<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~

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