가상현실시뮬레이션

최권택 교수님

201900000 조우현
201900000 서OO
202100000 한OO

2024. 06. 12.
기말고사 프로젝트

평가기준 

1. 캐릭터 이동 시뮬레이션 알고리즘과 훈련 시스템 적용 방안에 대해 설명하시오.
2. 물리엔진 시뮬레이션 방안에 대해 설명하시오.
3. 스토리 진행을 시뮬레이션하기 위한 스크립트 언어의 확장 방법에 대해 설명하시오.

목차

개요

제한 시간 내에 **퀴즈**를 **풀고**
친구들과 함께 미로를 **탈출하라!**

C# 상식 OX 퀴즈를 풀면서
미로를 탈출하는 교육용 서바이벌게임

주어진 시간은 단 5분!

친구 5명을 구하고 미로를 탈출하자!

개발 환경

제작

조우현 / 서OO / 한OO

지원기기

Windows

개발기간

~ 2024. 06. 11.

개발사양

Unity

캐릭터 이동 시뮬레이션

01 달리기

02 코드

Base로 StarterAsset 활용 / Mixamo 모델링 활용

캐릭터의 조작과 UI 상호작용의 원활함을 위해
UI가 활성화되거나, Left alt키를 누르고 있을 때만
마우스 포인터가 활성화 된다.

```
if (GameManager.Instance.isGameOver == false)
    {
        Cursor.visible = false;
        Cursor.lockState = CursorLockMode.Locked;
    }

    if (Input.GetKey(KeyCode.LeftAlt))
    {
        Cursor.visible = true;
        Cursor.lockState = CursorLockMode.None;
    }
```

캐릭터 이동 시뮬레이션

01 일반 포즈, 슈팅 포즈

02 코드

```
//우클릭을 할 때는 에임 이미지를 활성화하고 시네머신 카메라를 전환해
캐릭터를 화면의 왼쪽에 치우치게 하여 에임 타겟이 잘 보이도록 함.

private void HandleAttackAndFire()
{
    if (input.attack)
    {
        if (!isAttackingIdle)
        {
            animator.SetBool("IsAttackingIdle", true);
            isAttackingIdle = true;

            // 에임 활성화
            uiCanvas.SetActive(true);

            // 시네머신 카메라 전환
            mainCamera.Priority = 0;
            attackCamera.Priority = 1;
        }
```

```
if (input.fire)
    {
        animator.SetFloat("AttackSpeed", 1f);
        FireBullet();
        input.fire = false;
    }
  }
  else
  {
    if (isAttackingIdle)
    {
        animator.SetBool("IsAttackingIdle", false);
        isAttackingIdle = false;

        // 에임 비활성화
        uiCanvas.SetActive(false);

        // 시네머신 카메라 전환
        mainCamera.Priority = 1;
        attackCamera.Priority = 0;
    }
  }
}
```

캐릭터 이동 시뮬레이션

01 퀴즈 맞추기

총을 쏘고 퀴즈를 맞춘 모습

02 코드

```
// 총알 발사를 위해 레이캐스트로 메인카메라가 바라보는 방향의 정 가운데를 타겟하고 캐릭터 또한 해당 방향을 바라보게 됨. 또한 트리거를 무시할 수 있도록 하여 의도치 않게 레이캐스트가 트리거에 막히는 현상을 방지함.

 private void FireBullet()
   {
       Vector3 targetPosition = Vector3.zero;
       Transform camTransform = mainCameraComponent.transform;
       RaycastHit hit;

       if (Physics.Raycast(camTransform.position,
camTransform.forward, out hit, Mathf.Infinity, targetLayer,
QueryTriggerInteraction.Ignore))
       {
           targetPosition = hit.point;
           aimObj.transform.position = hit.point;
       }
       else
       {
           targetPosition = camTransform.position +
camTransform.forward * aimObjDis;
           aimObj.transform.position = camTransform.position +
camTransform.forward * aimObjDis;
       }
```

총을 쏘고 퀴즈를 틀린 모습

```
        Vector3 direction = (targetPosition -
firePoint.position).normalized;

        // 플레이어를 타겟 방향으로 회전
        Vector3 lookDirection = new Vector3(direction.x, 0,
direction.z);
        transform.rotation =
Quaternion.LookRotation(lookDirection);

        // 총 발사
        GameObject bullet = Instantiate(bulletPrefab,
firePoint.position, Quaternion.LookRotation(direction));
        Rigidbody rb = bullet.GetComponent<Rigidbody>();
        rb.AddForce(direction * bulletSpeed,
ForceMode.Impulse);
    }
```

캐릭터 이동 시뮬레이션 (NPC)

01 AI Navigation 활용

NPC와 상호작용 후 네비게이터

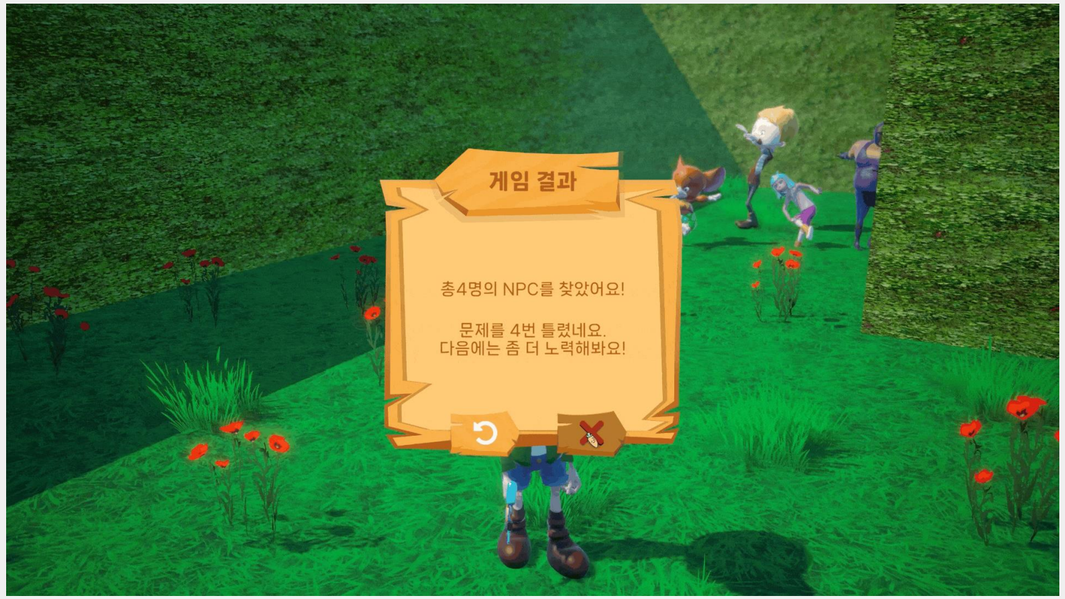
02 코드

// 강의시간에 사용했던 AI Navigation 기능과 스크립트를 활용하여,
첫번째 마주치는 NPC는 게임 시작과 동시에 캐릭터에게 찾아오며
그 후 계속 팔로잉함.

나머지 4개의 NPC는 미로 곳곳에 존재하며 가까이 다가가 트리거에 접촉하면
플레이어와 대화를 하는 스크립트와 플레이어를 따라가는 코루틴 함수가
실행됨.

플레이어와의 대화는 최초 단 한 번만 동작하기 위해 bool값을 활용

```
if (other.gameObject.CompareTag("Player"))
    {
        if (moveToPlayerCoroutine != null)
        {
            StopCoroutine(moveToPlayerCoroutine);
        }
        moveToPlayerCoroutine = StartCoroutine(MoveToPlayer());
    }
```



NPC의 댄스

```
 private void OnTriggerEnter(Collider other)
   {
      if (other.gameObject.CompareTag("Player") &&
justOneCheck == false)
      {
         GameManager.Instance.FindNPC++;
         string script =
Resources.Load<TextAsset>(filename).ToString();
         StartCoroutine(engine.PlayScript(script));
         justOneCheck = true;
      }
   }
```

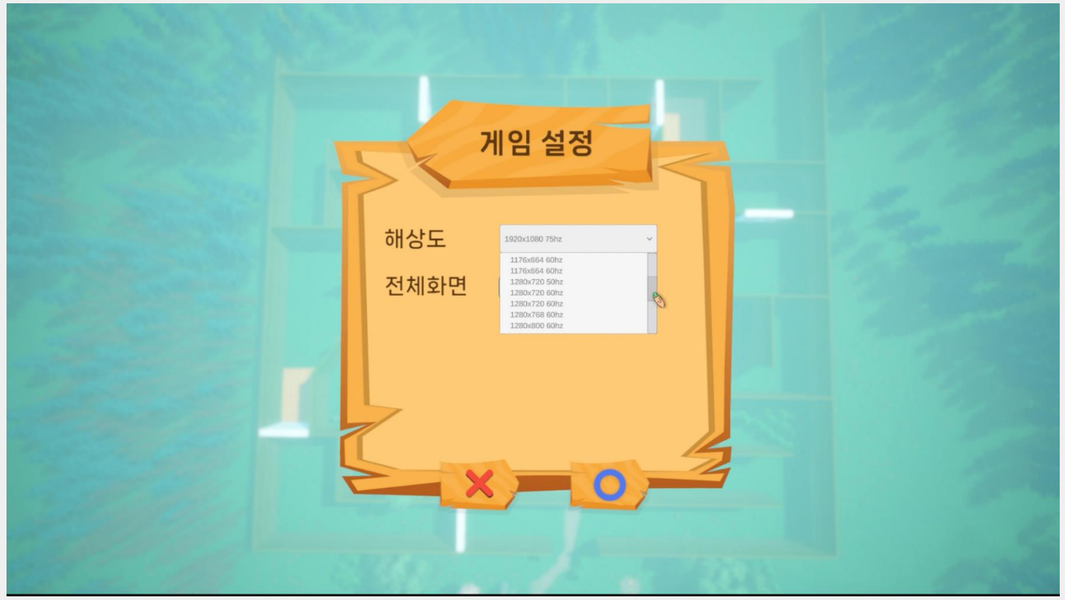
//또한 게임을 클리어하면 NPC는 플레이어를 따라오는
코루틴을 종료하고 그 자리에서 애니메이션(춤)을 재생함.

```
 public void StopMovementAndPlayAnimation()
   {
      if (moveToPlayerCoroutine != null)
      {
         StopCoroutine(moveToPlayerCoroutine);
         moveToPlayerCoroutine = null;
      }

      animator.Play(animationName);
   }
```

UI

시작 화면


게임 설정
해상도
1920x1080 75hz
1176x664 60hz
1176x664 60hz
1280x720 50hz
1280x720 60hz
1280x720 60hz
1280x768 60hz
1280x900 60hz
전체화면

환경 설정

UI

키 가이드 게시판

퀴즈 풀기

물리엔진 시뮬레이션

물에 빠지면 돌아가는 블록

02 코드

// 강의 시간에 사용했던 세이브블록과 체크포인트 블록을 활용함.
다만 npc가 캐릭터를 따라오므로, 태그로 플레이어가 빠진 것이 맞는 지 확인하는 코드를 추가하여 의도치 않은 현상 방지

#DeathBlock.cs

```
if (other.gameObject.CompareTag("Player"))
    {
        StartCoroutine("Do");
    }
```

#CheckPointBlock.cs

```
if (other.gameObject.CompareTag("Player"))
    {
        ori = transform.position;
    }
```

물리엔진 시뮬레이션

진흙탕 블록

사라지는 블록

물리엔진 시뮬레이션

불꽃놀이와 화면 엔딩

02 코드

//게임을 클리어했을 시 게임 클리어 지점에 있는 트리거를 'Player' 태그를 갖고있는 오브젝트인 플레이어가 밟으면 게임 결과 팝업이 나타나면서 스카이박스가 변경되고 불꽃놀이와 축하음악이 시작됨.

또한 ScenarioEngine을 활용해 전면에 위치한 전광판에 축하영상이 나타남. 또한 플레이어가 뒤로 가서 다시 트리거를 밟을 경우를 대비하여 딱 한번만 실행할 수 있도록 bool값을 활용함.

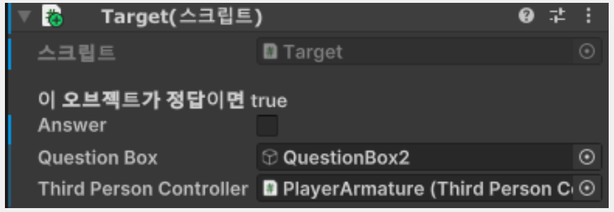
```
private void OnTriggerEnter(Collider other)
  {
      if (other.gameObject.CompareTag("Player") && justOneCheck
== false)
      {
          GameManager.Instance.PlaySound(congBgm);
          GameManager.Instance.GameClear();
          string script =
Resources.Load<TextAsset>("GameClear").ToString();
          StartCoroutine(engine.PlayScript(script));
          justOneCheck = true;
      }
  }
```

물리엔진 시뮬레이션

-

총을 쏘고 퀴즈를 맞춘 모습

02 코드



//문제에 대한 정답을 맞출 때 과녁에 물총을 맞추게 되는데,
이를 모듈화하여 하나의 스크립트를 통해 과녁의 O, X를 모두 처리할 수 있음.

맞췄을 경우에는 하나의 오브젝트로 묶여있는 QuestionBox 전체를
비활성화시켜 길을 열어주고, 못 맞췄을 경우에는 5초간 제자리에서 움직일 수
없음.

```
private void OnTriggerEnter(Collider other)
  {
     if (other.gameObject.CompareTag("Bullet"))
     {
        if (Answer) //O 과녁 동작
        {
           GameManager.Instance.Target_Success();
           QuestionBox.SetActive(false);
        }
```

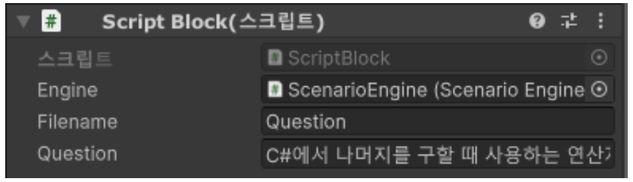
총을 쏘고 퀴즈를 틀린 모습

```
else // X 과녁 동작
        {
            if (!Failed) {
                Failed = true;
                GameManager.Instance.Target_Fail();
                moveSpeedSave = ThirdPersonController.MoveSpeed;
                sprintSpeedSave = ThirdPersonController.SprintSpeed;
                ThirdPersonController.MoveSpeed = 0;
                ThirdPersonController.SprintSpeed = 0;
                Invoke("ReturnValue", 5);
            }
        }
    }
}

public void ReturnValue()
{
    ThirdPersonController.MoveSpeed = moveSpeedSave;
    ThirdPersonController.SprintSpeed = sprintSpeedSave;
    Failed = false;
}
```

스토리 진행 시뮬레이션

01 코드



//시나리오엔진 코드 확장 1
시나리오 엔진 코드를 확장해 에디터상에서 string을 public으로 받고, 이 값을 시나리오 엔진에 전달하여 퀘스트를 표시할 수 있도록 개선함.

#ScriptBlock.cs

```
private void OnTriggerEnter(Collider other)
  {
     if (other.gameObject.CompareTag("Player"))
     {
        string script =
Resources.Load<TextAsset>(filename).ToString();
        StartCoroutine(engine.PlayScript(script, question));
     }
  }
```

#ScenarioEngine.cs 일부

```
public IEnumerator PlayScript(string script, string question = "")
```

function이 question일 경우 별도로 string을 전달받고, 아닐 경우 기본값을 넣어 기존의 코드에 영향이 없게하고 다른 function에 영향이 없도록 함

//시나리오 엔진 확장 2

function이 끝난 후 Canvas가 부자연스럽게 확 꺼지는 문제를 방지하기 위해 Canvas의 alpha값을 조절하여 Fade Out 기능을 추가함.

```
private IEnumerator FadeCanvas(bool fadeIn, float duration)
  {
     float startAlpha = fadeIn ? 0f : 1f;
     float endAlpha = fadeIn ? 1f : 0f;
     float elapsedTime = 0f;

     while (elapsedTime < duration)
     {
        elapsedTime += Time.deltaTime;
        float alpha = Mathf.Lerp(startAlpha, endAlpha, elapsedTime /
duration);
        canvasGroup.alpha = alpha;
        yield return null;
     }

     canvasGroup.alpha = endAlpha;
  }
```

문제마다 새로운 Script.txt 정의 필요 X

감사합니다

201900000 조우현
201900000 서OO
202100000 한OO

2024. 06. 12.
기말고사 프로젝트