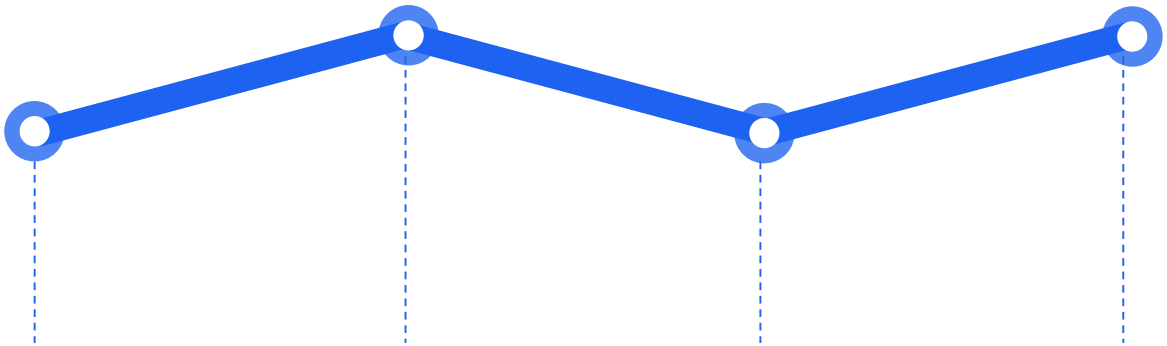
컴퓨터그래픽스

기말고사 대체 과제물 보고서

2017250039 임형택

목차

CONTENTS



01 서론

게임 제작 배경 및 방법 설명

- **방법 1 – 교과서 기반 동작 응용 애니메이션 제작**

  강의 시간에 배운 것을 응용하여 제작하였습니다.
  모델 및 애니메이션은 MIXAMO(www.mixamo.com)에서 가져왔으며
  스크립트는 카메라 부분은 블로그 (wergia.tistory.com/230)에서
  다른 부분은 Unity 공식 문서(docs.unity3d.com)에서 참고하였습니다.

- **애니메이션을 위한 게임 제작**

  애니메이션을 더욱 다양하게 사용하기 위해서 간단한 게임을 만들었습니다.
  게임은 메이플스토리의 끈기의 숲을 바탕으로 하였으며, 배경 음악 및 이미지
  파일 또한 메이플스토리의 이미지입니다. (착지 소리 및 바닥 이미지 제외)

02 제작

캐릭터 컨트롤러 스크립트 - 캐릭터 애니메이션 스크립트 - 맵 제작 - 배경 텍스처 및 사운드 삽입 순서로 제작했습니다.

2.1 | 캐릭터 컨트롤러 스크립트(TPSCAM.cs)

처음에는 움직임, 점프를 구현하고, 애니메이션을 대입해서 진행하다가 메인카메라랑 애니메이션을 연결하는 부분에서 어려움을 겪어 스크립트를 7개 정도 작성하다가 마지막에 하나로 통합한 스크립트입니다. 조금씩 살펴보겠습니다.

```
[SerializeField]
private Transform characterBody;
[SerializeField]
private Transform cameraArm;
```

기본적으로 처음에 변수 및 벡터 선언, 컴포넌트 임포트를 하는 부분입니다.
우선 characterBody와 cameraArm을 [SerializeField] 선언합니다.
public처럼 인스펙터 창에서 접근이 가능하지만 외부 스크립트에서 수정이 불가능하게 만들 수 있습니다. (private 변수지만 inspector 접근 가능)
characterBody는 애니메이션 부분(모델)이고, cameraArm은 메인카메라와 연결하여 회전을 가능하게 만든 빈 객체입니다.

```
Animator animator;
private Rigidbody rigid;
```

그리고 인스펙터 창의 컴포넌트인 animator(애니메이션)와 rigid(움직임)을 사용하기 위해 선언해주었습니다.

```
public static bool IsJumping;
public static bool IsCrouch;
public static bool IsRun;
public static bool IsFalling;
public static bool IsPause;
```

IsJumping, IsCrouch, IsRun, IsFalling, IsPause는 이름에서 알 수 있듯 상태들을 bool 형식으로 행동을 진행 중인지를 체크합니다.
애니메이션 스크립트에서도 사용하기 위해 static으로도 선언했습니다.
각각 점프, 앉기, 달리기, 추락, 정지(추락 딜레이)입니다.

02 제작

캐릭터 컨트롤러 스크립트(TPSCAM.cs)

```
public float speed = 3f;
public float JumpPower = 4f;
public float time;
public float time2;

public AudioSource mySfx;
public AudioClip jumpSfx;

Vector3 Savepoint;

private void Start()
{
    animator = characterBody.GetComponent<Animator>();
    rigid = GetComponent<Rigidbody>();
    IsJumping = false;
    IsCrouch = false;
    IsRun = false;
    IsFalling = false;
    IsPause = false;
}
```

speed, JumpPower, time, time2를 float 형으로 선언합니다.
속도, 점프 높이, 추락 시간과 추락 후 정지를 위해 time을 선언했습니다.

AudioSource, AudioClip은 캐릭터가 바닥에 닿았을 때, 소리를 내기 위해 인스펙트 창 컴포넌트를 가져왔으며, 벡터의 경우는 게임이 어려워서 위치 저장을 위해 선언했습니다.

게임 시작 시 적용되는 함수 Start()에선 animator와 rigid의 컴포넌트를 받아주고, 가만히 서있을 경우 그 무엇의 상태도 아니기 때문에 모든 상태를 false 처리했습니다.

02 제작

캐릭터 컨트롤러 스크립트(TPSCAM.cs)

```
private void Update()
{
    LookAround();
    Move();
    if (!IsPause)
    {
        Jump();
        RunCrouching();
    }
    else
    {
        Pause();
    }

    Save();
    CheckFalling();
    Cursor.visible = false;
}
```

대략 1초에 60번 체크하는 함수 Update() 에서는 LookAround(), Move(), Jump(), RunCrouching(), Pause(), Save(), CheckFalling() 함수들을 제작하여 선언했습니다.

각각 카메라 기능, 움직임, 점프, 달리기와 앉기, 추락 정지, 위치 저장, 추락 감지를 담당하는 함수입니다.

특히 if 문 (!IsPause)의 경우 추락했을 때, 움직임을 제한하기 위해 저렇게 작성하였고, Move() 함수문에서 볼 것이지만, 그 안에서도 같은 if문이 존재합니다. (여기서 바로 Move()를 if문에 넣으면 LookAround()와 맞물려 오류가 발생합니다. 그래서 transform.position 부분에만 넣었습니다.)

Cursor.visible = false; 의 경우 마우스 커서를 가리는 코드입니다.

02 제작

캐릭터 컨트롤러 스크립트(TPSCAM.cs)

```
private void LookAround()
{
    Vector2 mouseDelta = new Vector2(Input.GetAxis("Mouse X"), Input.GetAxis("Mouse Y"));
    Vector3 camAngle = cameraArm.rotation.eulerAngles;

    float x = camAngle.x - mouseDelta.y;
    if (x < 180f)
    {
        x = Mathf.Clamp(x, -1f, 30f);
    }
    else
    {
        x = Mathf.Clamp(x, 300f, 361f);
    }
    cameraArm.rotation = Quaternion.Euler(x, camAngle.y + mouseDelta.x, camAngle.z);
}
```

간단하게 설명하자면 mouseDelta에 마우스 수평 움직임과 수직 움직임을 Vector2의 형식으로 저장하고, camAngle의 경우 cameraArm(메인카메라가 들어있는 빈 객체)를 회전시킬 수 있게 만듭니다.

cameraArm은 360도 x, y, z가 회전이 가능한데, 위나 아래로 너무 회전할 경우, 어지럽기도 하고 불편하기도 하니 값에 제한을 주기 위해서 x에 저장 후, 아래에서 60 ° , 위에서 30 ° 각도를 제한 시켰습니다.

그리고 그 부분을 cameraArm.rotatio에 Vector3의 형식으로 넣어 카메라 회전을 완성 시켰습니다.

(이 부분은 https://wergia.tistory.com/230를 참고하였습니다.)

02 제작

캐릭터 컨트롤러 스크립트(TPSCAM.cs)

◆ 카메라 이동 및 각도 제한

02 제작

캐릭터 컨트롤러 스크립트(TPSCAM.cs)

```
private void Move()
{
    Vector2 moveInput = new Vector2(Input.GetAxis("Horizontal"), Input.GetAxis("Vertical"));

    Vector3 lookForward = new Vector3(cameraArm.forward.x, 0f, cameraArm.forward.z).normalized;
    Vector3 lookRight = new Vector3(cameraArm.right.x, 0f, cameraArm.right.z).normalized;
    Vector3 moveDir = lookForward * moveInput.y + lookRight * moveInput.x;

    characterBody.forward = lookForward;

    if (!isPause)
    {
        transform.position += moveDir * Time.deltaTime * speed;
    }
}
```

함수 Move의 경우, 캐릭터의 움직임을 담당하는데, Vector2의 형식으로 수평, 수직을 받아 moveInput에 넣습니다.
moveInput의 길이가 0이면 이동 입력 X, 0이 아니면 이동 입력이 발생하는 형식입니다.
그리고 카메라 회전에 따라 캐릭터도 회전을 해야하기 때문에 cameraArm.forward의 x, z 좌표를 받아오고,
cameraArm.right의 x, z 좌표 또한 받아옵니다.(y를 받으면 모델이 y축도 변하기 때문에 얼굴만 그럴 생각이 있지 않는 이상은 높이는 배제했습니다.) lookForward와 lookRight에 moveInput을 곱하여 바라보고 있는 방향이 이동 방향이 됩니다.
그리고 애니메이션 모델의 forward가 lookforward면 모델 또한 바라보고 있는 방향이 카메라가 보는 방향이 됩니다.

마지막으로 transform.position을 넣어 이동을 제작했습니다.(※ IsPause 작동 시 움직임 정지)

02 제작

캐릭터 컨트롤러 스크립트(TPSCAM.cs)

◆ 캐릭터 이동

02 제작

캐릭터 컨트롤러 스크립트(TPSCAM.cs)

```
private void Jump()
{
    if (Input.GetKeyDown(KeyCode.Space))
    {
        if (!IsJumping)
        {
            IsJumping = true;
            rigid.AddForce(Vector3.up * JumpPower, ForceMode.Impulse);
        }
        else
        {
            return;
        }
    }
private void Pause()
{
    time2 += Time.deltaTime;

    if (time2 >= 0.6)
    {
        IsPause = false;
        time2 = 0;
    }
```

함수 Jump는 Space가 입력되면 IsJumping 상태를 true로 바꾸고, rigid.AddForce를 이용하여 위로 점프를 합니다.

ForceMode.Impulse의 경우 rigidbody의 질량을 이용하여 짧은 순간의 힘을 가하는 코드입니다.

만약 IsJumping == true인 상태라면 더 이상 점프를 할 수 없게 그냥 return 시켰습니다.

IsJumping을 false로 되돌리는 함수는 나중에 나옵니다.

함수 Pause는 IsFalling = true인 상태에서 충돌 감지가 될 경우, 0.6초 동안 움직이지 못하게 하는 기능입니다.

원래 맨 마지막에 순서대로 작성한 함수인데, 너무 작아서 여기에 붙였습니다. 2장 뒤에 충돌 감지 함수가 나옵니다.

02 제작

캐릭터 컨트롤러 스크립트(TPSCAM.cs)

◆ 캐릭터 점프

02 제작

캐릭터 컨트롤러 스크립트(TPSCAM.cs)

```
private void RunCrouching()
{
    if (Input.GetKey(KeyCode.LeftControl))
    {
        IsCrouch = true;
        IsRun = false;
        speed = 1.5f;
        JumpPower = 5f;
    }
    else if (Input.GetKey(KeyCode.LeftShift))
    {
        IsRun = true;
        IsCrouch = false;
        speed = 5f;
        JumpPower = 4f;
    }
    else
    {
        IsCrouch = false;
        IsRun = false;
        speed = 3f;
        JumpPower = 4f;
```

함수 RunCrouching은 달리기와 앉기 기능을 동시에 함수로 넣었습니다.

왼쪽 컨트롤키 입력 시 앉기가 활성화되고, 동시에 달리기는 비활성화됩니다. 앉았을 때 스피드가 줄어들고, 점프를 더 높이 할 수 있습니다.

왼쪽 쉬프트키 입력 시 달리기가 활성화되고, 앉기는 비활성화됩니다. 달리기 상태에서는 스피드가 증가합니다.

둘 다 입력되지 않은 상태면, 둘 다 비활성화되고 그냥 기본 상태가 됩니다.

02 제작

캐릭터 컨트롤러 스크립트(TPSCAM.cs)

◆ 달리기

02 제작

캐릭터 컨트롤러 스크립트(TPSCAM.cs)

◆ 앉기

02 제작

캐릭터 컨트롤러 스크립트(TPSCAM.cs)

```
private void OnCollisionEnter(Collision collision)
{
    if (collision.gameObject.CompareTag("Ground"))
    {
        IsJumping = false;
        time = 0;
        mySfx.PlayOneShot(jumpSfx);
        if (IsFalling == true)
        {
            IsFalling = false;
            IsPause = true;
        }
    }
}
```

함수 OnCollisionEnter는 충돌 감지 하는 3개의 함수 중 처음으로 충돌 입력이 들어왔을 때(Enter)의 함수입니다.

땅으로 설정되어 있는 객체의 태그를 (“Ground”)로 변경 해놓았고, 캐릭터가 바닥에 닿았을 때, IsJumping의 상태가 그제서야 false가 되고, 공중 체공 시간을 체크하는 time 변수를 0으로 초기화합니다.

mySfx.PlayOneShot(jumpSfx)의 경우 한 번만 소리를 재생하게 하는 코드인데, 점프로 명칭했지만, 땅에 착지했을 때의 소리를 재생하게 했습니다.
그리고, 추락 상태면 초기화하고 IsPause로 추락 후 정지를 작동하기 위해 IsPause를 활성화합니다.

02 제작

캐릭터 컨트롤러 스크립트(TPSCAM.cs)

```
private void Save()
{
    if (Input.GetKey(KeyCode.R))
    {
        Savepoint = transform.position;
    }
    else if (Input.GetKey(KeyCode.T))
    {
        transform.position = Savepoint;
```

게임이 어려워서 R, T로 위치 저장, 저장 위치로 이동을 만들었습니다.
크게 중요하지 않으니 빠르게 넘어가겠습니다.

```
private void CheckFalling()
{
    if (!Physics.BoxCast(transform.position, transform.lossyScale / 2.0f, Vector3.down, transform.rotation, 2f))
    {
        time += Time.deltaTime;

        if (time >= 1.3)
        {
            IsFalling = true;
            IsJumping = true;
        }
    }
}
```

CheckFalling의 경우 RayCast의 발전형인 BoxCast(레이저 끝이 박스 모양)로 2f의 길이 밑의 충돌하는 콜라이더가 존재하지 않은 상태로 1.3초가 지나게 되면 추락 상태로 판단 후 IsFalling과 IsJumping을 활성화 시킵니다.
raycast가 아닌 boxcast인 이유는 블록 모서리에 서있을 경우 멋대로 추락 상태로 변경하는 것을 방지하기 위함입니다.
점프상태 또한 애니메이션에서의 조건문 통일을 위해 활성화 시켰습니다.

02 제작

캐릭터 컨트롤러 스크립트(TPSCAM.cs)

◆ 공중 체공 & 추락

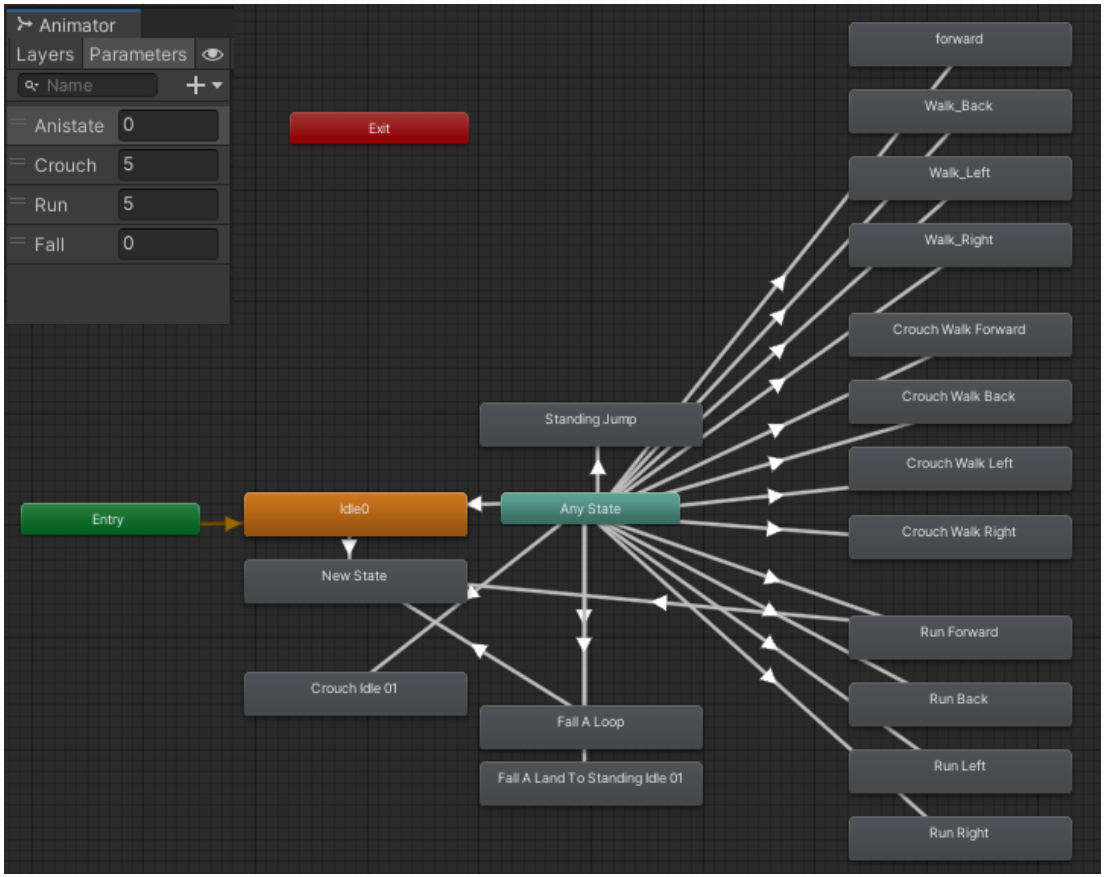
2.2 | 캐릭터 애니메이션 및 애니메이터

애니메이션(Animation)은 MIXAMO에서 그대로 가져왔습니다.

02 제작

캐릭터 애니메이터

애니메이터(Animator)를 살펴보면, Parameters를 Int 형식으로 Anistate(기본 상태: animation statement를 줄인 것입니다), Crouch, Run, Fall을 선언해놨습니다.
기본 상태 및 시작 상태는 Idle이며, Any State와 모든 애니메이션이 연결되어 있습니다.



02 제작

캐릭터 애니메이션 스크립트(Player_Animator.cs)

2.3 | 캐릭터 애니메이션 스크립트(Player_Animator.cs)

```
TPSCAM TPSCAM;

Animator animator;
string animationState = "Anistate";
string CrouchState = "Crouch";
string RunState = "Run";
string FallState = "Fall";

enum States
{
    idle = 0,
    up = 1,
    down = 2,
    left = 3,
    right = 4,
    Jump = 5,
    Phase = 6
}

enum CrouchStates
{
    idle = 0,
    up = 1,
    down = 2,
    left = 3,
    right = 4,
    Phase = 5
}

enum RunStates
{
    up = 1,
    down = 2,
    left = 3,
    right = 4,
    Phase = 5
}

enum FallStates
{
    falling = 1,
    fallingdown = 2,
    Phase = 3
}
```

인스펙터 창의 컴포넌트 Animator를 가져오고, 파라미터 값을 바꾸기 위해서 각각의 파라미터 값의 이름도 선언합니다.

그리고 각 상태에서 취할 행동들을 변수로 선언합니다.

참고로 Phase 상태의 경우는 애니메이션의 겹침을 방지하기 위해서 행동이 변경되었을 때 이전 행동에서는 Phase 상태가 활성화 됩니다.

02 제작

캐릭터 애니메이션 스크립트(Player_Animator.cs)

```
void Update()
{
    transform.eulerAngles = transform.eulerAngles + new Vector3(0, 55, 0);
    if (TPSCAM.IsPause == false)
    {
        UpdateState();
        UpdateCrouchState();
        UpdateRunState();
        UpdateFallState();

        if (TPSCAM.IsJumping == true)
        {
            if (TPSCAM.IsFalling == false)
            {
                animator.SetInteger(animationState, (int)States.Phase);
                animator.SetInteger(CrouchState, (int)CrouchStates.Phase);
                animator.SetInteger(RunState, (int)RunStates.Phase);
                animator.SetInteger(animationState, (int)States.Jump);
            }
            else
            {
                animator.SetInteger(animationState, (int)States.Phase);
                animator.SetInteger(CrouchState, (int)CrouchStates.Phase);
                animator.SetInteger(RunState, (int)RunStates.Phase);
            }
        }
    }
    else
    {
        UpdateFallState();
    }
}
```

길지만 간단하게 설명하자면, 모델의 Rotation이 안맞아서 y축을 55 ° 만 회전했습니다.

TPSCAM에서 static bool 형식으로 선언했던 상태에 따라 State() 함수들이 작동합니다.

IsJumping()을 예로 들자면, 나머지 상태를 비활성화하고, State의 점프만 활성화 하는 것을 볼 수 있습니다.
캐릭터 움직임에 따라 애니메이션이 달라져야 해서 if문이 많이 들어갔습니다.

대충 추락상태에서는 점프 애니메이션이 실행되지 않게 하고, IsPause 상태가 되면 주춤하는 행동을 재생하고 다른 애니메이션이 작동되지 않고 그런 식으로 작성되었습니다.

02 제작

캐릭터 애니메이션 스크립트(Player_Animator.cs)

```
void UpdateState()
{
    if (!TPSCAM.IsCrouch == !TPSCAM.IsRun)
    {
        if (Input.GetKey(KeyCode.D))
        {
            animator.SetInteger(animationState, (int)States.right);
        }
        else if (Input.GetKey(KeyCode.A))
        {
            animator.SetInteger(animationState, (int)States.left);
        }
        else if (Input.GetKey(KeyCode.W))
        {
            animator.SetInteger(animationState, (int)States.up);
        }
        else if (Input.GetKey(KeyCode.S))
        {
            animator.SetInteger(animationState, (int)States.down);
        }
        else
        {
            animator.SetInteger(animationState, (int)States.idle);
        }
    }
    else
    {
        animator.SetInteger(animationState, (int)States.Phase);
    }
}
```

기본 상태에 해당하는 애니메이션입니다.

해당되는 상태가 되면 작동하는 형식이며, 상태에 맞게 enum의 수치를 파라미터 값으로 제공하는 형식입니다.

그러면 애니메이션은 파라미터 값을 따라 애니메이션을 작동하게 됩니다.

중복 방지를 위해 상태를 벗어나면 Phase로 해당되지 않는 파라미터 값으로 변경하고 빠져나옵니다.

다음 장도 이런 느낌이니 특이한 것이 나오지 않는 한 자세한 설명은 하지 않겠습니다.

02 제작

캐릭터 애니메이션 스크립트(Player_Animator.cs)

```
void UpdateCrouchState( )
{
    if (TPSCAM.IsCrouch)
    {
        if (Input.GetKey(KeyCode.D))
        {
            animator.SetInteger(CrouchState, (int)CrouchStates.right);
        }
        else if (Input.GetKey(KeyCode.A))
        {
            animator.SetInteger(CrouchState, (int)CrouchStates.left);
        }
        else if (Input.GetKey(KeyCode.W))
        {
            animator.SetInteger(CrouchState, (int)CrouchStates.up);
        }
        else if (Input.GetKey(KeyCode.S))
        {
            animator.SetInteger(CrouchState, (int)CrouchStates.down);
        }
        else
        {
            animator.SetInteger(CrouchState, (int)CrouchStates.idle);
        }
    }
    else
    {
        animator.SetInteger(CrouchState, (int)CrouchStates.Phase);
    }
}
```

앉기 상태에 해당되는 애니메이션입니다.

02 제작

캐릭터 애니메이션 스크립트(Player_Animator.cs)

```
void UpdateRunState()
{
    if (TPSCAM.IsRun)
    {
        if (Input.GetKey(KeyCode.D))
        {
            animator.SetInteger(RunState, (int)RunStates.right);
        }
        else if (Input.GetKey(KeyCode.A))
        {
            animator.SetInteger(RunState, (int)RunStates.left);
        }
        else if (Input.GetKey(KeyCode.W))
        {
            transform.eulerAngles = transform.eulerAngles + new Vector3(0, -30, 0);
            animator.SetInteger(RunState, (int)RunStates.up);
        }
        else if (Input.GetKey(KeyCode.S))
        {
            animator.SetInteger(RunState, (int)RunStates.down);
        }
    }
    else
    {
        animator.SetInteger(RunState, (int)RunStates.Phase);
    }
}
```

달리기 상태에 해당되는 애니메이션입니다.

앞으로 달리는 모션만 이상하게 휘어 있어 각도 조절을 하였습니다.
(55 ° + (-30) ° = 25 ° 조절)

02 제작

캐릭터 애니메이션 스크립트(Player_Animator.cs)

```
void UpdateFallState()
{
    if (TPSCAM.IsFalling == true)
    {
        animator.SetInteger(FallState, (int)FallStates.falling);
    }
    else if (TPSCAM.IsPause == true)
    {
        animator.SetInteger(FallState, (int)FallStates.fallingdown);
    }
    else
    {
        animator.SetInteger(FallState, (int)FallStates.Phase);
    }
}
```

추락 상태에 해당되는 애니메이션입니다.
IsFalling 상태에선 추락 루프 애니메이션이, IsPause 상태에선 추락 후 대기모션이 실행됩니다.

다 작성한 뒤에 봤는데, Phase가 아니라 Pause입니다.
수정은 했는데 다시 캡쳐하기 뭐해서 그대로 진행하겠습니다.

03 실행

끈기의 숲(Forest of Patience) 실행 동영상

PPT 크기가 커져서 30MB 제한이 될 수 있어서 따로 파일로 올리겠습니다.

04 마무리

스크립트 제작 도움 및 출처

참고 및 출처

모델 및 애니메이션 - MIXAMO(www.mixamo.com)
캐릭터 움직임 및 카메라 이동 부분 - (wergia.tistory.com/230)
스크립트 제작 Unity 공식 문서(docs.unity3d.com)

이미지 출처

메이플스토리 게임 슬리피우드 배경 음악 및 배경 텍스처
잔디 텍스처 – 구글 검색 후 캡처해서 가져 옴

부득이하게 PPT 파일이 커서 몇 개의 GIF는 저 화질로 대체했습니다.