# 1. 유니티 엔진의 기초 원리(2, 3 주차)

유니티는 객체 지향 프로그래밍의 **컴포넌트 패턴**을 핵심으로 사용합니다.

### 1. 클래스와 오브젝트 (3 주차)

- **객체 지향의 핵심:** 사람이 현실 세상을 보는 방식에 가깝게 프로그램을 완성하는 것입니다.
- **클래스(Class):** 묘사할(추상화) 대상과 관련된 코드(변수와 메서드 등)를 묶는 틀입니다.
- **오브젝트(Object):** 클래스라는 설계도와 달리 실제로 존재하는 물건(실체)입니다.
- **인스턴스화(Instantiation):** 클래스라는 틀로 오브젝트를 찍어내 실체화하는 것입니다. 생성된 오브젝트를 인스턴스(Instance)라고 합니다.
- **오브젝트의 독립성:** 원본 클래스에서 여러 개의 오브젝트를 생성하더라도, 오브젝트는 서로 독립적이며 구별 가능한 실체입니다.

### 2. 컴포넌트 패턴 (Component Pattern) (2, 3 주차)

- **문제점:** 전통적인 상속(Inheritance)은 코드를 재사용하는데 한계가 있으며, 부모 클래스의 기존 기능이 자식 클래스의 기능과 충돌할 수 있습니다.
- **컴포넌트 패턴:** 유니티가 채택한 방식으로, 미리 만들어진 부품(컴포넌트)을 조립하여 완성된 오브젝트를 만드는 방식입니다.
    - **게임 오브젝트(Game Object):** 속이 빈 껍데기(Holder) 역할로, 컴포넌트를 붙이는 뼈대입니다.
    - **컴포넌트(Component):** 스스로 동작하는 **독립적인 부품**입니다.
- **장점:** 유연한 재사용, 기획자의 프로그래머 의존도 감소, 기능 추가/삭제가 쉽습니다.

### 3. MonoBehaviour 와 메시지 기반 방식 (2 주차)

- **MonoBehaviour 클래스:** 유니티의 모든 컴포넌트가 상속하는 클래스로, 컴포넌트에 필요한 기본 기능을 제공하며 유니티의 제어를 받습니다.
- **메시지 기반 형식:** 컴포넌트들은 서로 독립적이어서, 특정 기능을 실행시키기 위해 직접 찾아가는 대신 **브로드 캐스팅(Broadcasting)** 방식을 사용합니다.
    - **특징:** 메시지를 보내는 쪽과 받는 쪽 모두 상대방에 대해 신경 쓰지 않습니다.
- **유니티 이벤트 메서드:** Start(), Update(), OnTriggerEnter() 등 유니티가 특정 시점에 자동으로 브로드캐스팅하는 메시지에 의해 실행되는 메서드입니다

# 2. 유니티 인터페이스 및 씬 편집(2 주차)

## 1. 주요 에디터 창

- Scene 창: 게임 월드를 시각적으로 편집합니다.
- Hierarchy 창: 씬에 존재하는 모든 게임 오브젝트가 나열됩니다.
- Inspector 창: 선택한 게임 오브젝트의 정보(컴포넌트)가 표시됩니다.
- Game 창: 플레이어가 보게 될 화면을 표시합니다. (Main Camera 영역)
- Project 창: 프로젝트에 사용할 에셋(Assets)들이 표시됩니다.
- Console 창: 로그나 에러가 표시됩니다. (일반: 흰색, 경고: 노란색, 에러: 빨간색).

## 2. 씬 편집 툴 (툴 박스)

| 툴 | 단축키 | 기능 |
|---|---|---|
| Hand 툴 | Q | 씬 카메라 이동. |
| Translate 툴 | W | 오브젝트를 이동시킵니다. |
| Rotate 툴 | E | 오브젝트를 회전시킵니다. |
| Scale 툴 | R | 오브젝트의 크기(배율)를 조정합니다. |
| Rect 툴 | T | UI와 2D 오브젝트의 크기를 조정합니다. |
| Transform 툴 | - | 평행이동, 회전, 스케일 툴을 합친 툴입니다. |

# 3. 방향, 크기, 회전 및 공간 개념(4, 6 주차)

### 1. 벡터 수학 및 Vector3 응용 (4 주차)

- **벡터 정의:** 방향과 크기를 갖는 공간상의 화살표 또는 나열된 숫자 데이터 묶음입니다.
- **벡터 연산**
    - **크기(Magnitude):**

$$\sqrt{x^2 + y^2 + z^2}$$

    - **정규화(Normalization):** 크기를 1 로 만들어 방향만을 나타내는 벡터(Direction Vector)를 만듭니다.(Vector3.normalized)
    - **내적(Dot):** 두 벡터 간의 유사도(방향 일치 정도)를 측정하며, 수직이며 0 입니다.(Vector3.Dot)
    - **외적(Cross):** 두 벡터 모두에 수직인 벡터(법선 벡터)를 구합니다.(Vector3.Creoss)
- **Vector3 응용**
    - **두 점 사이의 거리:** Vector3.Distance(currentPos, destPos) 또는 (destPos - currentPos).magnitude.
    - **현재 위치에서 목적지로 향하는 방향:** (destPos - currentPos).normalized.

### 2. 회전: 쿼터니언 (Quaternion) (4 주차)

- **쿼터니언:** 회전을 나타내는 타입으로, 짐벌락을 피하기 위해 회전을 한 번에 처리합니다.
- **생성:** Quaternion.Euler(Vector3)를 사용하여 오일러 각(Euler Angle)으로부터 쿼터니언을 생성합니다.
- **회전 작용:** transform.rotation = Quaternion.Euler(new Vector3(x, y, z));

### 3. 유니티 공간 개념 (6 주차 내용 포함)

유니티는 위치, 회전, 크기를 측정하는 기준이 되는 여러 공간 개념을 사용합니다.

- **전역 공간 (Global Space) / 월드 공간 (World Space):** 월드의 중심이라는 절대 기준을 사용하여 위치를 측정합니다.
    - 전역 공간으로 평행 이동: transform.Translate(new Vector3(0, 1, 0), Space.World);
- **지역 공간 (Local Space) / 오브젝트 공간 (Object Space):**
    - 원래는 오브젝트 자신의 축을 기준으로 하는 **오브젝트 공간**과 부모 오브젝트를 기준으로 하는 **지역 공간**이 있었으나, 유니티는 편의상 이 둘을 합쳐 **지역 공간**이라고 부르고 있습니다.
    - **위치/회전/스케일 값 측정: 부모 게임 오브젝트를 기준**으로 측정합니다 (지역 공간).
        - transform.localPosition: 부모 기준의 위치를 설정/접근합니다.
    - **평행 이동: 게임 오브젝트 자신의 방향을 기준**으로 평행 이동합니다 (오브젝트 공간).
        - transform.Translate(Vector3.forward * speed * Time.deltaTime);
- **Transform 방향:**
    - transform.position: 오브젝트의 **전역 위치**(World Space)를 설정/접근합니다.
    - transform.rotation: 오브젝트의 **전역 회전**(World Space)를 설정/접근합니다.
    - transform.localScale: 오브젝트의 **지역 크기**(Local Space)를 설정/접근합니다

# 4. 게임 디자인: 아이데이션 및 프로세스(5, 7 주차)

## 1. 게임 제작 프로세스의 4 단계

| 단계 | 주요 활동 | 주요 결과물 |
|---|---|---|
| **아이데이션** | 아이디어, 조사, 프로토타입 | 아이디어 목록, 조사 노트, **프로젝트 목표** |
| **프리 프로덕션** | 게임 디자인 매크로, 스케줄 수립, 버티컬 슬라이스 개발 | 버티컬 슬라이스, 게임 디자인 매크로, 스케줄 |
| **풀 프로덕션** | 기능/콘텐츠 구현 및 테스트 | **알파 마일스톤**(기능 완성), **베타 마일스톤**(콘텐츠 완성) |
| **포스트 프로덕션** | 최종 출시 준비 및 사후 지원 | **출시 후보** (Release Candidate) |

- **버티컬 슬라이스(Vertical Slice)**
  - 게임 디자인의 단면 스냅샷이자, 게임의 '고품질 데모'입니다.
  - 전체 게임 경험에 필수적인 핵심 피처, 에셋, 내러티브 샘플이 포함됩니다.

## 2. 아이데이션 주요 활동 (5, 7 주차)

- **푸른 하늘 사고방식**
  - 제한 없이 새롭고 혁신적인 아이디어를 떠올리는 활동입니다.(예: 브레인스토밍, 마인드 맵, 자동기술법)
    - **브레인스토밍 규칙: 질보다 양에 집중, 특이한 아이디어 환영** (구현 생각 x), **토론하지 않기**
- **프로토타이핑**
  - **"프로토타입은 게임의 데모가 아니다"**. 게임에 대한 하나 이상의 아이디어를 탐색하는데 집중해야 하며, 간단해야 하고 요점을 간결하게 유지해야 합니다.
    - **게임 메커닉**: 게임의 **규칙과 과정**이며, 게임의 **동사**에 해당합니다.
- **플레이 테스트**
  - 가능한 자주 그리고 일찍 플레이하도록 해야 하며, **플레이어를 돕거나 간섭하지 않고** 관찰해야합니다.
  - **플레이어는 결코 "잘못 플레이"하지 않으며**, 그들의 행동은 게임 디자인 품질에 대한 진정한 표현입니다.

### 3. 프로젝트 목표 (7 주차)

아이데이션이 끝날 때 명확한 목표를 설정합니다.

- **경험 목표 (Experience Goal):** 플레이어가 경험하기를 원하는 **감정적 경험**을 설명합니다.
- **디자인 목표 (Design Goal):** 경험 목표를 보완하며, 하드웨어, 장르, 메터닉, 인터페이스 유형 등과 관련됩니다.

### 4. 게임 디자인 기술로서의 커뮤니케이션 (7 주차)

게임 디자이너는 **커뮤니케이터**입니다.

- **핵심 기술: 명확성, 간결함, 적극적으로 경청하기**
- **샌드위칭(Sandwiching):** 피드백을 **칭찬 → 건설적인 비판 → 칭찬** 순서로 전달하여 신뢰가 아직 형성되지 않은 관계에서 효과적입니다. 비판은 **사람이 아닌 작품**을 비난해야 합니다.