<절대 강좌! 유니티> 정리본

시작 날짜 : 2023.10.04

날짜 : 10.04

Page 1~16

게임 엔진 : 언리얼, 크라이, 하복, 게임 브리오, 소스 엔진 등..

뷰의 종류 – 프로젝트 뷰, 씬 뷰, 하이러키 뷰, 인스펙터 뷰, 게임 뷰, 콘솔 뷰

프로젝트 뷰: 파일 탐색기, 모든 파일들을 모아놓음.

하이러키 뷰 : 오브젝트들을 모아놓는 곳

인스펙터 뷰 : 오브젝트의 속성을 보는 곳

게임 뷰 : 게임 플레이시 보는 화면

콘솔 뷰 : 디버깅 시 로그를 출력하는 뷰(정보, 경고, 오류) Ctrl + Shift + C

하이러키 뷰는 게임오브젝트를 다른 게임오브젝트로 드래그 앤드 드롭해 차일드화할 수 있는 기능이 있는데 이를 **페어런팅** 이라 한다.

하이러키 뷰의 정렬 방식–Transform sort(생성된 순서대로), alphabetical sort(알파벳 순서대로 정렬

락킹 : 하이러키 뷰에서 특정 게임오브젝트를 선택할 수 없도록 한다.(씬 뷰에서만 선택 못함)

룩 : 하이러키 뷰에서 시각적으로 표시하지 않게 한다.

Play : Ctrl + P

Pause : Ctrl + Shift + P

Step : kCtrl + Alt + P (한 프레임씩 끊어서 실행하는 기능)

검색 : Ctrl + K (임포트한 리소스 찾을 때 사용)

날짜 : 10.12

Page 16~85

오브젝트 생성할 때마다 월드 좌표 (0,0,0) : orgin 에 설정하는 방법

Edit – Preferences – Scene View – Create Objects at Origin

캡슐의 크기를 사람의 표준 신장으로 생각하고 작업하기

유니티의 1유닛을 1m라 생각하고 작업하라는 뜻.

텍스처 : 3D 모델의 표면에 매핑시킬 이미지 파일

텍스처의 크기는 가로세로 2^n (1024 x 1024)형태일 때만 암축 지원, 속도가 가장 빠름

모바일에서 속도를 향상시키고 싶다면 2^n(POT) 형태의 텍스처 사용해야 함.

머터리얼 : 3D 모델에 적용할 텍스처의 다양한 속성을 설정하는 역할을

분홍색으로 변한 3D 모델 : 3D모델에 텍스처를 저장하고 있는 머터리얼이 연결되지 않았거나 끊어졌다는 뜻.

머터리얼에 텍스처를 지정하는 방법

1. 해당 텍스처를 Albedo, normal Map 슬롯에 연결한다.
2. 텍스처 슬롯 바로 옆에 있는 원(브라우저 버튼)을 클릭해 직접 텍스처를 선택한다.

~Render 계열의 컴포넌트 : 텍스처 정보를 가진 머터리얼을 연결하는 속성이 있다는 것을 뜻함.

셰이더 : 렌더링할 때 픽셀의 농담, 색조, 명암을 결정하는 프로그래밍 방식

렌더링 모드(Rendering Mode) : 불투명, 그물망 표현, 홀로그램 표현, 투명

알베도(Albedo) : 빛을 반사하는 정도, 반사율

메탈릭 속성(Metarlic) : 객체 표면에 금속의 재질을 표현하기 위한 텍스처

노멀 맵(Normal Map) : 표면의 세밀한 입체감이나 질감을 표현하기 위한 텍스처의 일

하이트 맵(Height Map) : 텍스처로 높낮이를 표현하는

오클루전(Occlusion) : 흑백의 텍스처로 간접조명에 의해 싱기는 명암을 더욱 뚜렷이 표시해 사물의 입체감과 깊이감을 살리는데 사용함.

이미션(Emission) : 스스로 빛을 방출하는 속성을 말

디테일 마스크(Detail Mast) : Secondary Maps에 적용할 마스크를 설정하는 텍스처 슬롯

날짜 : 10.13

Page 85~112

프리팹(Prefab) :자주 사용하는 객체를 미리 부품처럼 만들어놓고 재사용할 수 있게 하는 것

복사 : Ctrll + D

복제되는 프리팹 네이밍 방법 바꾸기 : Project Settings – Editor – Nubering Scheme – Game Object Naming

정점 스내핑(Vertex Snapping) : 메시의 꼭짓점과 꼭짓점을 붙이는 기능

오브젝트를 move툴로 선택 후 V를 누른상태에서 마우스 커서를 꼭짓점으로 이동 -> 꼭짓점 가운데 네모를 원하는 오브젝트 쪽으로 드래그해서 스내핑한다.

조명 - Directional Light, Point Light, Spot Light, Area Light

Directional Light : 전체 화면에 균일한 빛을 비춘다. 태양과 같은 조명, 실시간 조명 중 가장 비용이 적음

Point Light : 일반 전구와 같은 성격을 띰, Point Light가 있는 좌표를 기준으로 360도 퍼져나가는 조명, 빛의 범위를 조절할 수 있게하는 Range도 존재 , 실시간 조명

Spot Light : 손전등과 같이 Corn 모야으로 빛을 발하는 조명 , 처리비용이 가장 비싼 조명, 빛이 뻗어 나가는 각도를 조절하는 Spot Angle도 존재

Area Light : 사각형 형태의 조명, 한쪽 면에서 빛을 발하는 조명, 실시간 조명 x, 간접 조명

하늘 표현 방식 : Skybox, SkyDome, - 6 sided, Cube map, Panoramic, Procedural

프로시저럴 스카이박스(Procedural) : 텍스처 없이 직접 꾸밀 수 있음

큐브맵 스카이박스(Cubemap) :

Material – Shader – Skybox – 6Sided

날짜 : 10.16

Page 112~196

유니티 엔진의 개발 방식 : 컴포넌트 기반의 개발(CBD : Component Based Development),

데이터 기반의 개발(DOTS : Data Oriented Technology Stack)

CBD : 소프트웨어 개방 방법론, 독립적인 기능 단위로 컴포넌트를 제작한 다음 필요한 기능을 조립하는 방식

스크립트는 프로젝트 뷰에서 스크립트명을 수정하더라도 스크립트의 클래스명이 자동으로 수정되지 않기 때문에 직접 수정해야한다. 클래스명과 스크립트 파일명이 똑같아야 한다.

Add Component : Ctrl + Shift + A

Play : Ctrl + P

콘솔 뷰 : Ctrl + Shift + C

에셋이 사용된 모든 곳을 찾는 방법 : 해당 에셋 우클릭 후 Find References In Scene

11

캐릭터의 이동 : Transform 컴포넌트 이용, 물리엔진(PhysX, Box2d) 이용

컴포넌트의 캐시 처리 : 스크립트에서 접근해야 할 컴포넌트를 Awake 함수나 Start 함수에서 미리 변수에 할당한 후에 그 변수를 통해 접근하는 것을 말함.

유니티를 실행하면 게임뷰가 활성화 뷰로 자동 선택됨.

활성화 뷰 : 입력 장치로부터 값을 받아들일 수 있는 뷰를 의미하며 게임 뷰 탭에 파란색 막대가 표시된다.

가상 카메라 : Shift + F + 원하는 오브젝트

게임뷰에서는 안따라가도 씬 뷰에서는 가상 카메라가 해당 오브젝트를 따라다님

애니메이션 : 레거시 애니메이션, 메카님 애니메이션

레거시 애니메이션 : 하위 호환성을 고려한 애니메이션, 소스 코드로 컨트롤해야 함. (Annimation)

메카님 애니메이션 : 모션 캡처 애니메이션, 리타게팅 기능 (Animator)

날짜 : 11.09

Page 198~215

총기류 발사 – Projectile, RayCasting

Projectile : 총알이 물리적으로 발사

RayCasting : 레이저 빔과 같이 눈에 보이지 않는 광선을 발사해 물체를 검출하는 방식

Rigidbody : 강체

Rigidbody.Addforce() : 월드 좌표계를 기준으로 날아감

RigidRelativeForce() : 로컬 좌표계를 기준으로 날아감

Collider : 충돌

충돌 감지 조건 :

1. 충돌을 일으키는 양쪽 게임오브젝트 모두 Collider 컴포넌트가 추가돼 있어야 한다.

2. 두 게임오브젝트 중 움직이는 쪽에는 반드시 Rigidbody 컴포넌트가 있어야 한다.

연산 속도 : Sphere Collider > Capsule Collider > Box Collider

오브젝트 간의 충돌이 발생할 경우 : OnCollision~, OnTrigger~ 이벤트 함수 발생

이벤트 함수 = 콜백 함수(충돌이 감지되면 시스템(Back)에서 호출(Call)한다는 뜻)

IsTrigger : 충돌 감지 o, 물리적 충돌 x (감지 센서 역할을 많이 함)

IsTrigger를 체크하면 OnTrigger 함수 작동, 체크 안되어 있으면 OnCollision 함수 작동

날짜 : 11.11

Page 216~271

OnCollisionEnter(Collision coll)

coll.collider.tag == “문자열” or coll.gameObject.tag == “문자열”

: 게임오브젝트의 문자 속성을 가져오는 코드(실행 시 해당 문자열의 복사본을 생성한다. 이 복사본은 특정 메모리에 할당되며 가비지 컬렉션의 대상이 된다.)

가비지 컬렉션 : 메모리에 자동으로 할당된 저장공간을 일정 시간이 지난 후 더는 사용하지 않을 때 자동으로 메모리를 해제하는 동작

가비지 컬렉션 처리해야 할 대상이 많아질수록 해제하는 시점에 잠시 멈추는 랙 현상 발생

가비지 컬렉션이 발생되지 않는 CompareTag를 쓰는 걸 권장한다.

coll.collider.CompareTag(“BULLET”)

Shift + Alt + N : 빈 게임오브젝트 생성

빈 게임 오브젝트 생성 -> 다른 게임오브젝트의 하위로 드래그해서 차일드 : 빈 게임오브젝트의 Trasform 값이 이상하게 변경

원하느 게임오브젝트에서 Create Empty Child를 하면 부모의 Transform, Layer 속성을 상속받아 생성되므로 사소한 실수 방지 가능

런 모드 상태에서 수정한 값은 에디트 모드로 돌아가면 원래 값으로 롤백되기 때문에 런 모드 상태에서 Transform 옵션 Copy -> Component를 하고 에디트 모드로 돌아가서 Trasmform 옵션 Paste -> Component Values

Transform 컴포넌트만 가진 게임오브젝트는 씬 뷰에서 직접 선택할 수 없어서 하리어키 뷰에서 직접 선택한 후 수정해야 하는 불편함이 있다. 이때 사용하는 것이 Gizmo이다.

혹은 인스펙터 뷰에서 Icon 선택하기

Trail Renderer: 메시의 동적 생성을 쉽게 처리해준다. ex) 총알 발사 궤적

Object -> Component -> Effect -> Trail Renderer

Quaternion(쿼터니언) : 유니티에서 사용되는 각도의 단위, 복소수 4차원 벡터

쿼터니언 회전 방식 : 세 개의 축을 동시에 회전

오일러 각 : 3차원 공간의 절대 좌표를 기준으로 물체가 얼마나 회전했는지 측정하는 방식

오일러 각 회전 방식 : X, Y, Z 축 차례대로 회전

오일러각에서 X,Y,Z 축 중 2개의 축이 겹쳐지면 어느 축으로도 회전하지 않고 잠기는 짐벌락 현상이 생김. 쿼터니언 사용시 짐벌락 현상 X

외부에서 임포트한 3D 모델은 반드시 Transform의 Scale 속성을 수정하지 말고, FBX Import Setting의 Scale Factor 속성을 수정한다.

Random.Range(1.0f, 10.,0f) : 1.0f ~ 10.0f 범위의 난수 발생 (max까지가 범위)

Random.Range(1, 10) : 1 ~ 9 범위의 난수 발생 (max-1까지가 범위)

여러줄 주석 처리 : Ctrl + K -> Ctrl + C

여러줄 주석 처리 해제 : Ctrl + K -> Ctrl + U]

Physics.OverLapSphere(원점, 반지름, 검출대상 레이어) : 해당 영역에 있는 collider 컴포넌트 추출

검출 대상 레이어 :

1 << 3 : 비트 연산 표기법, 3번째 레이어를 의미한다. 이진수로 표현하면 2^3 = 8

OverLapSphere 함수는 실행 시 Sphere 범위에 검출될 개수가 명확하지 않을 때만 사용해야 한다. 메모리 Garbage가 발생하기 때문. 검출될 개수가 명확하다면 Physics.OverLapSphereNonAlloc 함수 사용

Physics.OverLapSphereNonAlloc() : 결괏값을 저장할 정적 배열을 미리 선언해 사용하며 실행 중에 배열의 크기를 변경할 수 없다.

Rigidbody.AddExplosionForce(횡 폭발력, 폭발 원점, 폭발 반경, 종 폭발력) : 폭발력을 생성

날짜 : 11.12

Page 272~305

오디오 구현 필요 요소 : AudioListener 컴포넌트, AudioSource 컴포넌트

AudioListener : 씬에서 반드시 한 개만 존재해야 함. Main Camera에 기본 컴포넌트로 추가되어 있음

Audio Clip : 압축 품질, 3D 효과, 스테레오, 모노 변환 등을 할 수 임ㅆ고 하단에 있는 Preview 영역에서는 음원의 파형 및 미리듣기를 할 수 있는 음원 파일

스테레오 음원으로 음향 효과를 극대화하는 게임이 아니라면 모노 음원을 변환하는 것이 성능 면에서 유리하다.

[RequireComponent(typeof(AudioSource))] : 어떤 스크립트에서 함께 있어야만 하는 컴포넌트를 자동으로 추가하고, 실수로 의존성이 있는 컴포넌트를 삭제한 것을 방지하기 위해 RequireComponent 어트리 뷰트를 사용한다.

블로킹 : 함수 안에서 수행하는 로직이 10초 정도 시간이 걸린다고 가정하면 10초 동안 메시지 루프의 다른 로직을 실행할 수 없다는 것을 의미한다. 즉 10초 동안 멈추는 현상이 발생하는 것을 말한다.

멀티 스레드(Multi thread) : 메시지 루프의 다른 함수가 정상적으로 실행되면서 시간이 오래 걸리는 함수를 병렬로 호출하는 것

코루틴(Coroutine) : 협력(Co-) + 루틴(Routine)의 합성어, 협력 루틴, 협력 동작

코루틴 함수 안에 있는 yield 키워드를 만나면 제어 권한을 유니티의 메인메시지 루프로 “양보”하는 방식으로 점진적인 작업을 처리할 수 있다.

yield return null : 다음 프레임까지 해당 코루틴을 잠시 대기(정지)하느 동안 메인 메시지 루프로제어권을 넘겨 다른 작업을 처리하라는 의미이다.

StartCoroutine() 함수를 이용해 코루틴을 호출한다. 전달 인자는 함수의 원형(포인터)을 사용한다. 호출할 함수명을 문자열로 전달하면 가비지 컬렉션이 발생하며 문자열로 호출한 코루틴은 개별적으로 정지시킬 수 없다는 문제가 있다.

Start 함수는 다른 이벤트 함수와는 다르게 코루틴으로 실행할 수 있다. ex) IEnumerator Start

날짜 : 11.15

Page 306~341

유한 상태 머신(FSM : Finite State Machine) : 적 캐릭터가 스스로 알아서 주변 환경에 적응하거나 들어오는 반응에 적절하게 반작용하도록 구현한 것

Animator - 메카님 애니메이션 - 제너릭, 휴머노이드

Animation – 레거시 애니메이션

animator controller – 애니메이터 뷰 : Alt를 누른 상태에서 마우스 드래그, 마우스 가운데 버튼으로 이동, A키를 사용하면 전체 스테이트를 한 번에 볼 수 있도록 자동으로 설정

스테이트 : 애니메이터 뷰에 추가된 애니메이션 클립 ex) Idle 스테이트

Transition(전이) : 스테이트 간의 연결선

Window - Package Manager – AI Navigation

오브젝트 – NavMeshSurface 추가

NavMeshSurface : 지나갈 영역 설정

NavMeshAgent : 내비메시 데이터를 기반으로 목적지까지 최단 거리를 계산해 이동하는 역할을 한다. A\* PathFinding 알고리즘 사용

적색 실선 : 추적 대상까지의 최단 거리

녹색과 청색 화살표 : 몬스터의 진행 방향과 추적 대상과의 각도를 표시

GameObject.FindWithTag(“태그명”) : 하이러키 뷰에 존재하는 오브젝트 중 해당 태그로 지정된 오브젝트를 찾는다.

FindGameObjectWithTag, FindGameObjectsWithTag와 같이 Find~ 계열의 함수는 처리 속도가 느리기 때문에 Update 함수에서 사용하지 않는다. 즉 Awake, Start 함수에서 변수에 할당한 후 사용하길 권장함.

날짜 : 11.16

Page 342~389

몬스터의 상태를 지속해서 검사하는 로직에서는 코루틴 함수를 사용한다. 물론 Update()함수에서 구현하더라도 잘못된 방법이라고 할 수 없지만 매 프레임 몬스터의 상태를 검사하기보다는 적정한 주기로 상태를 검사하는 방식이 좀 더 효율적이고 깔끔한 코드를 만들 수 있다.

anim.SetBool(“IsTrace”, true); : 문자열을 전달 방식

private readonly int hashAttack = Animator.StringToHash("IsAttack");

anim.SetBool(hashAttack, true); : 해시값을 전달하는 방식

문자열로 전달할 경우 오탈자가 있으면 오류 발생, 애니메이터 컨트롤에 정의한 파라미터는 모두 해시 테이블로 관리하기 때문에 문자열로 호출할 때마다 내부의 해시 테이블을 검색하므로 속도 면에서 불리하다. 따라서 스크립트에서 접근하려는 애니메이터 뷰의 파라미터 해시 값을 미리 추출해 인자로 전달하는 방식이 속도 면에서 바람직한 방법이다.

Resources 폴더에 있는 모든 에셋은 런 타임에서 직접 불러와 사용할 수 있는 특징이 있지만, 빌드할 때 실행 파일에 모두 포함된다. 따라서 사용하지 않는 리소스가 있어서는 안 된다.

Rigidbody – IsKinematic(운동 역학) : 물리 엔진의 시뮬레이션 연산으로 이동 및 회전하는 것이 아니라 스크립트 또는 애니메이션에 의해 Transform 컴포넌트의 속성값을 변화시켜서 이동 및 회전하는 것을 말한다.

Debug.Log($"Player hp = {currHp / initHp}");

C#의 문자열 보간 : + 기호를 사용하는 것은 번잡스럽기 때문에 $ 기호를 이용하여 문자열을 보간한 방식은 표시 하고 싶은 변수를 중괄호{}를 사용해 바로 사용할 수 있어서 코드를 간결하게 작성할 수 있다.

날짜 : 12.05

page 390~446

GameObject.FindGameObjectWithTag(string tag) : 파라미터로 전달한 태그로 설정한 하나의 게임오브젝트만 반환

GameObject.FindGameObjectsWithTag(string tag) : 해당 태그로 지정된 모든 게임 오브젝트를 Array 타입으로 반환

SendMessage 함수 : 첫 번째 인작로 전달한 함수명과 동일한 함수가 해당 게임오브젝트의 스크립트에 있다면 실행하라는 명령, 여러 게임오브젝트의 함수를 호출하는 로직에 사용하면 효율적

SendMessageOptions.DonrequireReceiver : 홀출한 함수가 없더라도 함수가 없다는 메시지를 반환받지 않겠다는 옵션

델리게이트(Delegate) : 함수를 참조하는 변수를 의미, C++의 함수 포인터와 같은 의미

델리게이트 선언 : delegate flaot SumHandler(float a, float b);

델리게이트 타입의 변수 선언 : SumHandler sumHandler;

public static event 델리게이트명 이벤트명;

public static event 변수타입 변수명;

OnEnable(), OnDisable : 스크립트가 활성화 되거나 비활성화될 때 수행되는 함수

이벤트 연결(OnEnable) : (이벤트가 선언된 클래스명).(이벤트명) += (이벤트 발생 시 호출할 함수)

이벤트 해지(OnDisable): (이벤트가 선언된 클래스명).(이벤트명) -= (이벤트 발생 시 호출할 함수)

유니티 UI 종류:IMGUI(Immediate Made GUI), UI ToolKit, Unity UI(UGUI)

IMGUI : 코드를 이용해 UI를 표시하는 방법, OnGUI 함수에서 코드를 구현함

UI Toolkt : UI Asset 파일로 UI 스타일을 정의하고 디자인한다.

Unity UI : 게임오브젝트 기반의 UI로서 모든 UI 구성요소를 게임오브젝트로 관리한다. 게임오브젝트에 컴포넌트를 추가하는 형식.

Canvas 객체 : Canvas 컴포넌트를 포함하고 있는 게임오브젝트의 일종, 모든 UI 항목은 반드시 이 Canvas 객체의 하위에 위치해야 한다.

EventSystem 객체 : 시스템에서 발생하는 키보드, 조이스틱, 스크린 터치 등의 입력 정보를 Canvas 하위에 있는 UI 항목에 전달하는 역할을 한다.

Screen Space – Overlay : 씬의 가장 상위 계층에 표현되며 어떤 3d 객체에 의해서도 가려지지 않는다. Canvas 크기는 화면의 해상도에 맞춰 자동 조절

Screen Space – Camera : Overlay 옵션과 동일, UI 항목을 렌더링하는 별도의 카메라 설정 가능.

Main Camera와 별도로 UI만을 위한 카메라로 이원화할 경우 사용한다.

World Space : 씬에 있는 다른 게임오브젝트에 직접 UI 항목을 추가한다, Canvas의 물리적 위치 이동 가능.

앵커프리셋 : alt, shift의 조합으로 객체의 위치, 앵커 포인트의 위치, 피벗의 위치를 설정한다.

날짜 : 12.07

Page 447~506

버튼을 눌렀을 시 사용할 함수 자동으로 등록하는 방법

private UnityAction action;

// UnityAction을 사용한 이벤트 연결 방식

action = () => OnButtonClick(startButton.name);

startButton.onClick.AddListener(action);

// 무명 메서드를 활용한 이벤트 연결 방식

optionButton.onClick.AddListener(delegate { OnButtonClick(optionButton.name); });

// 람다식을 활용한 이벤트 연결 방식

shopButton.onClick.AddListener(() => OnButtonClick(shopButton.name));

람다식 : 알론조 처치라는 수학자가 람다 계산식에 사용하는 식, 그의 제자인 존 매카시가 프로그래밍 언어에 도입을 했고, 그것이 람다식 문법, 익명 메소드를 만들기 위해 사용한다.

무명 메서드 : 람다식으로 만든 익명의 메소드, 델리게이트를 선언하고 델리게이트 타입으로 선언한 변수 없이 바로 사용할 수 있는 문법

람다식 문법 : 델리게이트\_타입 변수명 = () => 식;

델리게이트\_타입변수명 = (매개변수1, 2, ...) => 식;

델리게이트\_타입변수명 = (매개변수1, 2, ...) => {로직\_1;, 2;, ...};

TextMesh Pro : 여러 가지 텍스트 렌더링 기법을 이용해 무한으로 확대하거나 축소해도 글자의 외곽선이 뭉개지거나 해상도가 떨어지지 않는다.

텍스트로 한글 쓰는 방법 : 폰트를 가져온 후 Window – TextMesh Pro – Font Asset Creator를 톻해 폰트를 만든다.

Custon Characters : 한글로 출력해야 할 문구가 이미 정해져 있는 경우 필요한 한글 문장만 입력해 사용할 때 유용하다

Packing method – Optimum : 인코딩 시간은 오래 걸리지만, 텍스처 공간을 최소화 해줌.

hpbar = GameObject.FindGameObjectWithTag("HP\_BAR")?.GetComponent<Image>();

(클래스 또는 구조체) ? (속성 또는 메서드) : null이 아니면 뒤에 이쓴ㄴ 구문을 실행한다. null이면 null 반환

Game Manager : 게임의 전체적인 흐름을 관리, 게임에 활용되는 전역적인 데이터와 게임 승패에 관련된 조건을 체계적으로 관리하는 역할을 한다.

프로퍼티 : 객체지향 언어의 특징인 데이터 은닉성을 유지하면서 해당 데이터를 안전하게 외부에 노출하는 방법

// 게임의 종료 여부를 저장할 멤버 변수(필드)

private bool isGameOver;

// 게임의 종료 여부를 저장할 프로퍼티

public bool IsGameOver

{

get { return isGameOver; }

set

{

isGameOver = value;

if(isGameOver)

{

CancelInvoke("CreateMonster");

}

}

private 접근 제한자로 선언한 isGameOver 변수는 외부에 노출되지 않지만 IsGameOver라는 프로퍼티를 선언해 isGameOver 변수를 간접적으로 다른 클래스에 노출한다.

private 타입 멤버\_변수;

public 타입 프로퍼티명

{

get { return 반환\_값; } // getter 영역

set

{

멤버\_변수 = value;

} // setter 영역

}

외부에서 프로퍼티를 읽을 때 실행되는 영역은 get 부분이고 값을 대입할 때 실행되는 영역은 seㅅ 부분이다. 프로퍼티에 값을 대입하면 그 값은 value 키워드를 통해 전달된다.

Invoke : 일정 시간이 지난 후에 특정 함수를 호출할 때 사용한다.

Invoke(“호출할 함수”, 대기시간);

CancelInvoke : InvokeRepeating 함수로 인해 계속 반복하는 프로세스를 정지할 때 사용하는 함수

게임오브젝트 접근 후 해당 스크립트의 변수 값 변경 :

GameObject.Find("GameMgr").GetComponent<GameManager>().IsGameOver = true;

인스펙터창에서 Debug 모드 : 해당 스크립트의 private로 된 변수의 값을 전부 볼 수 있다.

날짜 : 12.10

Page 507~

싱글턴 패턴 : 메모리상에 오직 하나의 인스턴스만 생성하고, 그 인스턴스에 전역적인 접근을 제공하는 소프트웨어 디자인 패턴 중 하나이다.

GameManager 전역적 접근을 위해 static으로 선언 후 메모리에 상주시킨다.

외부 접근이 가능하도록 public으로 선언

instance : GameManager 클래스를 저장한 변수

DontDestroyOnLoad : 다른 씬으로 넘어가더라도 게임오브젝트를 계속 유지함.

오브젝트 풀링(Object Pooling) : 사용할 객체를 미리 만들어 놓은 후 필요할 때 가져다 사용하는 방식

GameObject.tranform.SetPositionAndRotation(position, rotation); : 위치와 회전값을 동시에 설정

\_monster?.SetActive(true); : null이 반환되면 뒤에 구문은 동작 x

교재 내 오류 : 몬스터가 플레이어의 총알에 의해 죽고 나서 몬스터의 상태가 Die로 고정되어 있어 몬스터의 SetActive가 falsel, true가 반복됨

해결 방법 : 몬스터가 Die 상태에서 CapsuleCollider를 true로 바꿀 때 같이 몬스터의 상태를 Ide로 바꿔주면 된다.

날짜 : 12.11

Page : 532~554

scoreText.text = $"<color=#00ff00>SCORE :</color> <color=#ff0000>{totScore:#,##0}</color>";

{12345:#,##0} -> 12,345

{123:0,000} -> 0,123

#,##0 : 3자리 숫자마다 콤마를 추가한다는 뜻

# : 해당 자릿수에 미달하면 숫자를 표시하지 않느다.

MonsterAction 코루틴에서 yield return NewWaitForSeconds(0.3f) 때문에 몬스터를 죽이고 나서 총알을 빠르게 연사하면 몬스터가 3번 죽는 오류가 있음. 시간을 수정하지 않는 이상 계속 지속될 듯.

PlayerPrefs : int, flaot, string,bool 타입의 변수를 저장하고 로드

PlayerPrefs.SetInt(“키이름”, 데이터);

PlayerPrefs는 보안성이 없어서 중요한 게임데이터는 절대로 PlayerPrefs로 관리해서는 안 된다.

PlayerPrefs를 사용하더라도 암호화해서 사용해야 함.

Projecttile : 발사체가 시각적으로 보이고 물리적인 충돌이 일어나게끔 하는 것.

Raycast : 눈에 보이지 않는 광선을 쏘아서 해당 광선에 맞은 물체가 적인지 여부를 판단한 뒤에 여러가지 후처리를 하는 방식

Raycast가 어떤 객체를 검출하기 위해서는 그 객체가 하나 이상의 Collider컴포넌트를 갖고 있어야 한다.

Physics.Raycast(firePos.position, // 광선의 발사 원점

firePos.forward, // 광선의 발사 방향

out hit, // 광선에 맞은 결과 데이터

10.0f, // 광선의 거리

1 << 6) // 감지하는 범위인 레이어 마스크

Physics.Raycast() : ;Ray를 투사해 조건에 맞는 객체에 닿으면 true값을 반환

AI-Navigation(Obsolete)-Object에서 각 물체마다 걸어다닐 수 있는지 못지나가는 물체인지 static으로 변경하고 Bake하여 이동 경로를 설정할 수 있다.

날짜 : 12.12

Page 556~588

Off Mesh Link Generation: 분리된 메시를 연결할 수 있다. ex) 계단에서 뛰어내리게끔 함.

Area Mask : NavMeshAgent에서 최단 거리를 계산할 때 경로의 가중치를 고려하게 함.

평평한 길을 진흙 길보다 가중치를 낮게 판단하여 평평한 길로 가게 함.

전역 조명 : 3D 지오메트리에서 직접 광원과 다른 물체에 반사된 간접 광원 및 그림자를 렌더링해 좀 더 사실적인 조명을 구현하는 것을 말한다.

조명 모드 – Realtime, Mixed, Baked

RealTime : 실시간 조명, 씬에 직접 광을 적용하여 매 프레임 조명 연산을 해서 화면 업데이트

Mixed : Lighting Mode의 속성에 따라 달라짐

Lighting Mode – Baked Indirect, Subtractive, Shadowmask

Baked : 런타임 시 조명 연산 처리를 하지 않는 조명, 정적인 물체에 전역조명과 그림자를 생성한다.

동적인 물체의 조명 효과는 라이트 프로브를 통해 적용

라이트 매핑 : 씬에 배치된 모든 3D 모델에 영향을 미치는 직접 조명, 간접 조명 및 그림자의 효과를 텍스처로 미리 만드는 과정을 말한다.

라이트맵 : 미리 만든 결과로 만들어진 텍스처 파일

Light Probe : 8개의 원 기즈모가 안보여서 작업 못함. 왜 안보이는지 모르겠음.

날짜 : 12.13

Page 600~653

SceneManager : 동적으로 씬을 생성, 해제하거나 기존에 ㅁ나들어진 씬을 호출하는 여러 메서드를 제공한다.

SceneManager.LoadScene() : 새로운 씬을 호출하는 함수로서 A 씬을 먼저 로드한 후 B씬을 추가해 로드한다.

// 목적지까지 남은 거리로 회전 여부 판단

if( agent.remainingDistance >= 2.0f)

{

// 에이전트의 이동 방향

Vector3 direction = agent.desiredVelocity;

if (direction != Vector3.zero) //look rotation viewing vector is zero오류가 계속 떠서 내가 추가한 코드

{

// 회전 각도(쿼터니언) 산출

Quaternion rot = Quaternion.LookRotation(direction);

// 구면 선형보간 함수로 부드러운 회전 처리

monsterTr.rotation = Quaternion.Slerp(monsterTr.rotation,

rot,

Time.deltaTime \* 10.0f);

}

}

Light Probe : 기즈모가 꺼져있어서 안보였음. 못했던 거 완료함.

프러스텀 컬링 : 카메라의 시야 범위에 들어와 있는 물체만을 렌더링하고 시야 범위 밖의 물체는 렌더링하지 않는다.

오클루전 컬링(Occlusion Culling) : 렌더링 부하를 줄여주는 기법 중 하나로 3D 게임 및 콘텐츠 개발에 필수적인 요소이다. 카메라 시야에서 다른 물체에 가려 보이지 않는 물체를 렌더링하지 않는 기법을 말한다.

Occluder Static : 가리는 물체 (벽 역할)

Occludee Static : 가려지는 물체

Occlusion Culling 해봤는데 뒤에 물체가 안없어져서 실패함.

Input System : 여러 기기(콘솔, 모바일, PC 등등)에서 사용하는 걸 편리하게 해줌.

Window – Package Manager – Input System 설치

ProjectSettings – Other Settings – Active Input Handling – Input System Package

Assets – Create – Input Action

Control Schemes : 사용할 플랫폼들

Action Maps : 캐릭터들의 이동, 공격과 같은 행동들을 구현

Actions : 행동들

Binding : 어떤 것을 입력받아 행동할 것인지

Path : 입력받을 키

Player Input 컴포넌트를 해당 캐릭터에 추가하고, 만들어둔 Action Maps를 연결.

Behavior : Input Actions 에셋에 정의한 액션이 발생했을 때 코드이 함수를 어떻게 실행 시킬 것인지를 결정하는 속성

Behavior – Send Message, Brodcast Message, Invoke Unity Events, Invoke C Sharp Events

Send Message : 게임오브젝트의 함수를 이용하여 호출

Brodcast Message : 하위 오브젝트의 함수도 호출

Invoke Unity Events : 기존 UI에서 하듯이 게임 오브젝트 연결 후 함수 할당

Invoke C Sharp Events : 모든 것들을 코드로 할 수 도 있고, Player Input 컴포넌트를 사용할 수도 있다.

날짜 : 12.21

Page : 654~

Direct Binding : PlayerInput 컴포넌트 사용 or 직접 스크립트에서 전부 작성

private InputAction moveAction;

// Move 액션 생성 및 타입 설정

moveAction = new InputAction("Move", InputActionType.Value);

// Move 액션의 복합 바인딩 정보 정의

moveAction.AddCompositeBinding("2DVector")

.With("Up", "<Keyboard>/w")

.With("Down", "<Keyboard>/s")

.With("Left", "<Keyboard>/a")

.With("Right", "<Keyboard>/d");

// Move 액션의 performed, canceled 이벤트 연결

moveAction.performed += ctx =>

{

Vector2 dir = ctx.ReadValue<Vector2>();

moveDir = new Vector3(dir.x, 0, dir.y);

// Warrior\_Run 애니메이션 실행

anim.SetFloat("Movement", dir.magnitude);

};

moveAction.canceled += ctx =>

{

moveDir = Vector3.zero;

anim.SetFloat("Movement", 0.0f);

};

// Move 액션의 활성화

moveAction.Enable();

네트워크 게임 : 물리적인 서버(하드웨어) + 네트워크 게임 엔진(소프트웨어)

포톤 리얼타임 : 다양한 개발 플랫폼에 맞춰진 포톤의 핵심 코어 엔진을 의미

포톤 클라우드 : 이 엔진을 바탕으로 클라우드 서비스를 제공하는 것으로 서버의 보안, 로드밸런싱, 백업 및 네트워크 트래픽 관리를 자동을 처리한다. 소프트웨어를 임대해 사용하는 방식

포톤 서버 : 물리적인 서버를 직접 운영하는 것을 의미

Window – Package Manager – Universal RP install

Assets – Create – Rendering – Universal Render Pipeline – Pipeline Asset(Forward Renderer)

Edit – Preject Settings – Graphics – Scriptable Render Pipeline Settings에 앞서 생성한 Asset 추가

Quatlity – Rendering에 앞서 생성한 Asset 추가

포스트 프로세싱 : 렌더링된 결과물(영상, 이미지)에 대한 후처리 작업

비네팅(Vignette) 효과 : 화면의 주변이나 모서리를 어둡게 처리하는 것으로 자연스럽게 화면의 가운데 부분으로 시선을 모으는 효과가 있다.

Main Camera에도 Rendering – Post Precessing을 설정해놔야 게임뷰에서도 적용할 수 있다.

톤 매핑 : 조명의 밝기를 HDR(High Dynamic Range)에서 사람이 인식할 수 있는 범위로 조정하는 효과를 말한다.

블룸 효과 : 광원 주위로 과장되게 표현하는 광학 효과로 뭉환적인 느낌을 주는 효과다.

시네머신 : Window – Package Manager – Cinemachine 설치

GameObject – Cinemachine – Virtual Camera

Follow, Look At에 Player를 연결하여 카메라가 플레이어를 보고 따라가도록 한다.

Pun 2 유니티스토어에서 설치 – 포톤 사이트에서 어플리케이션 ID 복사 후 입력 – setup project

MonoBehaviourPunCallbacks 클래스로 베이스 클래스를 써야 PUN의 다양한 콜백 함수를 오버라이드해서 작성해야 하며 비주얼 스튜디오나 VSCode를 이용한다면 인텔리센스 기능을 통해 소스 코드를 자동완성할 수 있어서 편리하다.