

게임 엔진 프로그래밍

제3강 게임 플레이 시스템



학습 내용

- 게임 월드의 구성
- 동적 구성 요소 구현: 게임 객체
- 데이터 주도 게임 엔진
- 게임 월드 에디터

게임 플레이 시스템

■ 게임 메카닉(Game Mechanics)

- 게임 내에 있는 여러 단위 간의 상호 작용을 규정하는 규칙들
 - 플레이어의 목표
 - 성공과 실패의 기준
 - 플레이어 캐릭터의 능력
 - 가상 월드에 존재하는 NPC 개체들의 수와 종류
 - 전체적인 게임을 통해 겪는 체험의 흐름
-
- 게임 메카닉은 일반적으로 설득력있는 이야기와 생동감 있는 캐릭터들과 밀접하게 섞임.
 - 물론, 이야기와 캐릭터가 필수적인 것은 아님. (예: 테트리스)

월드 구성 요소

■ 게임 월드

□ 게임의 배경이 되는 2D 또는 3D 의 가상 게임 세계

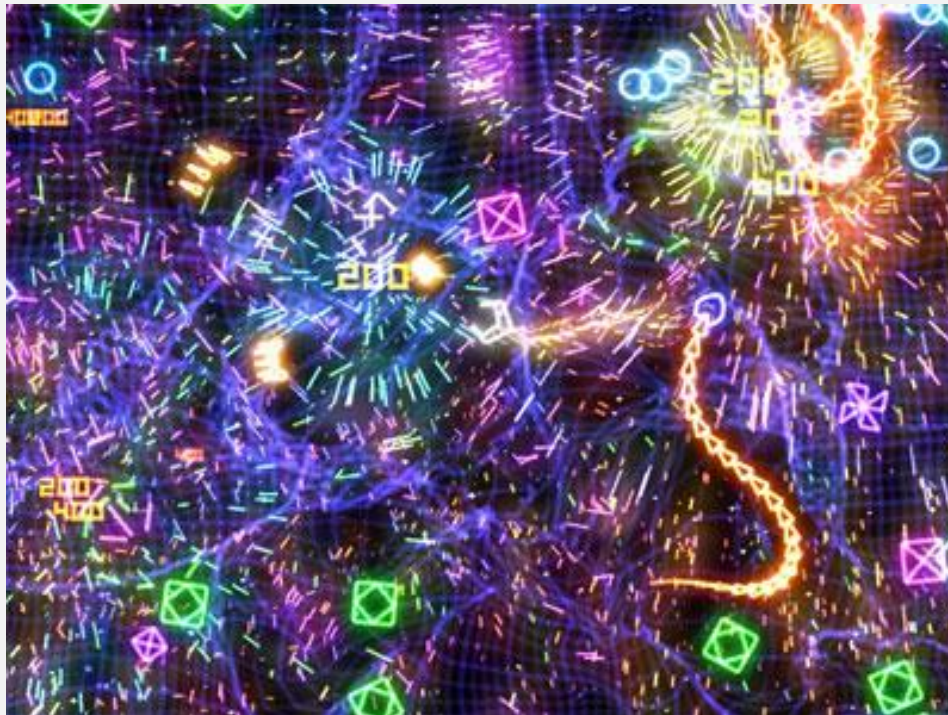
□ “정적 구성 요소”

- 움직이지 않거나, 게임 플레이에 적극적으로 관여하지 않는 요소들
- 지형, 빌딩, 길, 다리, 등등

□ “동적 구성 요소”

- 움직이이면서 게임의 진행에 직접적으로 개입하는 요소들
- 캐릭터, 탈것, 무기, 아이템, 수집할 수 있는 물건
- 파티클 이미터(Particle Emitter)
- 동적 광원(Dynamic Light)
- 영역(Region) - 중요한 게임이벤트를 검출하는, 눈에 보이지 않는 공간
- 스플라인(Spline) - 물체의 경로를 정의
- “게임 상태(Game State)” - 게임 월드의 동적 구성 요소들 전체의 상태를 포괄적으로 지칭

- 동적 구성 요소와 정적 구성 요소의 비율은 게임마다 다름.
- 일반적으로, 적은 수의 동적 구성 요소들이 넓은 정적 배경 공간 위에서 움직임.
 - 동적 구성 요소는 CPU 자원 측면에서 정적 요소보다 비쌘.
 - 물론, 동적 구성 요소가 많아야 게임 월드가 “활기차” 보임.
 - 정적 구성 요소가 없는 것도 있음. (예. Asteroids, Geometry War)



- 동적 요소과 정적 요소의 구별은 항상 명확하진 않음.

- 게임 엔진 마다 기준이 다르며, 아예 구분하지 않고 모든 요소를 동적으로 간주하는 엔진도 존재.



■ 동적 요소과 정적 요소의 구별은 “최적화” 관점에서 보는 것이 타당.

□ 어떤 물체의 상태가 변하지 않는다면?

- 미리 계산함으로써 런타임 시간 계산을 없앨 수 있음.
- 정적 배경 물체들 - 월드 공간으로 미리 배치 가능
- 라이트맵(Light Map)
- 그림자맵(Shadow Map)
- 정적 환경 차폐 정보(Static Ambient Occlusion Information)
- PRT(Precomputed Radiance Transfer) 구면조화 함수 계수(Spherical Harmonics coefficient)

□ 파괴가능한 배경의 구현?

- 세가지 다른 버전(멀쩡한 버전, 손상 입은 버전, 완전히 파괴된 버전)을 미리 준비
- 게임 진행에 따라 교체

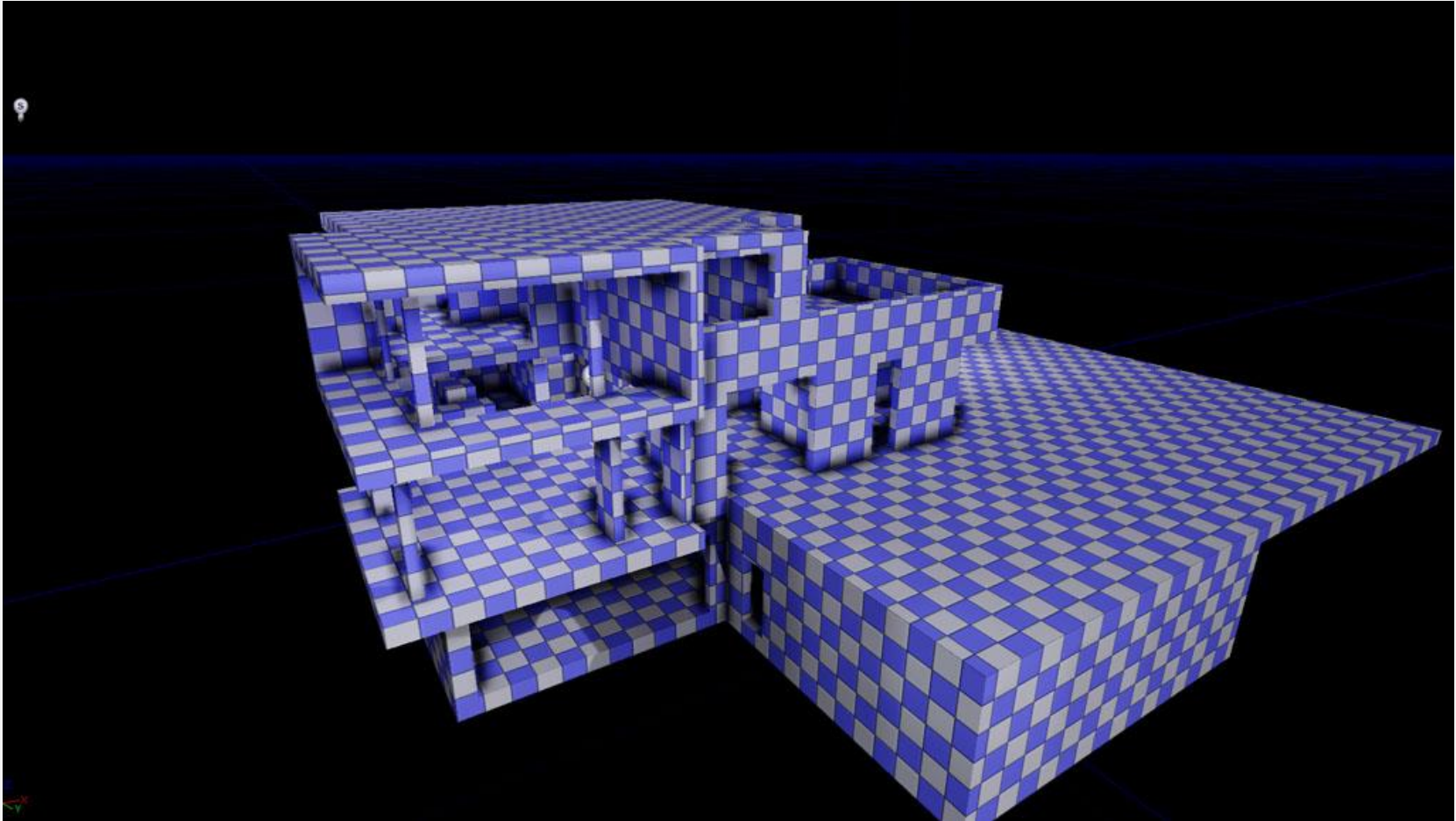
정적 지형(Static Geometry)

- 풀, 나무, 빌딩 등, 정적 배경으로 사용되는 물체들을 효과적으로 모델링 및 렌더링
- Geometry Instancing
 - 한 씬내에서 동일한 메쉬의 복사본을 여러 개를 렌더링하는 것.
 - 각 인스턴스의 위치와 방향만 살짝 다르게 줌으로써, 다양한 물체들이 있는 것 같은 느낌을 줌.
 - 그래픽 카드의 가속 기능(Hardware Instancing)을 통해 빠른 렌더링이 가능.



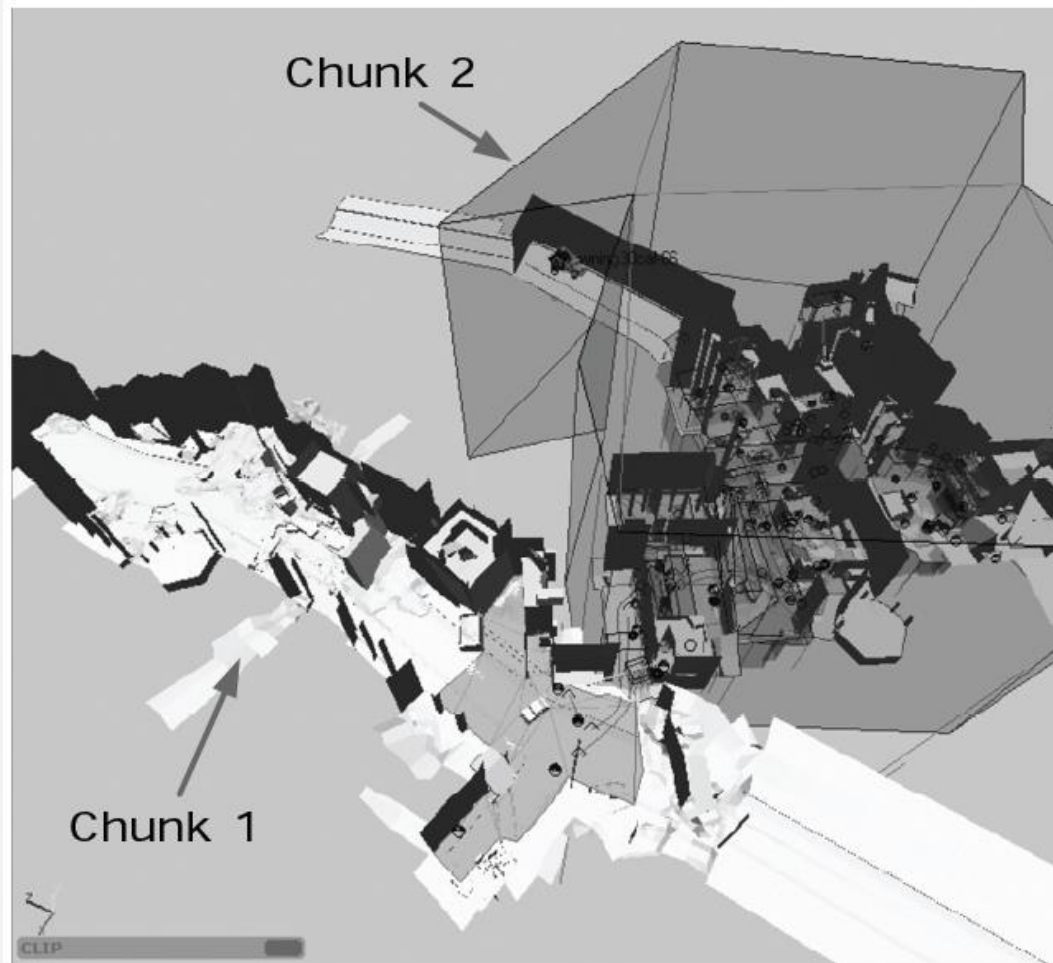
Brush Geometry

- 평면들로 둘러싸인 볼록한 입체들이 모여서 이루어진 형상
- 빠르고 만들기 쉽고, BSP 렌더링과 잘 어울림.
- 배경의 충돌 데이터를 만드는데 많이 사용됨.



월드 덩어리(World Chunk)

- 게임의 배경이 매우 넓은 경우, 여러 개의 지역으로 쪼개는 것이 일반적.
- World Chunk = 레벨 = 맵 = 스테이지 = 에어리어
- 게임의 흐름에 따라 청크 사이를 플레이어가 이동해 나가게 됨.



■ 레벨

- 초창기 게임기들의 한정된 메모리 제약 조건 내에서 다양한 게임 플레이를 제공하는 목적으로 만들어진 것.
- 최근의 게임들에서는 월드 청크가 바뀌는 것을 유저가 체감하기 어려워짐.

■ 다양한 청크 구조

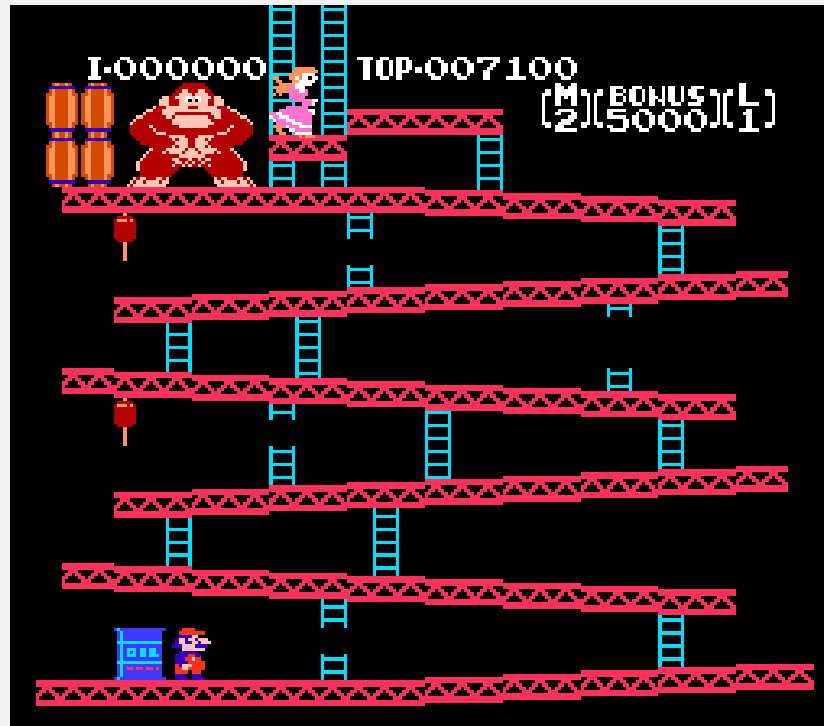
- 별형 구조(Star Topology) - 플레이어가 시작하는 허브 지역을 중앙에 두고, 이 허브에서 다른 지역들에 마음대로 접근하는 형태
- 그래프형 구조 - 각 지역이 일정하지 않은 방식으로 연결

■ 청크의 역할

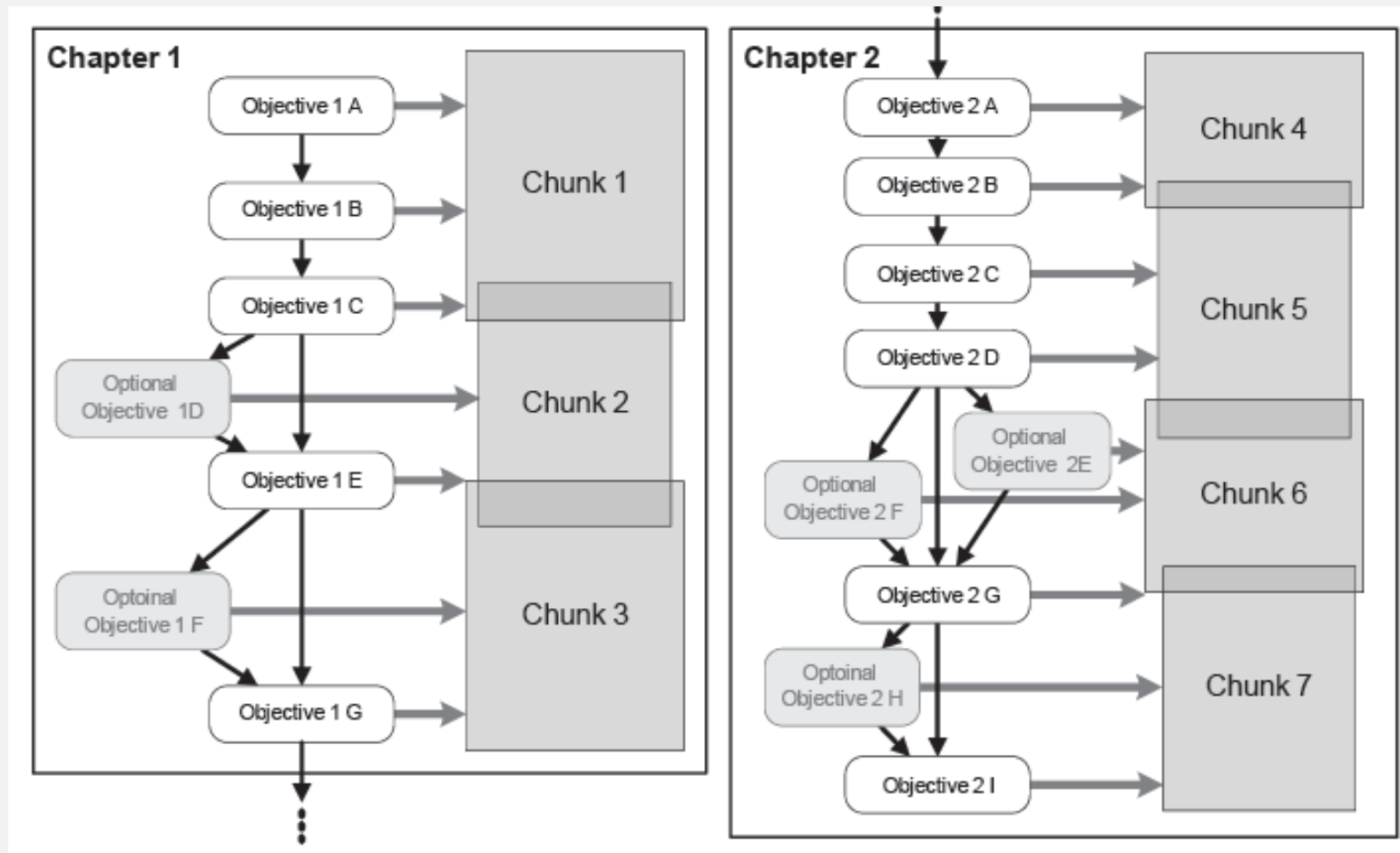
- 제한된 메모리의 활용
- 게임 전체 흐름을 조정하는 편리한 방식
- 분업에도 유리

고차원 게임 진행(High-Level Game Flow)

- 플레이어의 목표를 연속적인 배열이나, 트리, 그래프 등의 형태로 정의하는 것
- 목표 - 임무, 스테이지, 레벨, 웨이브 ...
- 각 목표의 성공 조건과 실패하는 경우의 벌칙을 지정
- 초기의 게임들은 월드 청크마다 하나의 목표를 지정
 - 새로운 레벨에는 새로운 목표



■ 오늘날의 게임들은 여러 개의 목표와 청크가 상호 연관



동적 요소 구현: 게임 객체

■ 게임 객체(Game Object)

- 게임 월드의 거의 모든 동적 구조 요소를 지칭
- Entity, Actor, Agent
- 게임 객체의 본질: 속성과 행동의 모음
 - 속성 - 게임 객체의 현재 상태
 - 행동 - 시간에 따라, 혹은 이벤트에 반응해서 상태가 변하는 방식
- 타입(Type)
 - 게임 객체는 타입에 의해 구분
 - 타입이 다른 객체는 다른 속성 스키마와 다른 행동을 가짐
 - 인스턴스 - 한 타입의 인스턴스는 속성 스키마와 행동은 같지만 속성들의 값이 달라짐.
- 팩맨의 객체 타입
 - 유령, 알약, 파워 쿠키, 팩맨

게임 객체 모델

- 가상의 월드에 존재하는 동적인 존재들을 모델링하고 시뮬레이션할 수 있게 게임 엔진이 지원하는 기능들
- 어떤 게임을 구성하는 구체적인 존재들을 시뮬레이션할 때 발생하는 문제들을 해결하는데 쓰이는 특정한 객체지향 프로그래밍 인터페이스
 - Ogre3d 엔진의 Entity 클래스와 API
- 게임 엔진을 만든 프로그래밍 언어를 확장하는 개념으로도 볼 수 있음.
 - 파이썬, 루아 등의 스크립트 언어로 제어할 수 있는 게임 객체 인터페이스

툴 측면의 디자인과 런타임 구현 측면의 디자인의 차이

■ 툴 측면의 객체 모델

- 디자이너가 월드 에디터에서 보는 게임 객체 타입들의 집합

■ 런타임 구현상의 객체 모델

- 툴 측면의 객체 모델을 런타임에 구현하는데 사용한 프로그래밍 언어의 기능과 소프트웨어 시스템의 정의됨.

■ 월드 에디터를 통한 객체 모델과 런타임 구현에 쓰이는 객체 모델은 일반적으로 일치하지만, 반드시 일치할 필요는 없음.

- 툴 상에서의 게임 객체 하나가 게임 실행 관점에서는 여러 클래스의 모음으로 구현될 수 있음.

데이터 주도 게임 엔진

- 초창기의 게임 개발은 대부분 하드코딩.
- 콘텐츠(리소스)의 양이 빠르게 증가함에 따라, 개발자들의 역량이 제작의 병목이 됨.
- 이제, 콘텐츠를 제작하는 사람들이 콘텐츠를 생성하고 컨트롤 할 수 있는 능력을 갖는 것이 중요해짐.
- Data-Driven Game Engine
 - 게임의 행동이 프로그래머가 제작된 SW에 의해서만 조정되지 않고, 아티스트와 디자이너가 데이터를 이용해서 조정할 수 있는 게임 엔진
 - 월드 에디터와 같은 툴이 필요
 - 다양한 입력 범위에 대한 안정적인 처리가 필요

게임 월드 에디터

- 게임 월드 덩어리를 정의하고, 그 안에 동적, 정적 구성 요소들을 채워 넣는 일
- 게임 객체들의 초기 상태값 지정
- 동적 구성 요소들의 행동을 조정
 - 스크립트 언어의 사용 또는 각종 파라미터의 조정

게임 에디터의 기능

■ 월드 덩어리 생성과 관리

- 생성, 결합, 분리, 파괴
- AI 네비게이션 지도, 플레이어가 짚을 수 있는 손잡이 정보, 엄폐 지점 정의

■ 게임 월드 시각화

- 완성된 게임 월드 콘텐츠를 효과적으로 뷰잉

■ 탐색

- 월드 안의 자유로운 탐색

■ 선택

- 구성 요소들의 선택 - 레이캐스팅 등을 이용

■ 계층

- 구성요소들의 계층화, 계층단위 편집, 시각화, ..

■ 속성그리드

- 객체들의 속성의 조정

■ 물체위치시키기와 정렬

■ 특수 객체 타입

- 조명, 파티클, 영역, 스플라인

■ 월드 덩어리의 저장 및 로딩

통합된 자원 관리툴

■ UnrealEd(언리얼 에디터)

- 게임 엔진과 직접 통한 - 에디터 변경 내용이 실행 중인 게임의 동적 구성 요소에 직접 적용
- 빠른 반복 개발이 가능
- 게임 자원 데이터베이스 관리 기능 통합
- 전체 자원에 대한 실시간 WYSIWYG 보기 지원

■ 데이터 처리 비용(Data Processing Costs)

- 대부분의 게임 리소스는 원본 소스를 제작한 후, 엔진에 최적화된 형식으로 변경하는 과정을 거침(Export)
- 월드 에디터에서 사용하는 자원이 원본일 수도 있고, export 된 형식일 수도 있음.
- UnrealEd: 에디터로 원본 리소스를 로딩할 때, 포맷 변경이 됨. 빠른 반복 생산은 가능하나, 월드 에디터 상에서 원본 자원을 수정하기는 어려움.
- 소스 엔진, 퀘이크 엔진: 최종적으로 게임을 빌드할 때, 리소스최적화가 수행.