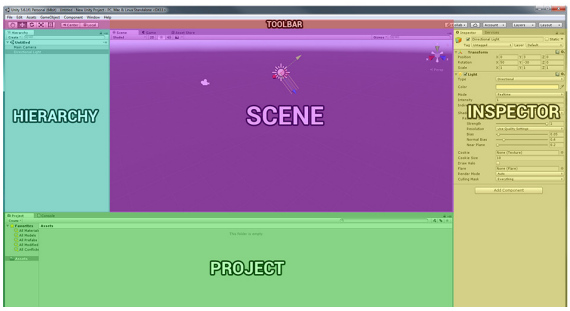
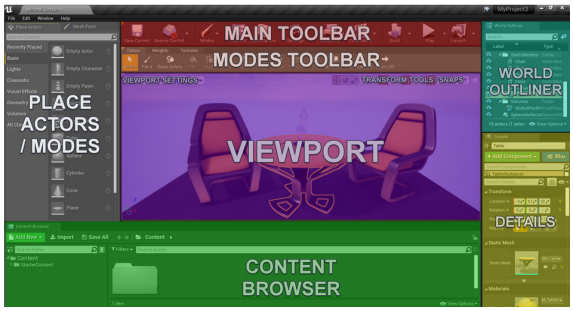
Unity에서 Unreal Engine 4로 가기 위한 지식

▼ 에디터

UNITY EDITOR



UNREAL EDITOR



- 레이아웃 드래그 앤 드롭으로 조정 가능

▼ 에셋 편집

- 유니티 → Inspector 탭에서 에셋 편집
- 언리얼 → 디테일 패널에 뜨기는 하지만 규모가 큰 편집 작업은 전용 창이나 탭에서 이루어짐

▼ 프로젝트와 파일

- 가장 중요한 서브폴더는 Content와 Source
- Asset 폴더와 비슷하게 Content 폴더 존재
  - 에셋을 import하려면 파일을 Content 디렉터리에 넣으면 자동으로 import

    - 외부 프로그램으로 파일을 변경할 때마다 자동으로 에디터 업데이트
- 유니티는 Scene 언리얼은 Map
- 프로젝트 세팅 → 메인 메뉴의 편집 / 프로젝트 세팅
    - 플레이어 세팅은 플랫폼 카테고리 밑의 플랫폼 세팅
- 블루프린트 전용 프로젝트의 경우 Source 폴더 없다.
    - 프로젝트에 코드 추가 사용
- Object는 Actor
    - 기본 Actor 이외에 특수 Actor 존재
        - Pawn → 플레이어나 AI Object
        - Character → 애니메이션 캐릭터에 사용
- 컴포넌트는 비슷한데 컴포넌트끼리 서로 붙는 계층구조가 저장됨
- 액터 만든 후 블루프린트 생성으로 프리팹 만들 수 있음
- 스크립트 만들면 Start와 Update와 비슷한 역할을 하는 InitializeComponent와 TickComponent 만들어짐

▼ 간략 용어집

카테고리	유니티	UE4
게임플레이 유형	Component	Component
	GameObject	Actor, Pawn
	Prefab	Blueprint Class

카테고리	유니티	UE4
에디터 UI	Hierachy Panel	World Outliner
	Inspector	Details Panel
	Project Browser	Content Browser
	Scene View	Viewport
Mesh	Mesh	Static Mesh
	Skinned Mesh	Skeletal Mesh
Material	Shader	Material
	Material	Material Instance
Effect	Particle Effect	Effect, Particle, Cascade
	Shuriken	Cascade
Game UI	UI	UMG(Unreal Motion Graphics)
Animation	Animation	Skeletal Animation System
	Mecanim	Persona, Animation Blueprint
2D	Sprite Editor	Paper2D
Programming	C#	C++
	Script	Blueprint
Physics	Raycast	Line Trace, Shape Trace
	Rigid Body	Collision, Physics
Runtime Platform	iOS, Web	Platforms

▼ 코드 작성

- 프로젝트 파일 수동 새로고침 필요할 때가 있음
  - UE4 새로운 버전 다운 후나 디스크의 소스 파일을 수동으로 변경한 후
  - 메인 메뉴의 “Refresh Visual Studio Project” 또는 프로젝트 디렉터리의 .uproject 파일에 우클릭 후 “Generate Visual Studio project files” 선택
- 나머지는 직접 사용하며 공부

▼ 잦은 질문

- 전 프로젝트 자동 로드 방법
  - “Always load last project on Startup” 옵션

- 입력 바인딩
  - 프로젝트 세팅→ 입력 카테고리 설정
- 프로젝트 시작 씬 바꾸는 법
  - 메인 메뉴 → Edit/Project Settings → Maps & Modes에서 스타트업 맵 바꾸기
- 게임 실행
  - 툴바의 플레이 버튼
  - 독립형 어플리케이션 실행은 플레이 옆의 드롭다운에서 “독립형 게임”
- 단위
  - 유니티는 1유닛 1미터
  - 언리얼은 1유닛 1센치
- 로그 출력 확인
  - Output Log 출력 창은 Window → Developer Tools 창 → 개발자 툴 에서 열 수 있다
  - 게임에 -log 명령줄 파라미터를 붙여 실행하면 게임 옆에 전용 로그 창 뜨는데 정말 유용하단다
- 예외 처리
  - 응 없어
  - 대신 chech() 함수로 치명적인 assert 오류 발동 가능
  - 오류 보고는 하지만 프로그램 안 멈추고 싶으면 ensure()
  -

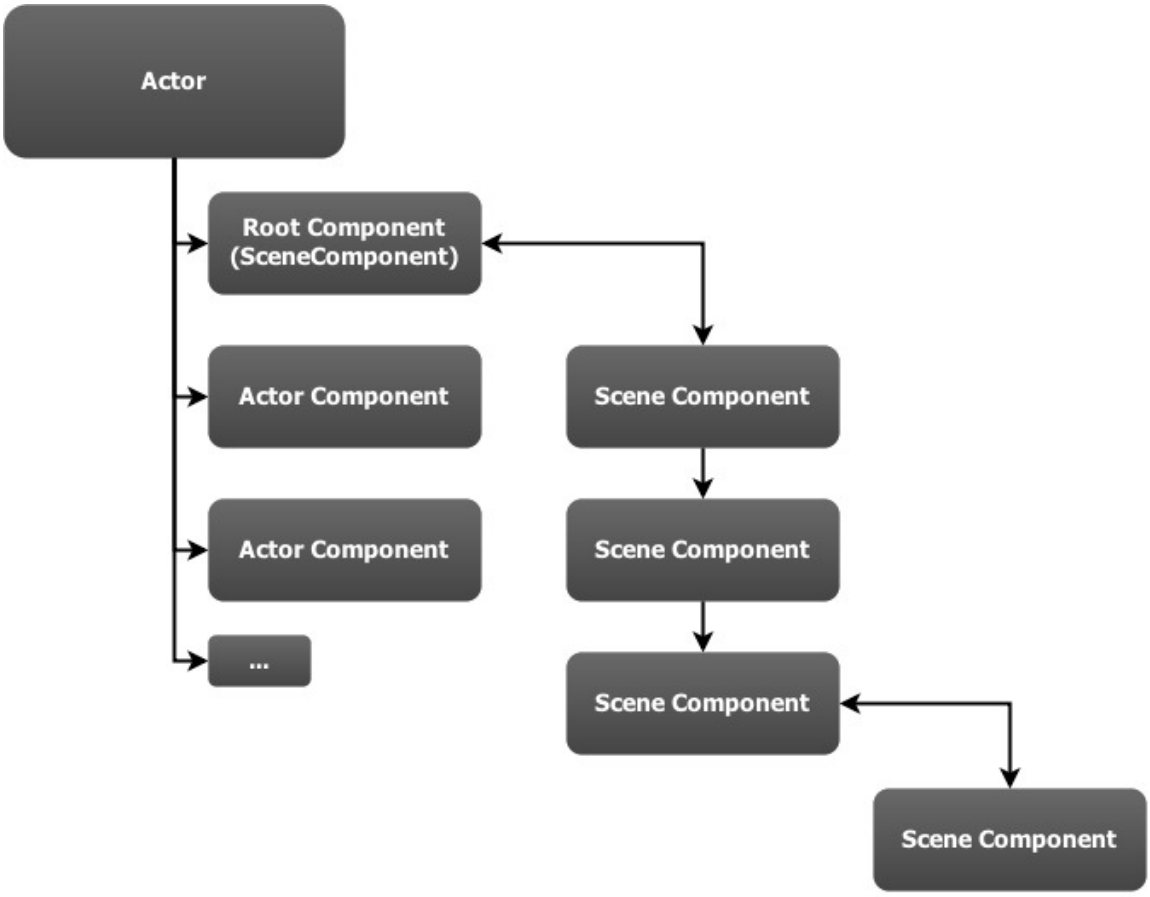
▼ Actor

- 흔히 쓰이는 Actors
  - Pawn
    - 제어가능한 게임 오브젝트
    - 전형적으로 플레이어의 아바타를 나타내는 액터 유형
    - Controller 소유를 통해 플레이어나 AI 등에 의해 움직임
  - Charcter

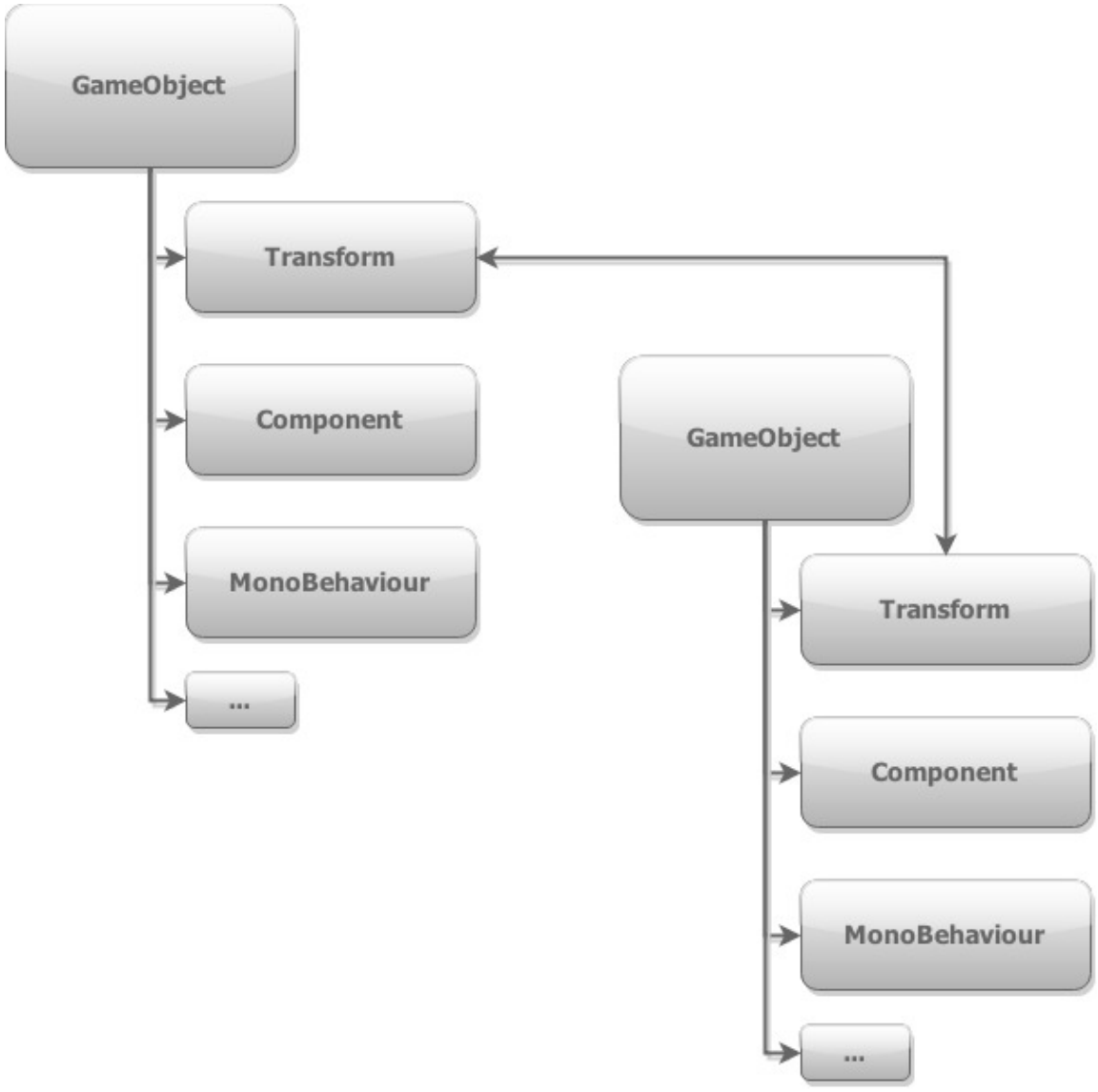
        - 이족보행 아바타용으로 디자인된 보다 전문화된 버전의 폰
        - 그러한 게임 오브젝트 유형의 복잡도를 다수 처리한다 (?)
    - Controller
        - 폰을 Possess 빙의하여 제어
        - 폰과 컨트롤러를 분리함으로써 플레이어를 가정하고 그와 똑같은 인터페이스를 사용하여 폰을 임의로 조작하는 AI 컨트롤러 작성 가능
    - Player Controller
        - 플레이어의 게임패드, 터치, 마우스/키보드에서 입력을 받음
        - 그 입력을 사용하여 빙의중인 폰또는 캐릭터를 구동시키기 위한 좀 더 전문화된 컨트롤러
- Object 클래스
    - World에 스폰 가능한 유일한 유형은 Actor
    - 즉 레벨에 배치하는 모든 것은 액터
    - 하지만 액터 외 다수를 포함해서 모든 언리얼 클래스의 기본 클래스는 Object

▼ 복합 오브젝트 계층구조

- 언리얼


Actor
Root Component
(SceneComponent)
Actor Component
Actor Component
...
Scene Component
Scene Component
Scene Component
Scene Component

- 유니티



▼ 블루프린트 클래스

- 블루프린트 클래스는 확장 가능
  - 언리얼은 기본적인 괴물 클래스 만들고 상속하여 드래곤을 만든다던지 귀신을 만든다던지 가능
  - 유니티는 프리팹 중첩 시키면 복잡해져서 힘듦
- 블루프린트, C++
  - 블루프린트만 가지고도 소규모 게임 만들기 가능
  - 대부분의 프로젝트는 블루프린트와 C++를 혼합하여 사용
  -