

# 가상세계 Feeding Animals 과제 보고서

2024.5.23

2020203052

소프트웨어학부 권빈

|         |                                   |   |
|---------|-----------------------------------|---|
| Easy1   | Vertical Player Movement          | O |
| Easy2   | Animals increasing proportionally | O |
| Easy3   | The food no longer moves          | O |
| Easy4   | Camera switcher                   | O |
| Medium  | Animal Hunger Bar                 | O |
| Hard    | Move to food                      | O |
| Expert1 | Move freely                       | O |
| Expert2 | Collision avoidance               | △ |

## 1. Easy1 Vertical Player Movement

```
// Update is called once per frame
참조 0개
void Update()
{
    horizontalInput=Input.GetAxis("Horizontal");
    verticalInput=Input.GetAxis("Vertical");//상하 키보드 입력받기
    transform.Translate(Vector3.right*horizontalInput*Time.deltaTime*speed);
    transform.Translate(Vector3.forward*verticalInput*Time.deltaTime*speed);//이동 동작

    if(transform.position.x<=-xRange){
        transform.position=new Vector3(-xRange,transform.position.y,transform.position.z);
    }
    if(transform.position.x>xRange){
        transform.position=new Vector3(xRange,transform.position.y,transform.position.z);
    }
    if(transform.position.z<=-zRange){//아래쪽으로 최대로 갈 수 있는 영역지정
        transform.position=new Vector3(transform.position.x,transform.position.y,-zRange);
    }
    if(transform.position.z>zRange){//위쪽으로 최대로 갈 수 있는 영역지정
        transform.position=new Vector3(transform.position.x,transform.position.y,zRange);
    }
}
```

verticalInput 이라는 변수 추가후 상하 키보드 입력을 받게하여  
입력에 따라 캐릭터의 z값 위치를 변경시키는 방법으로 캐릭터의 상하 움직임을  
추가해 주었다.  
x축 움직임과 마찬가지로 최대 Range를 설정하여 정해진 영역 밖으로 나가지 못하게  
만들었다.

## 2. Animals increasing proportionally

```
void Start()
{
    InvokeRepeating("SpawnAnimal",startDelay,spawnInterval);
}

// Update is called once per frame
참조 0개
void Update()
{
    참조 0개
    void SpawnAnimal(){
        Vector3 spawnPos= new Vector3(Random.Range(-spawnRangeX,spawnRangeX),0,Random.Range(-spawnRangeZ,spawnRangeZ));

        if(animals1num>0){
            Instantiate(animalsPrefabs[0],spawnPos,animalsPrefabs[0].transform.rotation);
        }

        spawnPos= new Vector3(Random.Range(-spawnRangeX,spawnRangeX),0,spawnPosZ);
        if(animals2num>0){
            Instantiate(animalsPrefabs[1],spawnPos,animalsPrefabs[1].transform.rotation);
        }

        spawnPos= new Vector3(Random.Range(-spawnRangeX,spawnRangeX),0,spawnPosZ);
        if(animals3num>0){
            Instantiate(animalsPrefabs[2],spawnPos,animalsPrefabs[2].transform.rotation);
        }
    }
}
```

InvokeRepeating을 이용해 SpawnAnimal을 주기적으로 실행시키고  
SpawnAnimal에서 각 해당하는 동물이 살아있으면 같은 종류의 동물을 Instantiate를  
이용하여  
하나를 생성시킨다.

## 3. The food no longer moves

```
if(Input.GetKeyDown(KeyCode.Space)){
    //Launch a projectile from the player
    Instantiate(projectilePrefab,transform.position,projectilePrefab.transform.rotation);
}
```

스페이스 바를 누르면 뼈다귀가 플레이어의 위치에 생성되게 만들었고  
기존 수업에서는 moveforward를 통해 앞으로 이동했지만 컴포넌트를 삭제해주어  
더이상 이동하지 않게 만들어 주었다

## 4.Camera switcher

```
void LateUpdate()
{
    //쉬프트를 누르면 카메라의 시점을 바꿈
    //캐릭터가 보는 정면값으로
    //시점도 perspective로 바뀌어야함
    if (Input.GetKeyDown(KeyCode.LeftShift) || Input.GetKeyDown(KeyCode.RightShift))
    {
        cameranum++;

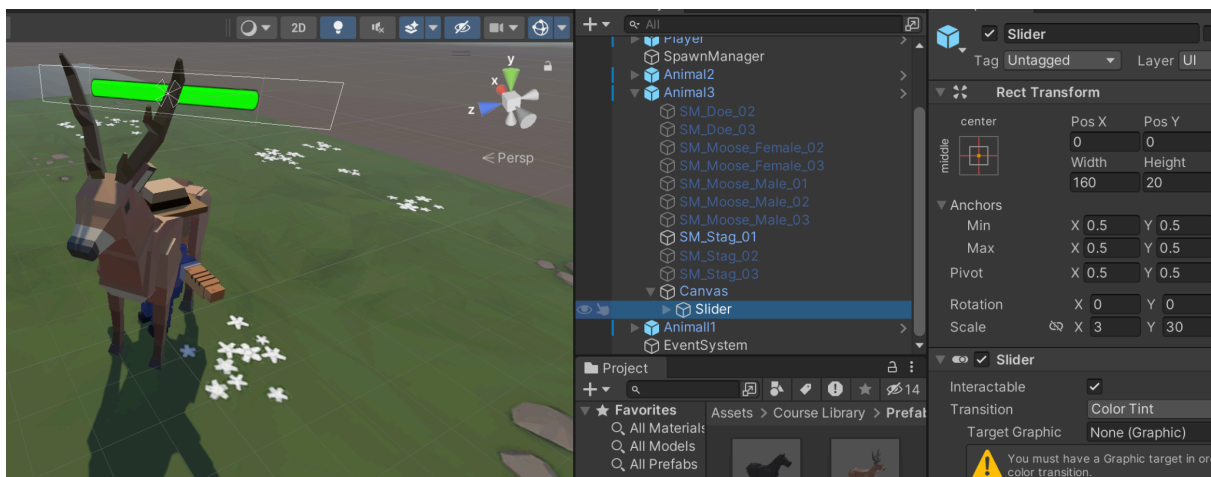
        if(cameranum%2==0){//원래 시점
            transform.position=new Vector3(0,25,10);
            transform.rotation=Quaternion.Euler(90,0,0);
            cameraCon.orthographic=true;

        }else{//1인칭 시점
            transform.position=player.transform.position+new Vector3(0,3,0);
            transform.rotation=player.transform.rotation;
            cameraCon.orthographic=false;
        }
    }
}
```

쉬프트를 누르면 카메라조정 변수를 증가시켜서 위에서 내려다보는 3인칭 시점과 농부의 눈이되는 1인칭 시점으로 바뀌게 한다.

또 바꿀때 카메라의 위치와 각도뿐만아니라 **ortographic**을 이용해 **Projection**도 바꿔준다.

## 5. Animal Hunger Bar



각 동물들의 Prefeb에 world space 모드인 canvas를 넣어주고 slide UI를 추가해 이를이용해 배고픔 게이지를 만들었다.

```

void Update()
{
    if(bhungry>0){
        behungry-=Time.deltaTime;
    }else{
        curhunger--;
        behungry=10;
    }

    UdateHungerbar();

    if(curhunger==0){
        if(animal.tag=="animal1"){
            SpawnManager.animal1num--;
        }else if(animal.tag=="animal2"){
            SpawnManager.animal2num--;
        }else if(animal.tag=="animal3"){
            SpawnManager.animal3num--;
        }
        Destroy(animal);
    }
}

참조 1개
private void UdateHungerbar(){
    hungerbar.value=(float)curhunger / (float)maxhunger;
}

```

일정 시간마다 **curhunger**의 값을 내리며 게이지를 관리 하였고 0이 되면  
 스폰메니저의 해당 동물변수를 줄인뒤 삭제하였다. **slider**의 값을 (현재 게이지/전체  
 게이지)를 하여서  
 바뀌는 배고픔 수치에 따라서 **slider**의 바가 업데이트 되게 하였다.

## 6. Move to food

```
void Update()
{
    Collider[] hitColliders = Physics.OverlapSphere(transform.position, detectionRadius);
    Transform closestFood = null;
    float closestDistance = Mathf.Infinity;

    foreach (var hitCollider in hitColliders)
    {
        if (hitCollider.CompareTag("food"))
        {
            float distance = Vector3.Distance(transform.position, hitCollider.transform.position);
            if (distance < closestDistance)
            {
                closestDistance = distance;
                closestFood = hitCollider.transform;
            }
        }
    }
}
```

overlapSphere를 이용하여 일정 범위 안에있는 콜라이더들을 탐지하고 저장한다  
hitCollider에 저장된 요소들을 하나씩 탐색하며 태그가 food이고 거리가 가장 가까운  
요소의 위치를 저장한다.

```
if (closestFood != null)
{
    Vector3 direction = closestFood.position - transform.position;
    Quaternion rotation = Quaternion.LookRotation(direction);
    transform.rotation = Quaternion.Slerp(transform.rotation, rotation, Time.deltaTime * rotationSpeed);

    Vector3 moveDirection = closestFood.position - transform.position;
    moveDirection.y = 0;
    transform.position = Vector3.MoveTowards(transform.position, transform.position + moveDirection, speed * Time.deltaTime);

    if (Vector3.Distance(transform.position, closestFood.position) < 0.1f)
    {
        Destroy(closestFood.gameObject);
        closestFood=null;
        if(curhunger<3){
            curhunger++;
        }
    }
}
```

위치가 설정되어있으면 해당위치로 Lookrotation을 이용해 방향을 돌리고

MoveToward를 이용해 해당위치로 이동한다.

음식을 만나게 되면 음식을 삭제시키고 배고픔 수치를 올려준다.

또한 closestFood를 null로만들어주어 다시 자유롭게 움직이며 음식을 탐색할 수 있게  
만들어준다.

## 7. Move freely

```
void Start()
{
    targetPosition = transform.position;
    // 랜덤한 회전을 시작합니다.
    StartCoroutine(ChangeRotationRoutine());
}

// Update is called once per frame
참조 0개
void Update()
{
    targetPosition += transform.forward * moveSpeed * Time.deltaTime;

    targetPosition.x = Mathf.Clamp(targetPosition.x, -20f, 20f);
    targetPosition.z = Mathf.Clamp(targetPosition.z, -10f, 30f);

    transform.position = Vector3.Lerp(transform.position, targetPosition, smoothMoveTime);

    transform.rotation = Quaternion.Slerp(transform.rotation, targetRotation, smoothRotateTime * Time.deltaTime);
}
참조 1개
IEnumerator ChangeRotationRoutine()
{
    while (true)
    {
        targetRotation = Quaternion.Euler(0, Random.Range(0, 360), 0);

        yield return new WaitForSeconds(rotationChangeInterval);
    }
}
```

코루틴을 이용해 일정시간마다 랜덤한 회전 방향을 가지게 만들고 회전한 방향을 기준으로 계속 앞으로 가게 만들었다. 이때 **Clamp**를 이용하여서 완전히 화면밖으로 나가지 못하게 설정하였다.

## 8. Collision avoidance

일단 아이디어는 앞으로만 움직이니 다른 동물이 앞에있는지 판단하고 있으면 방향을 돌리자는 생각이였다.




따라서 일단 각 동물들의 박스 콜라이더의 길이를 늘려주었다

```
private void OnCollisionEnter(Collision collision)
{
    targetRotation = Quaternion.Euler(0, Random.Range(0, 360), 0);
}
```

그 뒤에 위에 자유롭게 움직이게 할때 사용한 **targetRotation**을 재설정하여 앞에 물체가 있으면 바라보고 있는 동물이 방향을 바꾸게 하였다.

## Comment

과제를 하면 할수록 느끼는건데 진짜 유니티 안에 존재하는 것들이 엄청 많다. 모르는 것들을 해결하면서 이것 저것 알아가는게 재미가 있었던 것 같다. 마지막 **Collision avoidance**는 비슷한 거리에서 음식을 향해 다수의 동물이 달려갈때 겹치는 현상을 어떻게 해결해야할지 모르겠어서 아쉬웠다.

 가상세계 과제 2 시연영상

<https://youtu.be/Ee0MXFjUIEA?si=TSsvpwgUzES4iSg3>