3D콘텐츠 이론 및 활용

12주(2). 캐릭터 모션처리(애니메이터 컨트롤러)

- 캐릭터 이동
- 캐릭터 상태 머신

학습개요

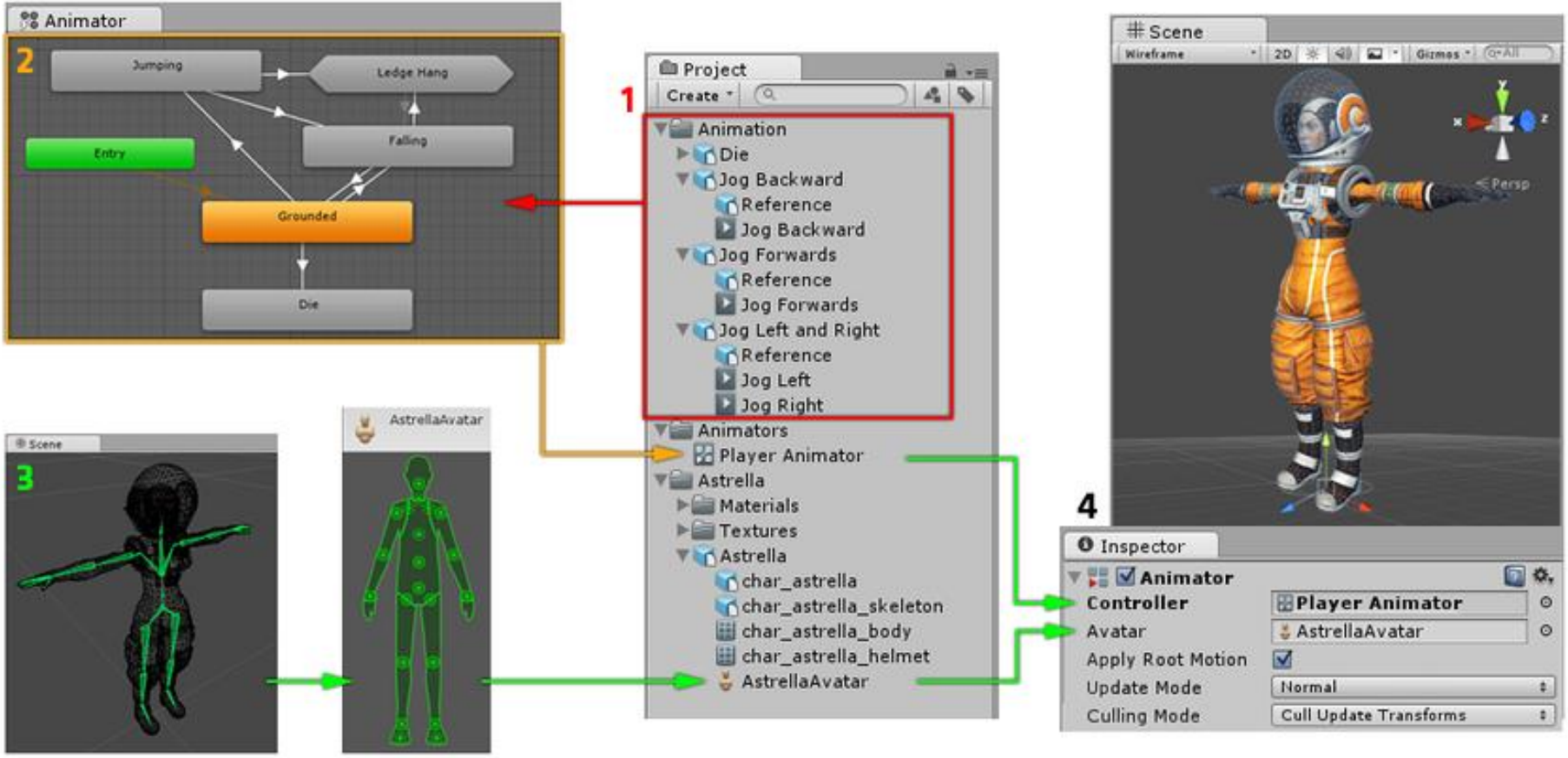
학습목표

- Character Controller 컴포넌트를 이해하고 적용할 수 있다.
- 플레이어를 이동하기 위한 스크립트를 작성할 수 있다.
- Animator 컴포넌트 설정할 수 있다.
- Animator 뷰를 이용하여 애니메이션을 설정할 수 있다.

학습내용

- Character Controller 컴포넌트
- 플레이어 이동 코딩
- Animator 컴포넌트
- Animator 뷰

<애니메이션 시스템>



1.블렌드 트리, 2.상태머신(status machine),
3. 아바타(메카님), 4. 애니메이션 컨트롤러

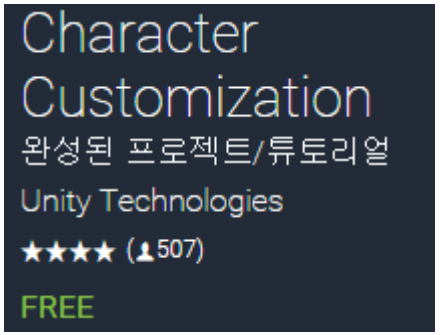
1. 캐릭터 모델

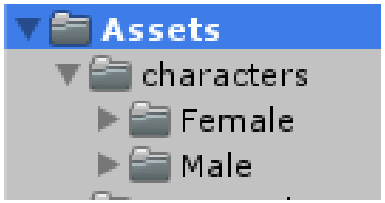
1) 모델링

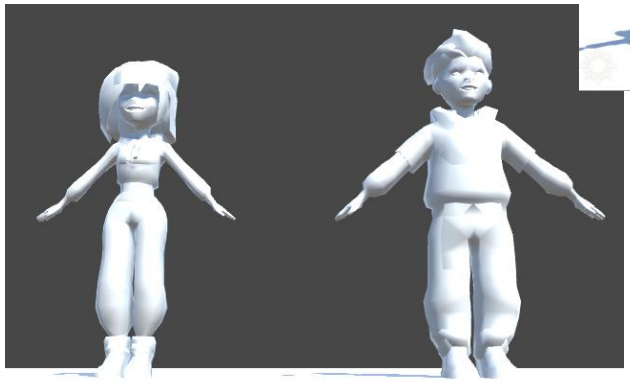
- Unity는 .FBX, .dae (Collada), .3DS, .dxf 및 .obj, FBX 익스포터 파일을 로드 할 수 있음
- 모델 파일에는 애니메이션 데이터가 포함되기도 함.
- 유니티 지원 프로그램
  - Maya, Cinema 4D, 3ds Max, Cheetah3D, Modo, Lightwave, Blender

2) 매트리얼 입히기

- Character Customization 다운로드
- Character 폴드만 남기고 삭제







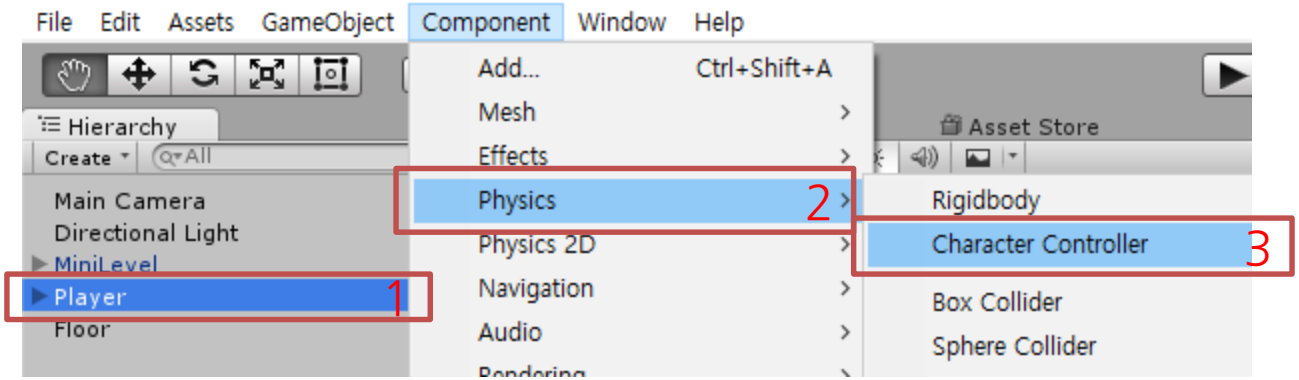
2. 플레이어 이동 처리

1) 플레이어 이동

- 포지션 이동
- 충돌 후 포지션이동
- 장애물 벽과의 충돌
- 기울어진 발판에서 움직임 처리
- 가상월드에서 충돌처리는 Character Controller 컴포넌트를 이용

2) Character Controller 컴포넌트 사용하기

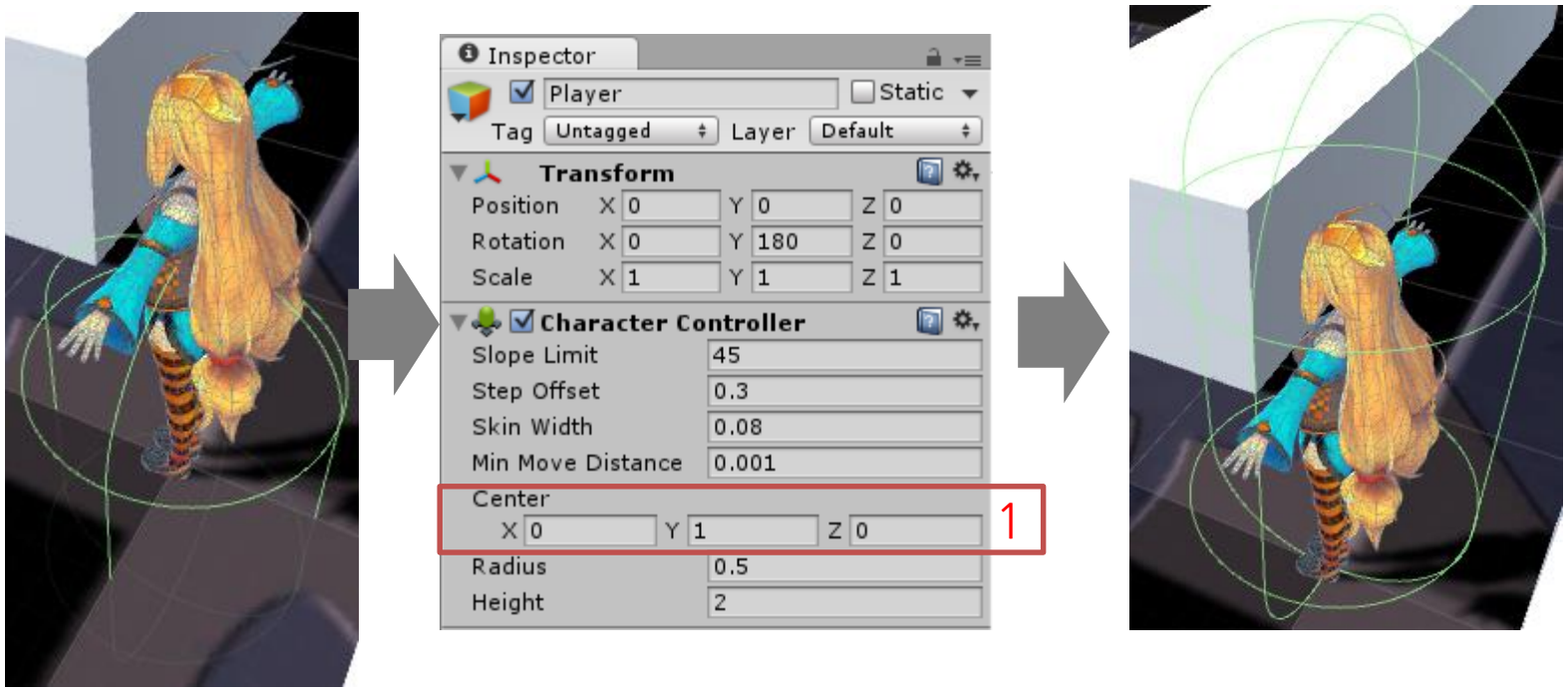
- Character Controller 컴포넌트 추가하기
- 컴포넌트가 추가되면 캐릭터에 캡슐형태의 녹색선이 표시됨



2. 플레이어 이동 처리

2) Character Controller 컴포넌트 사용하기

- Character Controller 위치 조절
  - 센터의 높이 값을 0.7 ~ 1로 주어 위치를 조절한다.

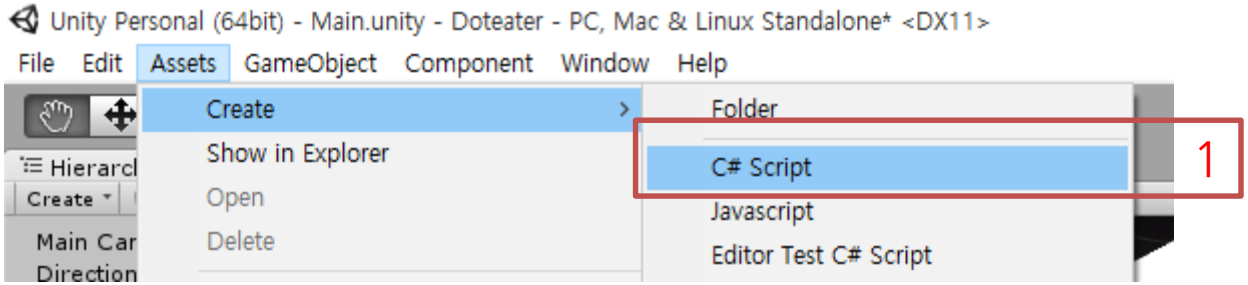


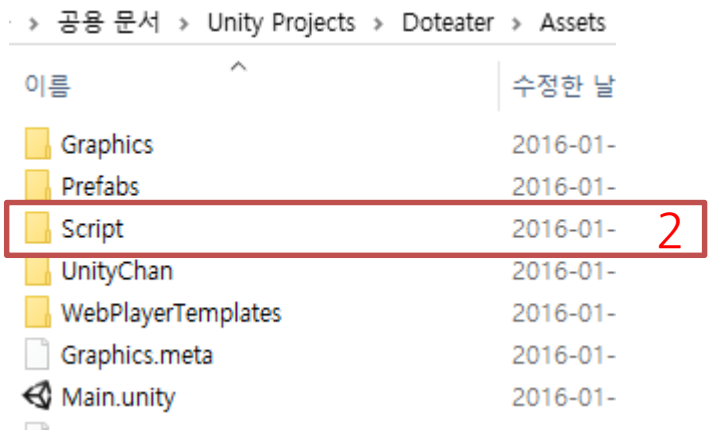
2. 플레이어 이동 처리

3) 플레이어 조작하여 이동하기

- 스크립트 만들기

- 메뉴 또는 프로젝트 뷰에서 생성가능
- 생성 후 파일이름은 Player로 변경
- 생성된 파일은 Assets 폴드에 Script폴드를 만들어 관리하는 것이 효율적

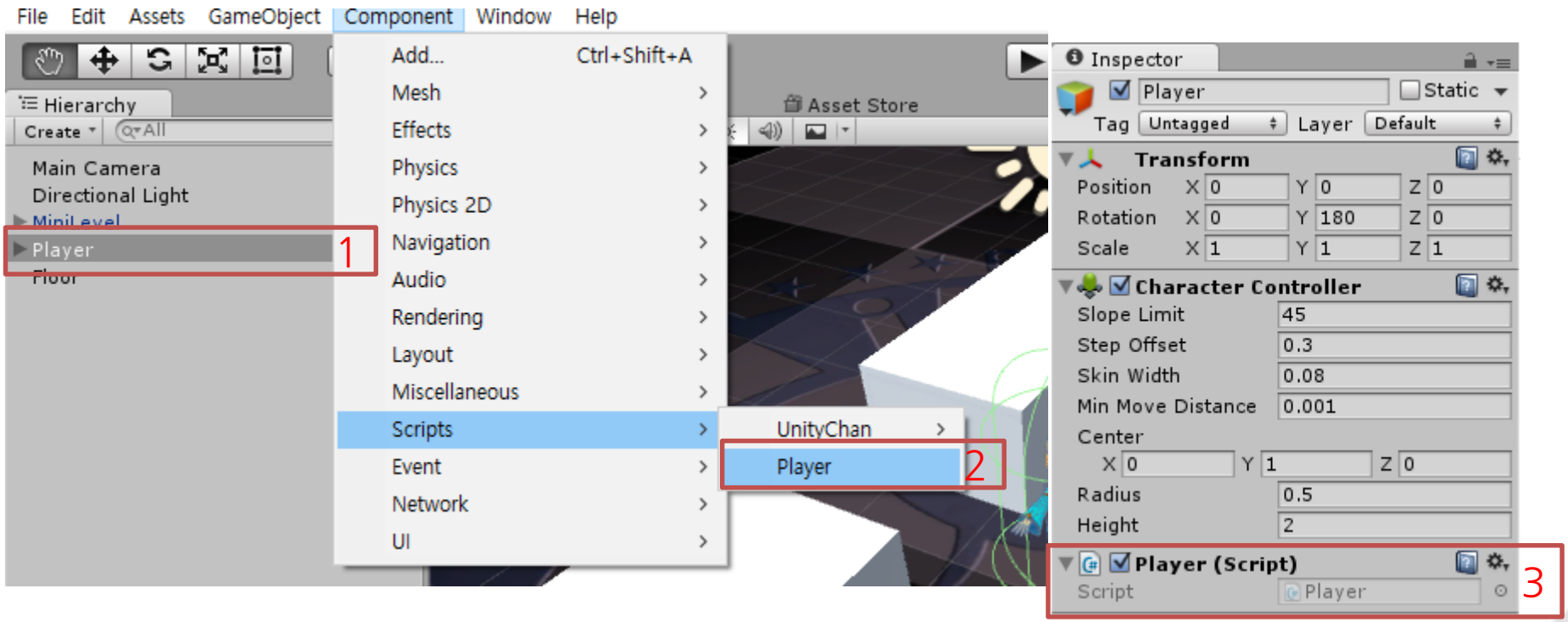




2. 플레이어 이동 처리

3) 플레이어 조작하여 이동하기

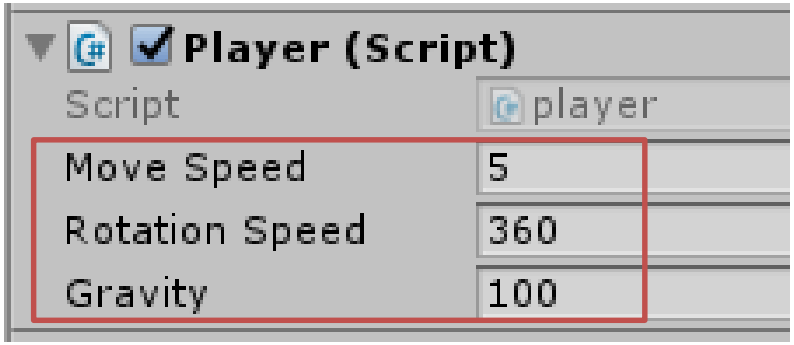
- 플레이어 게임오브젝트에 스크립트 적용하기



2. 플레이어 이동 처리

3) 플레이어 조작하여 이동하기

- player.cs 스크립터 작성 (1/2)
  - 캐릭터를 이동하려면 이동속도, 회전속도, 중력(Y축)으로 이루어진다.




```
public class player : MonoBehaviour
{
    public float moveSpeed = 5f;
    public float rotationSpeed = 360f;
    public int gravity = 100;

    CharacterController characterController;

    void Start ()
    {
        characterController = GetComponent<CharacterController> ();
        animator = GetComponent<Animator> ();
    }
}
```

2. 플레이어 이동 처리

3) 플레이어 조작하여 이동하기

- Player.cs 스크립터 작성 (2/2)

```
void Update ()
{
Vector3 direction = new Vector3 (Input.GetAxis ("Horizonta
l"), 0, Input.GetAxis ("Vertical")); // 입력 값 얻기

if (direction.sqrMagnitude > 0.01f) { // 자연스러운 방향전환
   Vector3 forward = Vector3.Slerp (transform.forward, dir
ection, rotationSpeed * Time.deltaTime / Vector3.Angle (tr
ansform.forward, direction));
   transform.LookAt (transform.position + forward);
}
        direction.y -= gravity * Time.deltaTime;
        characterController.Move (direction * moveSpeed *
Time.deltaTime); // 이동처리
}
```

▪ 주요 메서드

`Input.GetAxis ("Horizontal"), Input.GetAxis ("Vertical")`
: X, Z의 이동량, 키 입력에 따라 달라진다. 각각 -1.0 에서 1.0까지 값을 가짐
(좌우, 아래위)=왼쪽아래, 오른 쪽 위

`direction.sqrMagnitude`
: 방향과이동량을 비교하여 이동중인 경우, 현재 플레이어의 방향과 입력된 키 입력 방향을 사용하여 새로운 방향을 계산하여 반영

transform.LookAt(transform.position + forward)
: 게임오브젝트의 방향을 부드럽게 변경.(트랜스폼 속성의 위치 값에 수정된 forward 값을 반영)

characterController.Move(direction * moveSpeed * Time.deltaTime)
: Move함수를 이용하여 이동처리, position속성을 변경하지 않고 위치만 변경하면 끊김이 생기고 이로 인해 충돌처리가 어렵고 속도 값도 얻을 수 없음

Vector3.Slerp
: 그래픽스 용어, 구형 보간, 두 방향 사이에 곡선을 그리며 보간하는 데 사용

생각해보기

구현한 플레이어의 이동이 부자연스럽게 처리되는 이유와 해결방법을 생각해 보자

Animator 컴포넌트는 씬에서 게임 오브젝트에 애니메이션을 추가하는데 사용
Animator 컴포넌트에는 어떤 애니메이션 클립을 사용할 것인가, 그리고 그들을 언제 어떻게 블렌드하여 애니메이션 할 것인가를 명확히 정의하기 위해 Animator Controller를 사용함.

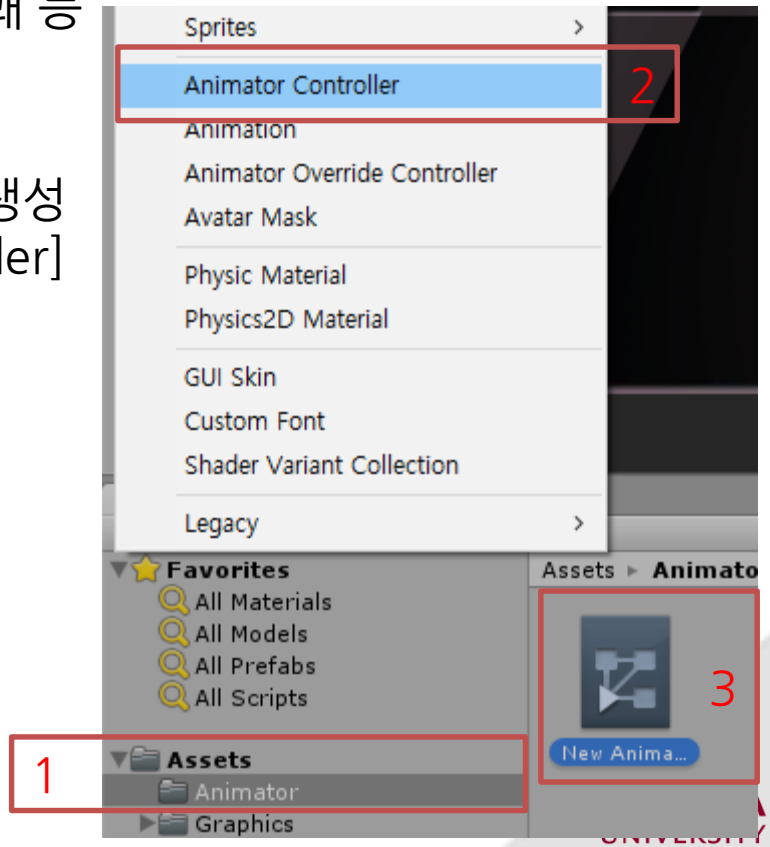
3. 플레이어 모션 설정

1) Animator 컴포넌트 설정하기

캐릭터의 (동작)상태를 전환하기 위해서는 Animator Controller 파일로 관리한다.

- 캐릭터의 상태
  - 대기, 걷기, 뛰기, 휴식, 뒤로 걷기, 놀램, 불쾌 등

- Animator Controller 만들기
  - 프로젝트 Assets 폴드에 Animator 폴드를 생성
  - 프로젝트 뷰 - Create - [Animator Controller]
  - 생성된 파일을 Player로 수정

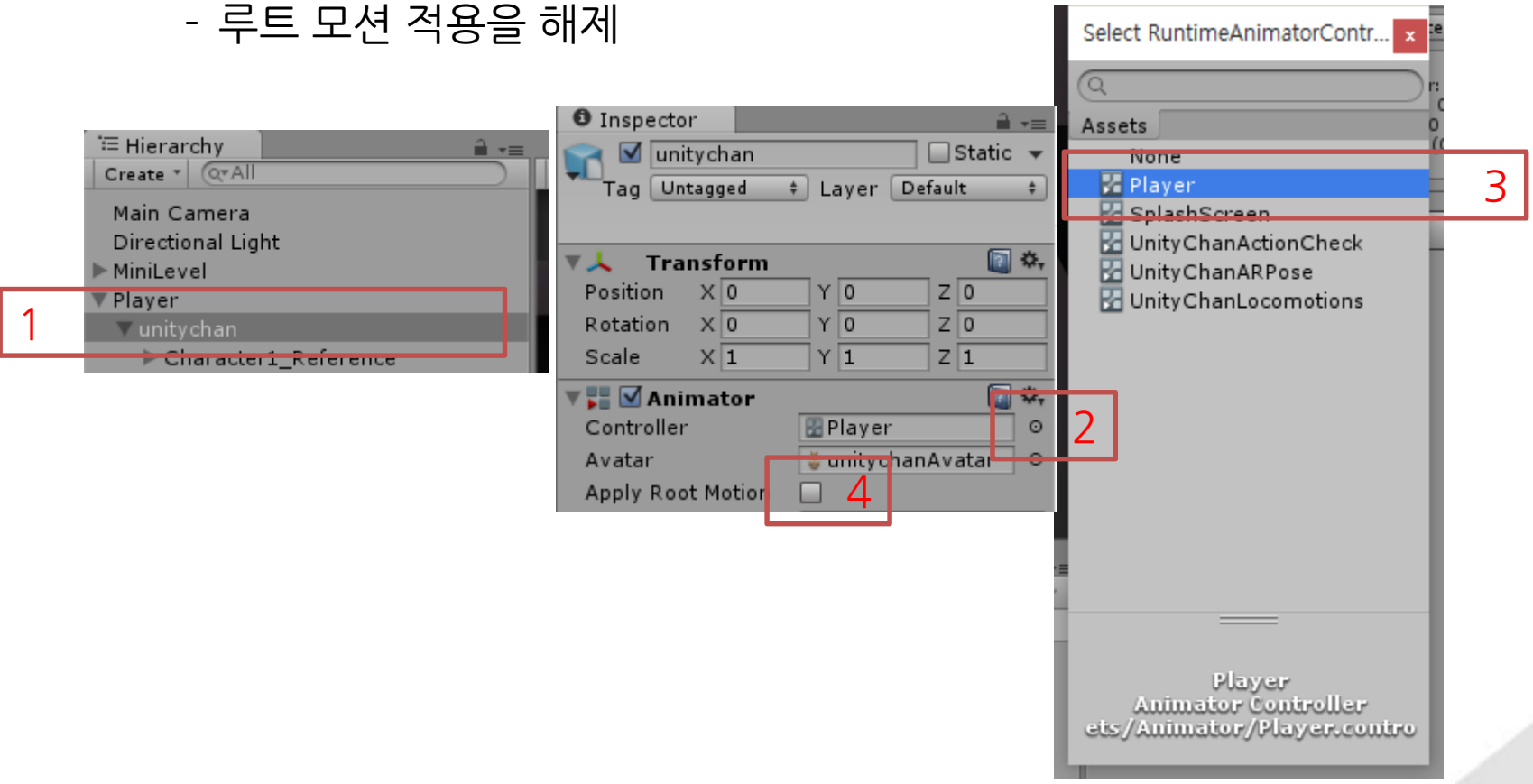


3. 플레이어 모션 설정

1) Animator 컴포넌트 설정하기

- Animator 컴포넌트 설정하기

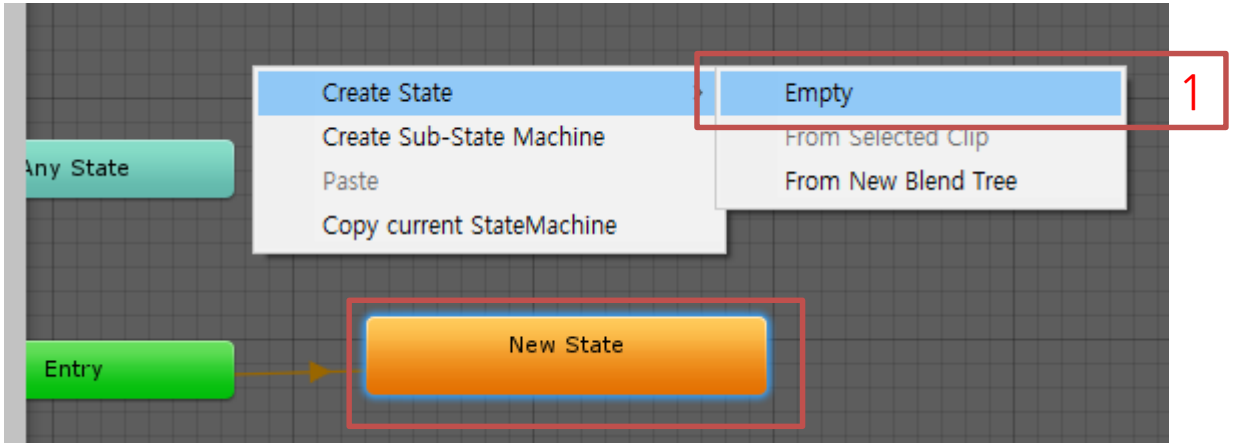
  - 자식 유니티짱 오브젝트를 선택
  - 방금 생성한 컨트롤러로 지정
  - 루트 모션 적용을 해제



3. 플레이어 이동 처리

2) Animator 뷰를 이용한 애니메이션 설정

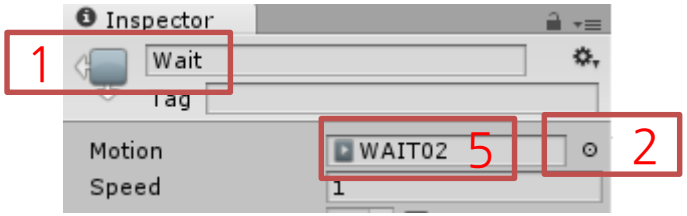
- 애니메이트 뷰 설정
- 빈 스테이트 만들기(우 클릭 팝업창)

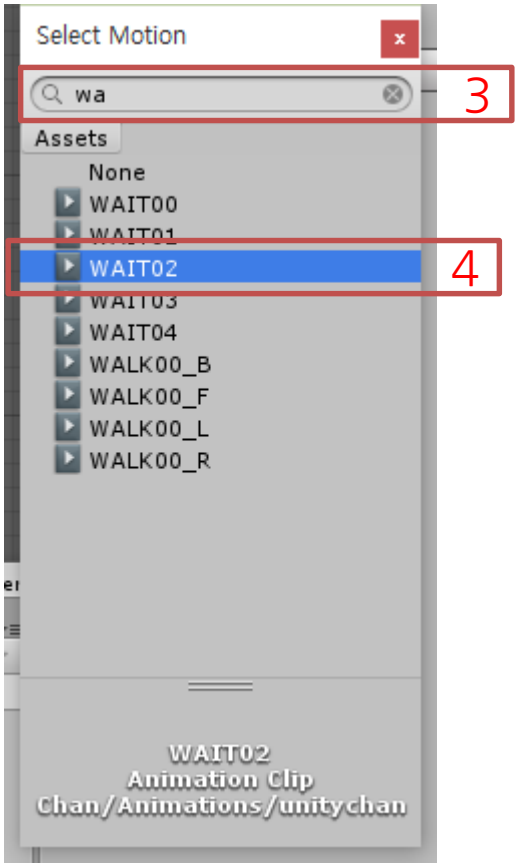


3. 플레이어 이동 처리

2) Animator 뷰를 이용한 애니메이션 설정

- Wait 상태 만들기 예
  - 생성된 New State를 클릭하여 Inspector에서 이름을 Wait으로 준다.
  - 모션을 찾아 등록 후 확인
  - Run 상태도 추가해 보자

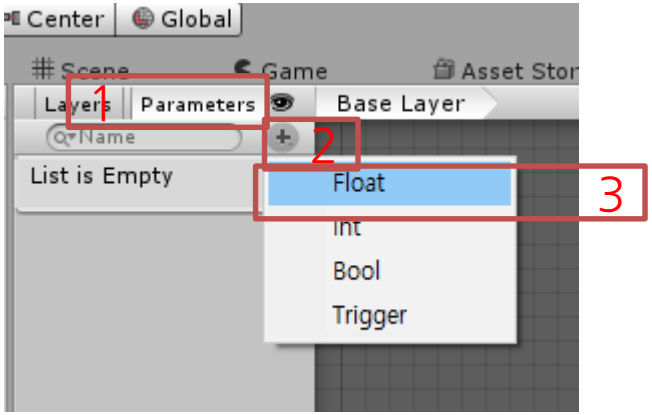


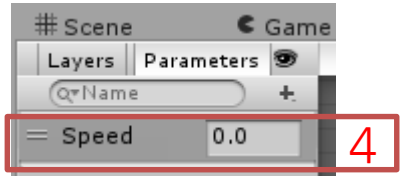


3. 플레이어 이동 처리

2) Animator 뷰를 이용한 애니메이션 설정

- 상태전환을 위한 기준 설정
  - <조건> 이동 속도 조절을 위해 이동속도가 0이면 대기상태 그 이상이면 달리기 상태로 설정
  - 파라메터를 Float으로 지정하고 new Float이 생성되면 이름을 Speed로 변경하고 0으로 초기화





3. 플레이어 이동 처리

2) Animator 뷰를 이용한 애니메이션 설정

- 스크립트 코딩

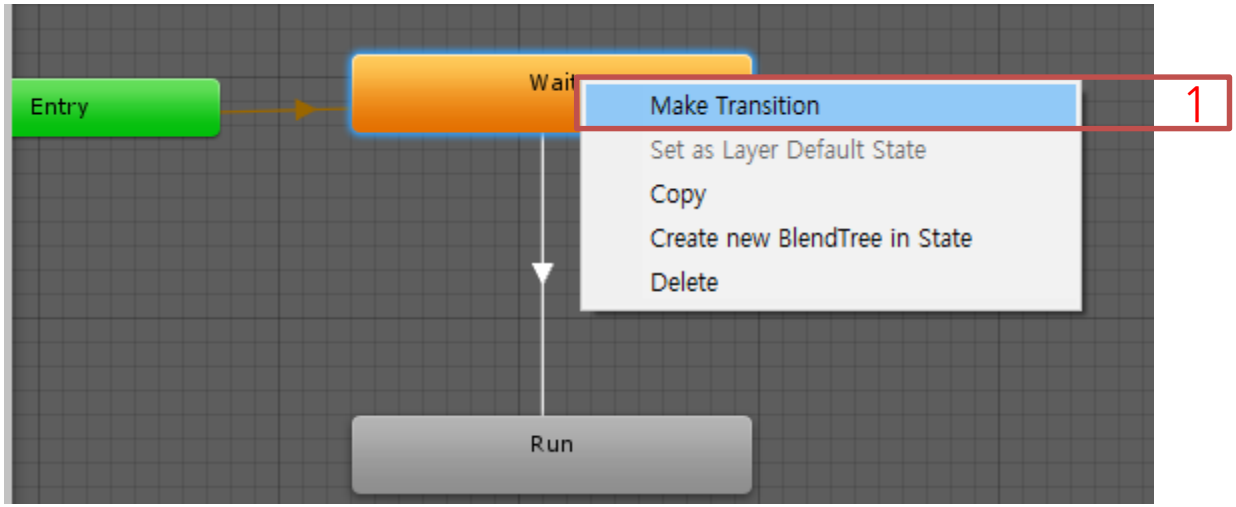
  - 플레이어 이동속도를 Animator Controller의 파라미터로 전달하도로 처리

```
Animator animator;
…

void Start () {
        characterController = GetComponent<CharacterController>();
        animator = GetComponentInChildren<Animator>();
}

void Update () {
. . .
characterController.Move(direction * moveSpeed * Time.deltaTime);
animator.SetFloat("Speed", characterController.velocity.magnitude);
}
```

3. 플레이어 이동 처리

2) Animator 뷰를 이용한 애니메이션 설정

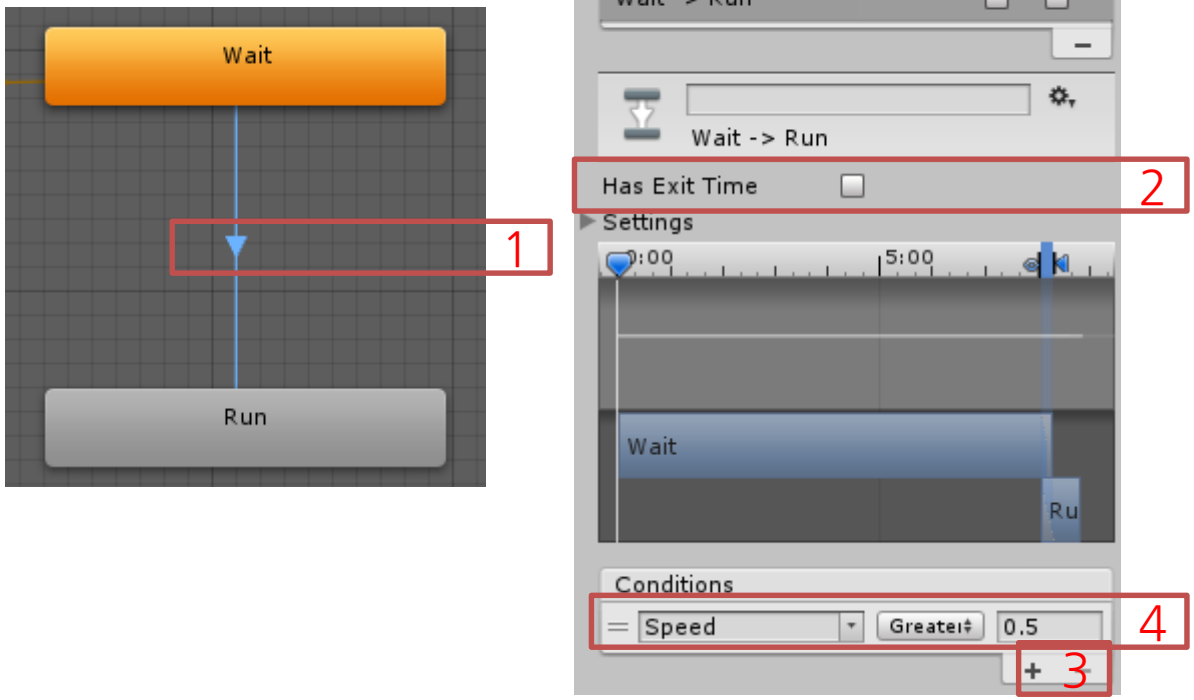
- 상태 관계 설정
  - Wait 상태를 우 클릭 후 Make Transition 으로 run상태로 변환



3. 플레이어 이동 처리

2) Animator 뷰를 이용한 애니메이션 설정

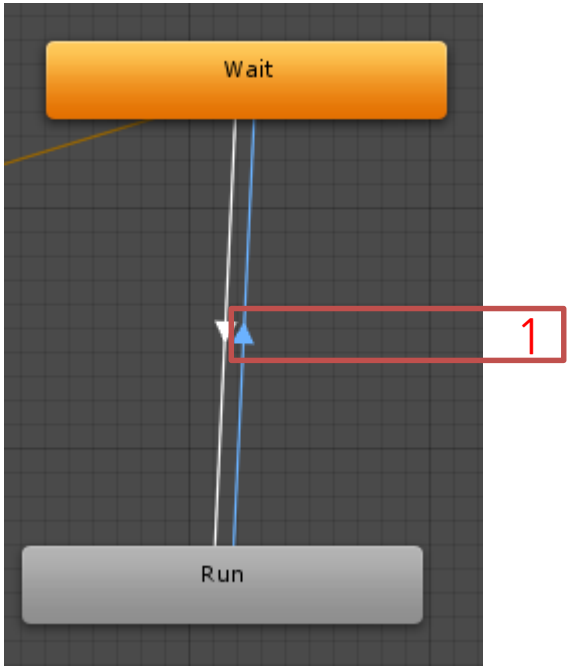
- 상태 전환 조건 설정 (wait → run)
  - Wait 과 run상태 연결선을 클릭하여 조건 설정
  - 스피드가 0.5보다 크면 전환상태로 설정

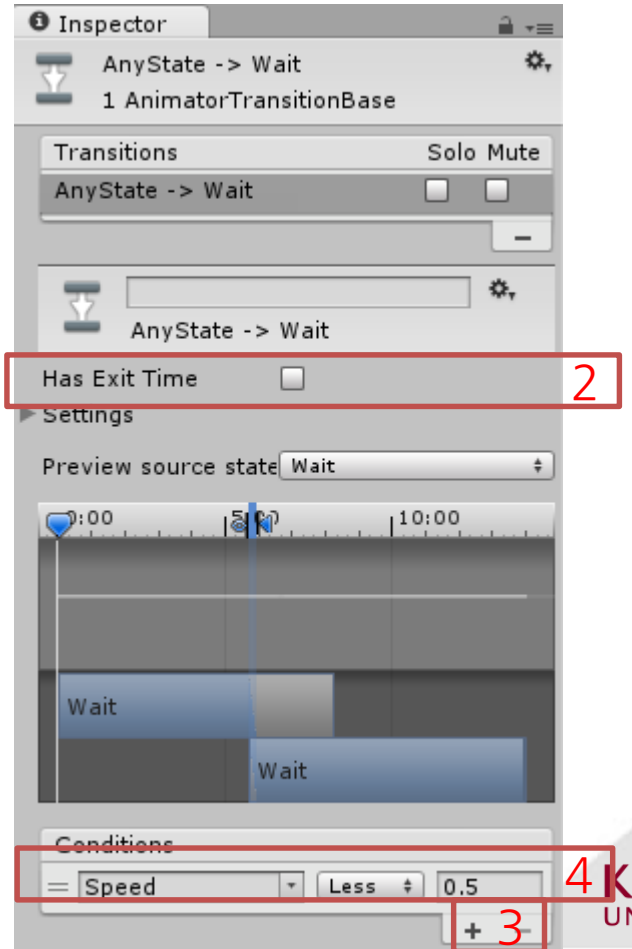


3. 플레이어 이동 처리

3) Animator 뷰를 이용한 애니메이션 설정

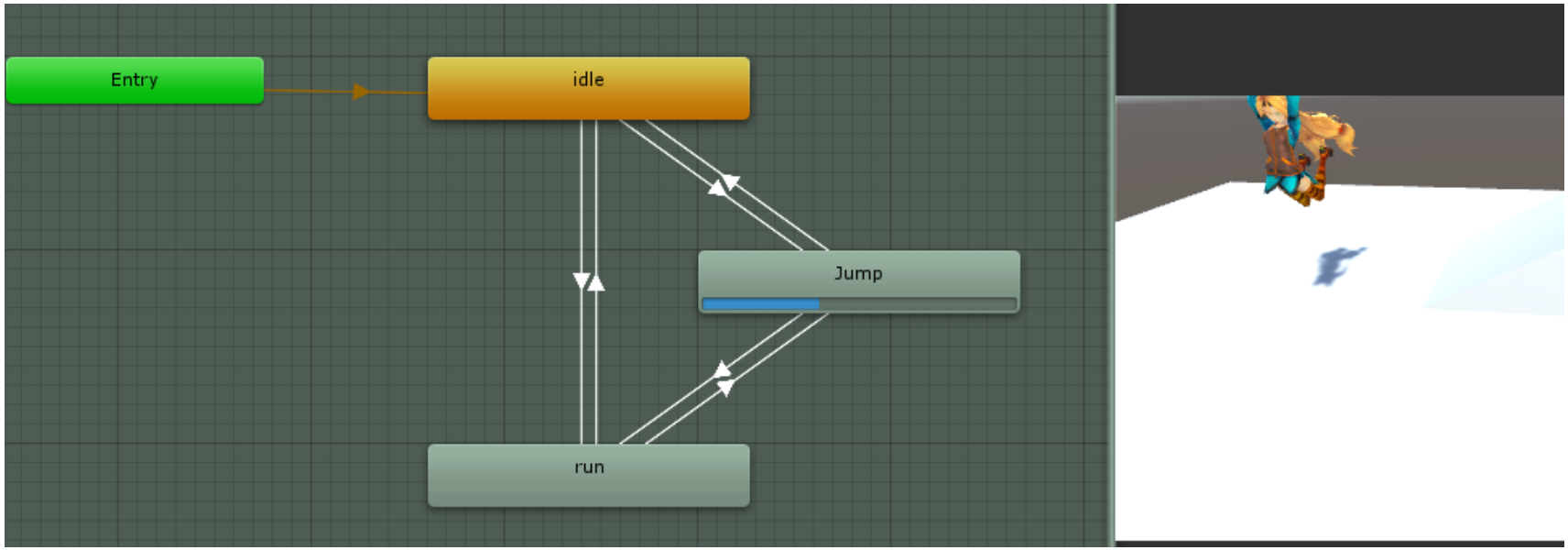
- 상태 전환 조건 설정 (run → wait)
  - Run상태를 우클릭하고 Make Transition, 연결선을 wait으로 연결해 줌
  - 연결선을 클릭하여 조건 설정
  - 스피드가 0.5보다 작으면 전환상태로 설정





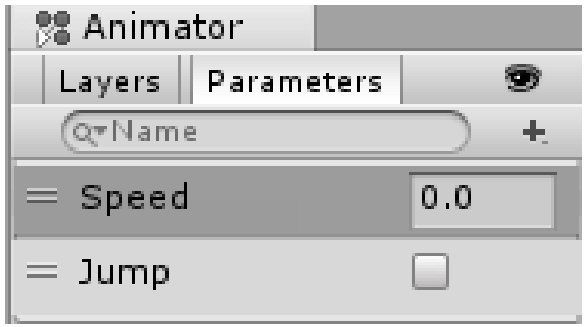
실습

1 스페이스 키를 누르면 점프 애니메이션이 동작하도록 구현하기



실습

```
bool JumpKeyFlag = false;

    if (Input.GetKey (KeyCode.Space)) {
        JumpKeyFlag = true;
    } else {
        JumpKeyFlag = false;
    }

    animator.SetBool ("Jump", JumpKeyFlag);
```



학습정리

1. 물리적인 운동이 없는 캐릭터 오브젝트에는 Character Controller 컴포넌트를 적용한다.

2. 캐릭터의 동작 상태를 전환하기 위해서는 Animator Controller 파일로 관리한다.

3. 자연스러운 애니메이션을 설정하기 위해 Animator 뷰를 설정한다.

참고자료

- 유니티 5 3D 게임 제작 입문, 저자 마쓰다스, Bata, Maruchu, 우니타지야무오, 모리사토린, 역자 김범준,길벗, 2015.11