게 구현해보고 있다.

- 이동 애니메이션이 안 뜨는 오류 원인 또 찾음 => 새로 쓴 isGrounded에서 에러가 발생한 것으로 추정됨. Raycast 방식말고 다른 방법을 찾아야 하나?
- (해결!) => 기존의 RaycastHit2D hit = Physics2D.Raycast(transform.position, Vector2.down, 0.1f, groundmask); 에선 바닥을 인식하지 못했음. => groundmask (layermask)를 빼고 하단의 조건에서 if (hit.collider != null) 여기에 hit.collider.CompareTag("Ground")를 && 로 연결해 추가함. => 땅(Tag : Ground)에 닿아있을 때 걷기 애니메이션 안 나오는 버그 해결

8월 29일

- 애니메이션을 코드로 다시 짜고 있음. -> 애니메이션이 자꾸 서로 겹치는 원인이 뭔가 하고 계속 살펴봄. -> RayCast로 isGrounded를 체크하는 코드에서 Laymaks 지우고 CompareTag로 수정. 이거 자체는 문제가 없음
- 점프했을 때 Jump 애니메이션이 Idle 애니메이션에 오버라이딩 되는 문제 발생.
- 방금 새로 'Idle', 'Walk', 'Jump' 애니메이션 코드 커플링 관계를 고민해보니까

1. Move() 함수에서 isGrounded 일 때 'Walk' 혹은 'Idle' 애니메이션 클립을 재생함.
2. OnCollisionEnter2D() 함수에서 Ground 태그 오브젝트에 충돌할 때 jumpCount = 1, isGrounded = true로 설정해 줌
3. Update에서 점프키를 눌렀을 때 Jump()함수가 실행되게 한 코드에서 isGrounded = false가 없었음.
4. 그렇기 때문에 여전히 Walk와 Idle 애니메이션 클립 실행조건이 충족되서 Jump 애니메이션이 오버라이딩 된거임. -> 수정한 후에는 제대로 점프 애니메이션이 출력됨.

- * 결론 -> 코드로 애니메이션 짤거면 먼저 트리로 그리면서 관계도 다 짠 다음에 코드 짜라. 안 그러면 커플링이 서로 얽혀서 고생한다.

- Const string에서 Const를 붙여주면 해당 변수가 상수화된다. 후에 수정이 불가능하다. 그렇다면 왜 쓰냐? -> 원주율 같이 값이 바뀔 필요가 없는 것 에다가 붙여서 후에 값이 수정되지 않도록 한다. -> 코드의 안정성이 높아진다. # 참고 Const는 배열로 만들 수 없다. readonly로만 만들 수 있다.

- 기존에 메카님에서 블렌드 트리로 일반 공격 모션 1, 2, 3 중에 랜덤으로 하나 뽑는 걸 할 때는 r = random.range(0, 3)로 r을 설정하고 if (r=1) , if(r=2), if(r=3)으로 설정했었다. 하지만 이렇게 하면 코드가 쓸 데 없이 길어지고 효율적이지도 않다.
-

C# PlayerMovement.cs × C# CameraManager.cs ×

Rider detects naming conventions in opened solutions and updates settings accordingly. Click 'Configure' to change.

```
194
                Frequently called  1 usage
195         IEnumerator Attack()
196         {
197             string[] Attacks = new string[3];
198
199             Attacks[0] = Player_Attack01;
200             Attacks[1] = Player_Attack02;
201             Attacks[2] = Player_Attack03;
202             if (!isAttacking)
203             {
204                 isAttacking = true;
205                 if (attackCount == 1)
206                 {
207                     //난수생성
208                     int r = UnityEngine.Random.Range(0, 3);
209
210                     if (isGrounded)
211                     {
212                         ChangeAnimationState(Attacks[r]);
213                     }
214                     else if (!isGrounded)
215                     {
216                         attackCount = 0;
217                         ChangeAnimationState(Attacks[r]);
218                     }
219                 }
220                 Invoke( methodName: "AttackComplete",  time: 0.5f);
221             }
222
223             yield return new WaitForSeconds(0.1f);
224         }
225
226
                Frequently called  1 usage
227         void AttackComplete()
228         {
229             isAttacking = false;
```

- 이를 배열에 넣고 r = random.range(0, 3), 배열[r]로 뽑아 쓰는 방식으로 코드를 수정함.
- (if문을 3번 쓰는 것보다 코드가 훨씬 깔끔해짐.
- (오류발생) 코드로 공격 모션을 구현했으나 여전히 공격 버튼을 눌렀을 때 공격이 나갈 때도 있고 안 나갈 때도 있음 => 유니티의 고질적인 문제? 싱글쓰레드 -> 하나의 쓰레드에 다 넣으려고 하니까 병목이 난 것이 원인인가? -> 공격을 코루틴으로 빼서 멀티 쓰레드 방식으로 작동시켜주면 개선이 될까?
- 공격 모션 자체는 점프 공격의 경우는 공격모션이 제대로 나오는 게 확인 됨. 다만 점프 공격 도중 땅에 닿았을 경우 공격모션이 다 끝나지 않았음에도 도중에 끊기고 원래 상태로 돌아가는 오류가 발생했다.
- 서 있을 때 공격 모션은 Idle, Walk 애니메이션에 오버라이딩 되고 있다. 이 부분 확인 필요함. 코루틴으로 공격 코드를 따로 작성했으나 공격 안 나가는 현상은 해결되지 않았다. 빈도 수가 조금 줄어 들은 것 같긴 하지만 여전히 명확한 해결을 얻진 못한 상태.

- 공격 애니메이션이 Idle, Walk 애니메이션에 오버라이딩 되는 버그 해결함.

  → bool 값 변수 isAttacking = false를 디폴트로 선언.

  → 이동, Idle 상태 애니메이션 조건에 isAttacking false 추가

  → attack 코루틴에 if(!isAttacking) {isAttacking = true}를 줘서 공격 모션 중일 때 다른 Idle, Walk 애니메이션으로 오버라이딩 되지 않도록 설정함.

  → ChangeAnimationState() 종료 후 Invoke(AttackComplete를 불러옴. isAttacking을 다시 false로 변경. (공격 애니메이션이 끝난 후 Idle, Walk로 변경 가능)

- • 해결되지 않은 문제점 -> 공격 버튼을 눌렀을 때 공격이 나갈 때 있고 안 나갈 때 있는 문제가 해결이 안됨 -> 싱글쓰레드 병목 현상이 원인인가 싶어 공격 모션을 코루틴으로 뺐으나 해결 안됨. -> 추가적인 조치 필요.

코드로 애니메이션을 구현하면서 느낀 점

게임의 성능과 깨끗한 코드를 위해 추가적으로 배워야 할게 많다는 것을 깨달았다.

예를 들어 공격버튼을 눌렀을 때 공격이 랜덤하게 나오는 버그는 아직 해결하지 못했지만 여러 가지 원인과 해결책을 탐구함.

첫 번째로는 단순히 코드에서 조건을 잘못 줘서 오류가 난 경우를 가정하였음. 코드를 함수 별로 전부 다 분할하여 커플링마다 오류가 나는 부분을 찾는 식으로 디버깅을 시도함.

싱글쓰레드 방식인 유니티에서 여러 처리를 한꺼번에 하려고 해 쓰로틀링이 걸린 경우를 가정하고 공격 모션을 코루틴으로 제작해 인위적으로 멀티쓰레드 방식으로 처리가 되도록 조정해보거나 드로우콜을 줄이기 위해 Sprite Atlas를 사용함.

컴퓨터 메모리에 대해 이해(스택, 힙)하려 정보를 찾아보고 컴퓨터가 밑바닥에서 어떻게 돌고 있는지 개념적으로라도 알기 위해서 이진법과 비트연산에 대해 공부했음. 이 과정이 평소에 작성하는 코드에 대해 조금이나마 근본적인 이해를 할 수 있었음.

컴퓨터 과학이랑 자료구조, 알고리즘 등에 대한 필요성을 직접 체감하게 되었다는 것이 가장 큼

간단한 예를 들어서 철수 영희 바둑이, 이렇게 3명이 있다고 가정했을 때

if 사용자 철수, print 남자

if 사용자 영희, print 여자

if 사용자 바둑이, print 댕댕이

이런 식으로 쓸 수도 있지만 그냥 Array 써서

{철수, 영희, 바둑이}

{남자, 여자, 댕댕이}

이런 식으로 하거나

대상이 두 명이면

(남자)? 철수: 영희 이런 식으로 삼항연산자로 처리해도 되는 거고

해당하는 항목 값이 없는 경우(null / undefined)에는 옵셔널 체이닝 (?.)으로 처리하면 되는 것.

클라이언트를 뜯을 때 몬스터 리소스에서 있었던 무수히 많은 0과 1도 위와 일맥상통하다고 볼 수 있음. 예를 들어서 몬스터의 데이터를 하나 저장하는 데 있어서도

1) 몬스터인가 플레이어인가? 1, 0
2) 보스몬스터인가? 일반몬스터인가? 1, 0
3) 근접공격인가? 원거리공격인가? 1, 0
4) 선공인가 후공인가? 1, 0

이런 식으로 이진법으로 나타낼 수 있고 만약 몬스터에 수가 너무 많아져서 이진법으로 나타내는 숫자를 전부다 치기 힘들다면 새로운 데이터를 1번 위치에 생성하고 시프트 연산자로(< >) 맨 뒤로 밀어버리면 코드는 조금만 치고 원하는 것은 할 수 있게 되는 것임.

결국 자료의 성향에 따라 그에 어울리는 자료구조의 방법도 다 다른 것이다. 요리로 비유하자면 계란후라이 하더라도 그냥 기름 있으니까 엑스트라버진 올리브유 들이붓고 하는 거랑, 엑스트라버진 올리브유는 타는 점이 낮아서 튀김에는 적합하지 않으니 쓰면 안된다랑 어떤 계란후라이가 더 맛있냐는 불 보듯 뻔한 것.

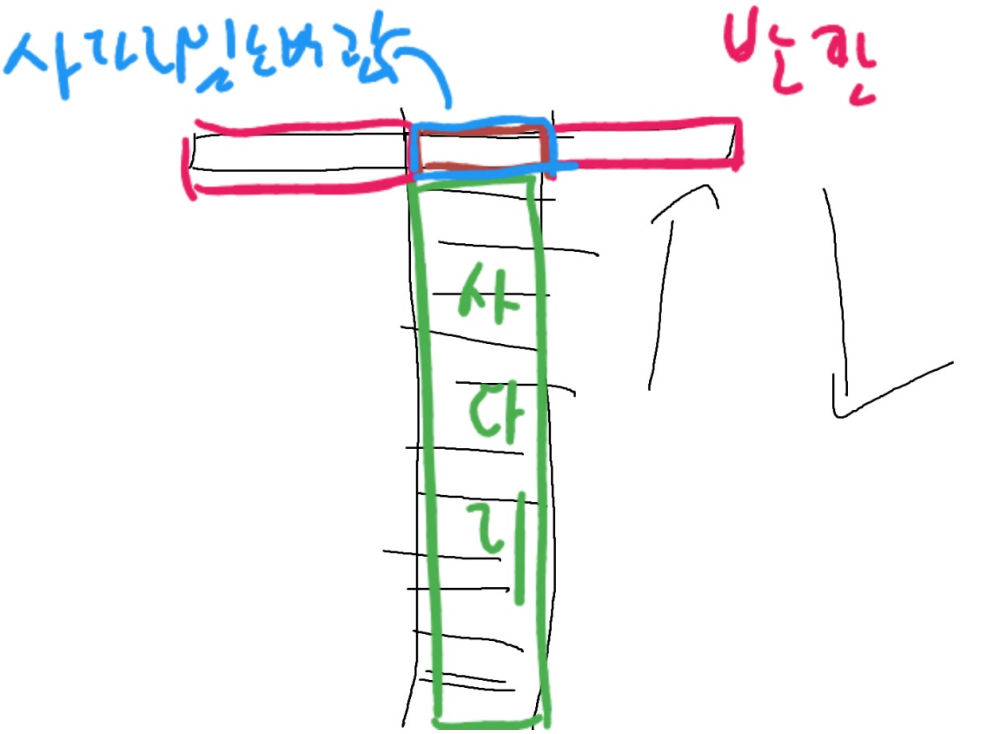
그래서 결론은? CS랑 자료구조 꾸준히 공부하자.

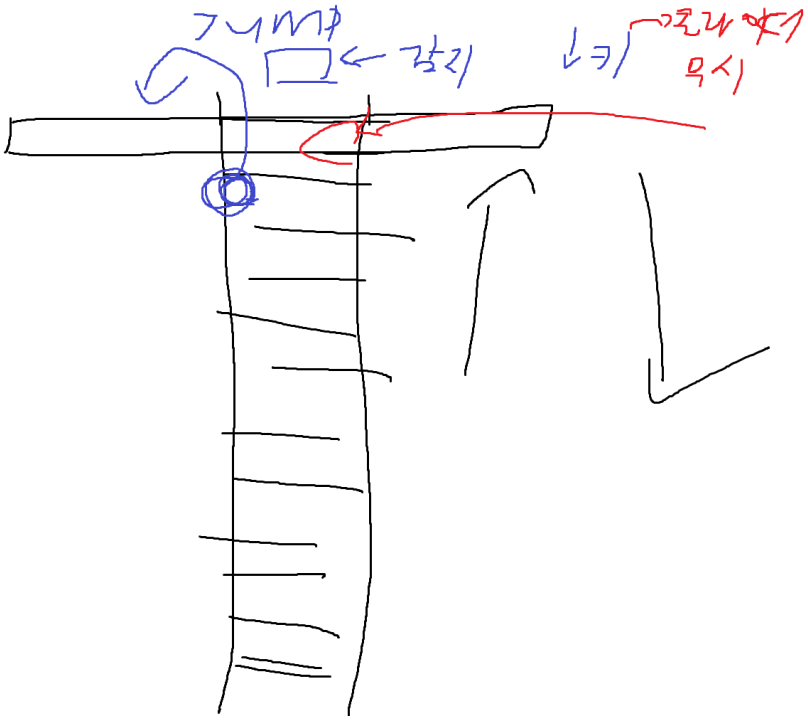
+여기에 더해서 객체지향이랑 디커플링에 대해 좀 더 이해해야 할 것 같다. 지금 모작하면서 짜놓은 코드는 캐릭터 랜더링, 물리, 애니메이션, 사운드 등 게임에 필요한 코드를 PlayerMovement 코드 하나에 다 때려 박아놓은 상태임. 단순히 코드 길이가 긴거면 그나마 나은데 코드 안에 요소들이 죄다 커플링되어 있으니까 오류 떴을 때 하나 수정할 때 연관된 거 다 머리 속으로 생각하고 있어야 하고. 이게 엄청 비효율적이고 변수가 더 많아지면 내가 감당을 못할 거 같음. 기능별로 코드 쪼개고 필요할 때 가져오는 식으로 변경해야 할 것 같음.

8월 30일

- 코드로 사다리를 타고 올라가는 것 구현했음. 바닥을 뚫고 올라가는 것 확인. 다만 바닥에서 사다리를 타고 내려가는 거가 구현이 아직 안됐음. 수정 필요함.

- 사다리 맨 윗부분에 감지용 콜라이더를 추가하고 해당 콜라이더와 충돌하고 아래 방향 키를 눌렀을 때 발판 콜라이더가 없어지도록 설계?





- 발판의 기존 rotationOffset은 0으로 되어 있음. 아래에서 위로 점프했을 때 발판에 막히지 않음. rotationOffset을 180으로 수정하면 아래에서 위로 가는 건 안되고 위에서 아래로 떨어지는 것은 가능해진다.

- 사다리 작동 로직

  → 두 번째 그림처럼 파란색 감지용 오브젝트를 새로 생성, 콜라이더 부여.

  → OntriggerEnter로 플레이어가 해당 콜라이더와 충돌했을 때 bool 변수 = True를 주고 사다리 물리처리를 담당하는 Vertical 스크립트에서 해당 bool 변수가 true이고 하단 방향키를 눌렀을 때 rotationoffset =180으로 줘서 사다리 타고 내려가게 설계. 플레이어의 y위치가 발판의 y위치보다 얼마 만큼 낮을 때 다시 rotational offset = 0f; 주는 방식으로 설계?

- 하단 점프 로직 설계

  → 아래 방향키를 누른 채로 Jump 버튼을 눌렀을 때 rotationOffset이 180으로 변경 되도록 처리. 위에서 아래로 가는 콜라이더가 작동하지 않도록 설계. 이렇게 하니까 점프 후 콜라이더에 걸리지 않고 하단으로 내려감. (다만 하단 점프 모션을 새로 만들 필요가 있을 거 같다.) 플레이어가 하단 점프로 발판을 통과해서 다른 발판(태그가 Ground인)에 충돌했을 때 rotationOffset이 리셋 되도록 설계.

  - 사다리랑 하단 점프 로직은 좀 더 고민할 필요가 있음.

- 맵 오브젝트와 플레이어 간의 상호작용을 구성하면서 느낀 점.

  1. 상호작용을 구현할 때 bool 변수 등 변수에 변화를 줘서 상호작용을 설계할 거면 맵 오브젝트가 아니라 캐릭터에 변화 값을 부여하는 게 맞다.
  2. 특히 싱글플레이 게임이 아닌 멀티플레이 게임 같은 경우에는 꼭 그렇게 설계해야 된다.
  3. 예를 들어서 위의 사다리 작동 로직을 멀티플레이 게임에 적용했다고 가정해보자. 플레이어가 사다리를 내려가는 조건(bool 변수가 false이고 isGrounded일 때)을 충족하여 사다리 내려가는 애니메이션이 발동한다. 이때 bool 변수 값은 true로 바뀌게 되고 해당 상태에서 다른 플레이어가 사다리를 내려가려고 한다면 조건이 충족되지 않아 내려갈 수가 없다. 그렇기 때문에 맵과 플레이어의 상호작용을 구현할 때는 플레이어를 중심으로 코드를 짜는 것이 좋다.

2022년 9월 2일 현재 진행 상황 및 차후 계획

현재 리소스 수집, 맵과 캐릭터 제작, 캐릭터 움직임 구현, 오브젝트와의 상호작용까지 진행되었고 캐릭터 애니메이션과 오브젝트 상호작용은 추가적으로 작업 진행중임.

지금까지 작업한 내용 중에서 가장 시간이 많이 걸리는 부분은 애니메이션 제작이었음.

기존에 메카님을 활용한 애니메이션 제작에서 코드로 관리하는 것으로 변경한 것은 잘한 선택이지만 코드로 관리하기 위해서는 변수 간에 충돌이 일어나지 않도록 주의해서 설계해야 함. → 현재 시간이 많이 걸리고 있는 원인

결합도(Coupling)은 낮추고 응집도(Cohension)은 높게 코드를 작성해야 하는데 이를 위해서는 코드를 적절하게 분할해서 사용할 줄 알아야 함. 코루틴으로 멀티쓰레드 방식으로 처리하거나 싱글톤 혹은 옵저버로 필요한 기능만 상속 받아 꺼내 쓸 수 있도록 코드가 작성하는 등 추가적으로 배워야 할 것들이 많음.

탑다운 방식으로 방식으로 작업하고 있으나 하나 작업하는 데 시간이 오래 걸려서 기초를 조금 더 공부하고 가야겠다고 판단함. 유튜브 유니티 관련 채널인 골드메탈 강의에서 '유니티 3D 쿼터뷰 액션게임 제작'을 따라해 보고 마이크로소프트 C# 가이드 등을 공부하면서 기본 역량을 좀 더 늘리고 나서 곧 바로 모작을 이어 나갈 예정.

메이플스토리 모작 개발이 끝나면 3D 게임 제작 포트폴리오를 작성하거나 게임잼에 참가하며 개발 역량을 키워나갈 예정임.