|  |
| --- |
|  |
| 조이스틱 실습 |
|  |
| 지능형 인터페이스 |

**교수 : 김상연**

3조

조장 : 김영규

팀원 : 이재영

팀원 : 여창민

조이스틱 실습

지능형 인터페이스

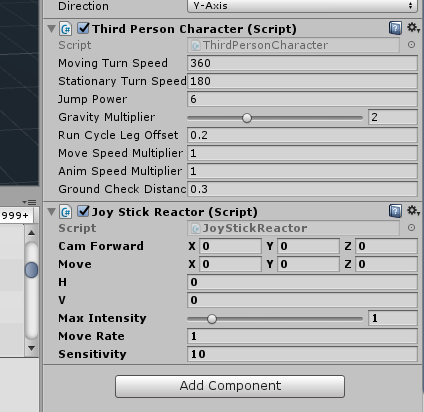
1. 과제 설명
2. 스크립트 코드(주석 포함 및 소스 코드 설명)
3. 고찰
4. 구동 동영상 별첨
5. 과제 설명

. 유니티의 기본 에셋인 ThirdPersonCharacter를 조이스틱을 사용하여 움직이도록 구현하시오. 기존의 ThirdPersonCharacter는 키보드를 사용하여 조작하였다. FPS게임처럼 W버튼을 누르면 앞으로, S버튼은 뒤로, A버튼은 왼쪽으로, D버튼은 오른쪽으로. 이런 ThirdPersonCharacter를 지난 실습이었던 조이스틱 실습과 연동을 하는 것이 이번 과제였다. 키보드 WASD가 해야할 역할을 조이스틱의 움직임으로 대체하는 것이다. 조이스틱에 움직임을 주었을 때 아두이노와 컴퓨터는 그 움직임을 인식하고 유니티에 전송하여 유니티에서는 ThirdPersonCharacter 에 움직임을 가한다.

1. 스크립트 코드

JoysticReactor

|  |
| --- |
| namespace Ardunity  {  [RequireComponent(typeof(ThirdPersonCharacter))] //ThirdPersonUserControl 복사  [AddComponentMenu("ARDUnity/Reactor/Effect/JoyStickReactor")]  public class JoyStickReactor : ArdunityReactor  {  //ThirdPersonUserControl에 있는 변수 복사-----------------------------------  private ThirdPersonCharacter m\_Character;  private Transform m\_Cam;  public Vector3 m\_CamForward;  public Vector3 m\_Move;  private bool m\_Jump;  // 좌우 입력에 따라 변화할 값  public float h = 0;  public float v = 0;  //----------------------------------------------------------------------------    // 초기화  private void Start()  {  // get the transform of the main camera  if (Camera.main != null)  {  m\_Cam = Camera.main.transform;  }  else  {  Debug.LogWarning(  "Warning: no main camera found. Third person character needs a Camera tagged \"MainCamera\", for camera-relative controls.", gameObject);  // we use self-relative controls in this case, which probably isn't what the user wants, but hey, we warned them!  }  // ThirdPersonCharacter객체와 m\_Character을 연결  m\_Character = GetComponent<ThirdPersonCharacter>();  }  .  .  .  // 조이스틱Update함수  void Update()  {  if (\_analogInput\_x != null && \_analogInput\_y != null)  {  //move object  Debug.Log("X : " + ((int)\_x).ToString() + " / Y : " + ((int)\_y).ToString());  switch (GetDirection())  {  //GetDirection함수에서 결정된 경로대로 움직이게 함.  //h와 v값을 조이스틱 움직이는 방향에 따라 조정  case 1: // 앞왼  h = 10; v = 10; break;  case 2: // 앞  h = 0; v = 10; break;  case 3: // 앞오  h = -10; v = 10; break;  case 4: // 왼  h = 10; v = 0; break;  case 5: h = 0; v = 0; break; //정지  case 6: // 오  h = -10; v = 0; break;  case 7: // 뒤왼  h = 10; v = -10; break;  case 8: // 뒤  h = 0; v = -10; break;  case 9: // 뒤오  h = -10; v = -10; break;  }  }  }  //조이스틱GetDirection  private int GetDirection()  {  // 1 2 3  // 4 5 6  // 7 8 9 방향으로 움직이게 함  if ((int)\_x < 4) // 왼  {  if( (int)\_y > 4 ) { return 3; } // 앞  else if ((int)\_y < 4 ) { return 9; } // 뒤  else { return 6; } // 왼쪽으로만  }  else if ((int)\_x > 4) // 오  {  if ((int)\_y > 4) { return 1; } // 앞  else if ((int)\_y < 4) { return 7; } // 뒤  else { return 4; } // 오른쪽으로만  }  else  {  if ((int)\_y > 4) { return 2; } // 앞  else if ((int)\_y < 4) { return 8; } // 뒤  else { return 5; } // 정지  }  }  .  .  .  //ThirdPersonUserControl의 FixedUpdate함수 변경  public void FixedUpdate()  {  bool crouch = Input.GetKey(KeyCode.C);  // calculate move direction to pass to character  if (m\_Cam != null)  {  // calculate camera relative direction to move:  m\_CamForward = Vector3.Scale(m\_Cam.forward, new Vector3(1, 0, 1)).normalized;  m\_Move = v \* m\_CamForward + h \* m\_Cam.right;  }  else  {  // we use world-relative directions in the case of no main camera  m\_Move = v \* Vector3.forward + h \* Vector3.right;  }  #if !MOBILE\_INPUT  // walk speed multiplier  if (Input.GetKey(KeyCode.LeftShift)) m\_Move \*= 0.5f;  #endif  // pass all parameters to the character control script  m\_Character.Move(m\_Move, crouch, m\_Jump);  m\_Jump = false;  }  }  } |



ThirdPersonUserControl 컴포넌트를 제거한 후 JoyStickReactor의 변수와 ThirdPersonCharacter 객체를 연결해 캐릭터가 움직이도록 하였다.

1. 고찰

이번 과제는 단독적인 하나의 과제라기 보다는 지난 과제와 연계를 해서 해결하는 문제였다. 아직까지도 유니티 사용법이 익숙지 않았기에 다른 과제들 보다도 해결하는데 시간이 오래 걸렸다. ThirdPersonCharacter 의 움직임과, 조이스틱의 움직임 각기는 제어할 수 있었지만 그 둘을 잇는데 힘듦이 많았다. 하지만 차근차근 하나씩 뜯어보고 해결하니 언젠가는 문제가 해결되었다. 앞으로 단독적인 문제보다는 이런 식으로 연계된 문제가 많을 터인데 차분히 해결해 나가야겠다.