**Squad Shooter 템플릿 매뉴얼 (Unity 기반)**

**전체 기능 개요**

**Squad Shooter 템플릿**은 Unity 엔진으로 제작된 완성형 탑다운 슈팅 게임 프로젝트로, 여러 시스템이 통합되어 있습니다. 주요 구성 요소와 기능은 다음과 같습니다:

* **플레이어 캐릭터 시스템**: 플레이어 캐릭터 컨트롤 및 커스터마이즈 시스템을 제공합니다. 기본 캐릭터와 추가 캐릭터(스킨)는 데이터베이스로 관리되며, 특정 레벨 도달 시 잠금 해제됩니다 ([Experience System | Notion](https://wmelongames.notion.site/Experience-System-f88c15e9f92d4cf8a0dea0b2a5035ecd#:~:text=,player%20skin%20in%20our%20case)) ([Experience System | Notion](https://wmelongames.notion.site/Experience-System-f88c15e9f92d4cf8a0dea0b2a5035ecd#:~:text=Image%3A%20Untitled)). 캐릭터별 능력치와 업그레이드(예: 체력 상승 등)가 가능하며, **여러 캐릭터 추가**도 지원됩니다 (새 캐릭터 프리팹을 등록하면 캐릭터를 늘릴 수 있음). 또한 게임 진행 중 캐릭터를 변경하는 기능까지 갖추고 있어 유연한 플레이 스타일을 구현할 수 있습니다.
* **적 AI 시스템**: 다양한 종류의 적 캐릭터와 AI를 포함합니다. 적들은 NavMesh 기반으로 플레이어를 추적하고 공격하며, 각 적 유형별로 고유한 행동 패턴과 속성이 있습니다. 템플릿에는 **AI 적 시스템과 커스터마이즈 기능**이 포함되어 있어 새로운 적 유형을 손쉽게 추가하거나 기존 적 모델을 교체할 수 있습니다 ([Squad Shooter | Template + Editor](https://www.gameassetdeals.com/asset/274208/squad-shooter-template-editor#:~:text=TEMPLATE%20INCLUDES)). 적들의 공격 방식(원거리 사격, 근접 공격 등)과 속도/체력 등의 수치는 데이터로 조정 가능하며, **보스 전투**와 같은 특별한 적도 구현할 수 있습니다.
* **무기 시스템**: 여러 **총기 무기와 탄환 시스템**을 제공합니다. 권총, 산탄총, 기관단총 등 다양한 무기가 구현되어 있으며 각 무기는 **속성치**(데미지, 연사속도, 사거리 등)와 **레어리티(등급)** 개념을 가집니다. 무기의 \*\*등급(레어리티)\*\*별로 UI 색상이 표시되고 이름이 구분되며, 새로운 무기나 등급을 추가할 수 있는 구조를 갖추고 있습니다 ([Weapon System | Notion](https://wmelongames.notion.site/Weapon-System-c516b2022bfc476fb0024d89c81d8da4#:~:text=How%20to%20add%20a%20weapon)). 무기 변경 및 업그레이드 시스템도 포함되어 있어, 게임 중 플레이어가 무기를 교체하거나 강화할 수 있습니다 (예: 더 높은 등급의 무기 해금 또는 공격력 업그레이드). 탄환(Bullet) 객체 역시 독립적으로 관리되어 여러 종류의 탄환 효과(관통, 폭발 등)를 추가할 수 있습니다 ([Weapon System | Notion](https://wmelongames.notion.site/Weapon-System-c516b2022bfc476fb0024d89c81d8da4#:~:text=How%20to%20modify%20weapon%20settings)).
* **레벨 및 월드 시스템**: **다수의 레벨**을 다양한 **월드**로 구성하여 게임 진행을 단계화합니다. 여러 \*\*월드(World)\*\*가 존재하며, 각 월드는 고유한 환경 테마(예: 사막, 폐광 등)를 지니고 해당 월드의 장애물/배경이 적용됩니다 ([World System | Notion](https://wmelongames.notion.site/World-System-a719e8ab7f5a446c8e25d4e35c687c80#:~:text=Worlds%20group%20levels%20of%20the,assets)). 각 월드는 다수의 \*\*스테이지(Level)\*\*를 포함하여 캠페인 형태로 진행되며, 월드별 난이도와 환경 차이를 줍니다. 템플릿에는 \*\*레벨 에디터(Level Editor)\*\*가 내장되어 있어 개발자가 직접 새로운 레벨을 편리하게 생성/편집할 수 있습니다 ([Squad Shooter | Template + Editor](https://www.gameassetdeals.com/asset/274208/squad-shooter-template-editor#:~:text=%F0%9F%A6%B8%F0%9F%8F%BB%E2%80%8D%E2%99%82%EF%B8%8F%20Character%20system%20%2B%20customization)). 레벨 에디터를 통해 월드와 레벨을 추가하거나 수정하고, 레벨 내 적 배치, 장애물 배치 등을 시각적으로 구성할 수 있습니다. 또한 **여러 개의 방(Room)으로 구성된 레벨** 구조를 지원하여 한 스테이지 안에서도 구획된 전투 구역과 진행 경로(게이트로 연결)를 만들 수 있습니다. 레벨 데이터는 **World\_XX 에셋**으로 관리되며, 월드별로 레벨 리스트와 환경설정이 저장됩니다 ([World System | Notion](https://wmelongames.notion.site/World-System-a719e8ab7f5a446c8e25d4e35c687c80#:~:text=Worlds%20group%20levels%20of%20the,assets)) ([World System | Notion](https://wmelongames.notion.site/World-System-a719e8ab7f5a446c8e25d4e35c687c80#:~:text=Down%20below%20you%20will%20see,related%20to%20the%20selected%20world)).
* **경험치 및 업그레이드 시스템**: 플레이어의 **경험치(XP) 획득 및 레벨 업** 시스템이 포함되어 있습니다. 각 스테이지 클리어 시 일정량의 경험치를 획득하며, 누적 XP가 요구치를 채우면 플레이어 레벨이 상승합니다 ([Experience System | Notion](https://wmelongames.notion.site/Experience-System-f88c15e9f92d4cf8a0dea0b2a5035ecd#:~:text=,player%20skin%20in%20our%20case)). 플레이어 레벨이 오르면 새로운 캐릭터 스킨 등 콘텐츠가 해금되며 ([Experience System | Notion](https://wmelongames.notion.site/Experience-System-f88c15e9f92d4cf8a0dea0b2a5035ecd#:~:text=,player%20skin%20in%20our%20case)) ([Experience System | Notion](https://wmelongames.notion.site/Experience-System-f88c15e9f92d4cf8a0dea0b2a5035ecd#:~:text=Image%3A%20Untitled)), 이를 통해 플레이어의 진행 동기를 부여합니다. 경험치 요구량은 **경험치 데이터베이스**(Experience Database)로 관리되어 레벨별 필요 XP를 조정할 수 있습니다 ([Experience System | Notion](https://wmelongames.notion.site/Experience-System-f88c15e9f92d4cf8a0dea0b2a5035ecd#:~:text=To%20configure%20how%20much%20experience,folder)). 또한 코인을 사용한 **능력 업그레이드** 시스템도 갖추고 있어, 플레이어는 스테이지 사이에 **무기 공격력**이나 **캐릭터 능력치**를 업그레이드할 수 있습니다. 이러한 업그레이드 시스템은 **자동 난이도 조절**과 연계되어, 각 레벨마다 “필요 업그레이드 수치”를 설정해 두면 플레이어의 실제 업그레이드 상황에 따라 적의 능력치가 자동으로 조정됩니다 (플레이어가 충분히 강화됐다면 적을 약화시키고, 강화가 부족하면 적을 강화하여 난이도 균형을 맞춤) ([Difficulty Balancing | Notion](https://wmelongames.notion.site/Difficulty-Balancing-08ea6e1f9a6e4d43a9e58618558a6be9#:~:text=Each%20level%20has%20a%20field,required%2C%20enemies%20will%20feel%20stronger)). 이를 통해 게임 전반의 **난이도 밸런싱**을 손쉽게 관리할 수 있습니다 ([Difficulty Balancing | Notion](https://wmelongames.notion.site/Difficulty-Balancing-08ea6e1f9a6e4d43a9e58618558a6be9#:~:text=,them%20stronger%2C%20normal%2C%20or%20weaker)) ([Difficulty Balancing | Notion](https://wmelongames.notion.site/Difficulty-Balancing-08ea6e1f9a6e4d43a9e58618558a6be9#:~:text=To%20%E2%80%9Cexplain%E2%80%9D%20the%20game%20how,want%20the%20player%20to%20be)).
* **UI 및 사용자 인터페이스**: 게임 진행에 필요한 다양한 UI가 제공됩니다. 메인 메뉴, 일시정지, 게임 오버/클리어 창, 체력/탄약 등의 **HUD**, 그리고 **상점(UI Store)** 인터페이스가 포함됩니다. **UI Store**는 게임 내 상점 화면으로, 여기에서 플레이어는 \*\*인게임 통화(코인)\*\*로 업그레이드하거나, **현금 결제(IAP)** 또는 **보상형 광고 시청**을 통해 아이템을 구매할 수 있습니다 ([UI Store | Notion](https://wmelongames.notion.site/UI-Store-18bc0f2fe155808193e4d5d7fd72bdfd#:~:text=The%20UI%20Store%20displays%20various,Module%20and%20the%20Rewards%20System)). 기본적으로 제공되는 상품으로는 코인 패키지(인게임 화폐 지급), No Ads(광고 제거), 스타터 팩(코인 + 광고제거 묶음) 등이 있습니다 ([UI Store | Notion](https://wmelongames.notion.site/UI-Store-18bc0f2fe155808193e4d5d7fd72bdfd#:~:text=Rewards%20Implemented%20in%20this%20Template)). UI 및 UX는 모바일 환경에 최적화되어 있으며, 시각적인 요소와 애니메이션이 깔끔하게 구현되어 있습니다. 개발자는 제공된 UI 프리팹과 애니메이션을 커스터마이즈하여 자신만의 스타일로 변경할 수 있습니다.
* **수익화 (Monetization)**: 모바일 게임의 수익화를 위한 **광고 및 인앱결제** 모듈이 완비되어 있습니다. AdMob, Unity Ads, Unity LevelPlay 등의 주요 광고 네트워크 연동을 지원하며, 배너 광고, 전면(인터stitial) 광고, 리워드 동영상 광고 기능이 미리 통합되어 있습니다 ([Monetization (Ads & IAP) | Notion](https://wmelongames.notion.site/Monetization-Ads-IAP-13bc0f2fe155801f9db2d7b4f0e43b74#:~:text=,developers%20looking%20to%20generate%20revenue)). 개발자는 코드 수정 없이 **광고 ID만 설정**하면 바로 광고를 구현할 수 있을 정도로 사용이 간편합니다 ([Monetization (Ads & IAP) | Notion](https://wmelongames.notion.site/Monetization-Ads-IAP-13bc0f2fe155801f9db2d7b4f0e43b74#:~:text=,developers%20looking%20to%20generate%20revenue)) ([Monetization (Ads & IAP) | Notion](https://wmelongames.notion.site/Monetization-Ads-IAP-13bc0f2fe155801f9db2d7b4f0e43b74#:~:text=)). 또한 **인앱 구매(IAP)** 시스템도 포함되어 있어, 예를 들어 게임 내 "광고 제거", "코인 패키지" 등의 상품을 손쉽게 판매할 수 있습니다 ([Squad Shooter | Template + Editor | Packs | Unity Assets - Codeintra](https://www.codeintra.com/items/1143/squad-shooter-template-editor-packs-unity-assets?srsltid=AfmBOorIJ3ALhSAcenBUZ1cBiU0tBoe7z0iznulZST8BYoAYzDXnaCb-#:~:text=%F0%9F%92%B0%20Monetization%3A)). 제공되는 Monetization 설정 파일에서 간단히 토글을 켜거나 끌 수 있으며, 필요 시 **추가 광고 네트워크 커스터마이즈**도 가능한 유연한 구조입니다 ([Squad Shooter | Template + Editor | Packs | Unity Assets - Codeintra](https://www.codeintra.com/items/1143/squad-shooter-template-editor-packs-unity-assets?srsltid=AfmBOorIJ3ALhSAcenBUZ1cBiU0tBoe7z0iznulZST8BYoAYzDXnaCb-#:~:text=%F0%9F%92%B0%20Monetization%3A)). 이 수익화 시스템은 **게임 보상 시스템**과도 연계되어, 보상형 광고 시 코인 보상 또는 아이템 지급 등이 자동 처리됩니다. iOS/Android 등 **모바일 플랫폼에 최적화**되어 있고, **WebGL 및 데스크톱**에서도 동작하도록 지원됩니다 ([Squad Shooter | Template + Editor | Packs | Unity Assets - Codeintra](https://www.codeintra.com/items/1143/squad-shooter-template-editor-packs-unity-assets?srsltid=AfmBOorIJ3ALhSAcenBUZ1cBiU0tBoe7z0iznulZST8BYoAYzDXnaCb-#:~:text=%E2%80%8E%20%E2%80%8E%20%E2%80%8E%20%E2%80%8E%20%F0%9F%94%B8,Unity%20Ads)).

以上의 기능들을 통합하여 **Squad Shooter 템플릿**은 즉시 플레이 가능한 완성도 높은 슈팅 게임을 구현하며, 구조가 깔끔하고 커스터마이즈가 쉬워 새로운 아이디어를 적용하기에 용이합니다 ([Squad Shooter | Template + Editor by Watermelon Games](https://wmelongames.itch.io/squad-shooter-template-editor#:~:text=This%20fully%20customizable%20shooter%20template,point%20for%20your%20next%20hit)) ([Squad Shooter | Template + Editor by Watermelon Games](https://wmelongames.itch.io/squad-shooter-template-editor#:~:text=%E2%9C%A8%20Clean%20structure%20and%20code)). 개발자는 이 템플릿을 기반으로 자신의 게임을 빠르게 제작하고 원하는 대로 확장할 수 있습니다.

**게임 개발 단계별 가이드**

이 섹션에서는 Squad Shooter 템플릿을 활용하여 게임을 개발하는 과정을 단계별로 안내합니다. Unity 프로젝트 생성부터 캐릭터/레벨 구성, 수익화 설정, 빌드까지 **중급 개발자** 관점에서 필요한 절차를 다룹니다.

**1. 프로젝트 설정 (Project Setup)**

1. **Unity 설정**: 템플릿에 권장되는 Unity 버전을 설치합니다 (예: Unity 2021 LTS 또는 2022 LTS 버전). Watermelon Games에서는 해당 템플릿에 최적화된 Unity 버전을 권장하고 있으므로, 가능하면 동일한 버전을 사용합니다 ([Squad Shooter | Notion](https://www.notion.so/wmelongames/Squad-Shooter-1a169bfcedf84bddac47d1df1685b803#:~:text=Recommended%20Unity%20version%20Unity%206,LTS)). Unity Hub를 통해 권장 버전을 설치한 후 **3D 템플릿**으로 새 프로젝트를 만듭니다 ([Project creation and template import | Notion](https://wmelongames.notion.site/Project-creation-and-template-import-55e83b532df046a9abe39169f0db25df#:~:text=,Unity%20project%20using%203D%20template)).
2. **템플릿 임포트**: 새로운 빈 Unity 3D 프로젝트를 준비했으면, Squad Shooter 템플릿 패키지를 가져옵니다. 패키지 임포트 방법은 두 가지입니다 ([Project creation and template import | Notion](https://wmelongames.notion.site/Project-creation-and-template-import-55e83b532df046a9abe39169f0db25df#:~:text=1,the%20%E2%80%9CImport%E2%80%9D%20button)):
   * Unity Asset Store에서 구매한 경우, Unity 에디터의 **Package Manager** 혹은 Asset Store 창에서 다운로드 후 임포트합니다.
   * 또는 제공된 .unitypackage 파일을 수동으로 임포트할 수 있습니다: Unity 상단 메뉴에서 **Assets > Import Package > Custom Package**를 클릭하고 패키지 파일을 선택합니다 ([Project creation and template import | Notion](https://wmelongames.notion.site/Project-creation-and-template-import-55e83b532df046a9abe39169f0db25df#:~:text=1,the%20%E2%80%9CImport%E2%80%9D%20button)). 임포트 과정에서 **모든 항목을 선택한 뒤 Import 버튼**을 눌러 템플릿의 에셋, 스크립트, 설정 파일들을 프로젝트에 통합합니다 ([Project creation and template import | Notion](https://wmelongames.notion.site/Project-creation-and-template-import-55e83b532df046a9abe39169f0db25df#:~:text=1,the%20%E2%80%9CImport%E2%80%9D%20button)). 임포트 완료 후 나타나는 **“Install/Upgrade” 창**이 있다면 Dependencies(의존성 패키지) 설치를 위해 **Install/Upgrade** 버튼을 눌러 필요한 의존 패키지(예: Cinemachine, NavMesh components 등)를 설치합니다 ([Project creation and template import | Notion](https://wmelongames.notion.site/Project-creation-and-template-import-55e83b532df046a9abe39169f0db25df#:~:text=,to%20install%20dependencies)).
3. **초기 실행 테스트**: 임포트가 완료되면, 먼저 제공된 데모 씬을 실행해 제대로 동작하는지 확인합니다. 템플릿의 기본 게임 씬은 **Assets/Project Files/Game/Scenes/Game.unity** 파일로 제공됩니다 ([Project creation and template import | Notion](https://wmelongames.notion.site/Project-creation-and-template-import-55e83b532df046a9abe39169f0db25df#:~:text=,to%20run%20the%20game)). 해당 씬을 열고 Unity 에디터에서 **재생(Play)** 버튼을 눌러 게임이 정상적으로 시작되는지 확인합니다 ([Overview | Notion](https://wmelongames.notion.site/Overview-9d232a8c70a04e6e9e5d2e5150a0e1f6#:~:text=First%20launch)). 또는 상단 메뉴의 **Actions > Game Scene** 단축 메뉴를 통해 빠르게 Game 씬을 열 수도 있습니다 ([Overview | Notion](https://wmelongames.notion.site/Overview-9d232a8c70a04e6e9e5d2e5150a0e1f6#:~:text=First%20launch)). 정상적으로 임포트되었다면 메인 메뉴 UI와 함께 플레이어, 적 등이 등장하는 게임 데모가 실행될 것입니다.
4. **프로젝트 구조 파악**: 프로젝트 탭에서 **Project Files** 폴더를 열어 구조를 살펴봅니다. 폴더 구조는 이후 섹션에서 자세히 다루겠지만, **Game** 폴더에는 씬, 프리팹, 스크립트 등이, **Data** 폴더에는 각종 ScriptableObject 자원(데이터베이스, 설정)이 정리되어 있습니다. 임포트 직후 오류가 없고 데모가 동작한다면, 본격적인 커스터마이징을 시작할 준비가 된 것입니다.

**2. 캐릭터 구성 (Player Character Setup)**

이 단계에서는 **플레이어 캐릭터를 설정 및 커스터마이즈**하는 방법을 안내합니다. 기본 템플릿에는 하나의 플레이어 캐릭터 프리팹과 여러 캐릭터 스킨 데이터가 포함되어 있으며, 개발자는 이를 바탕으로 새로운 캐릭터를 추가하거나 기존 것을 변경할 수 있습니다.

* **캐릭터 프리팹 확인**: Project Files/Game/Prefabs/Characters (혹은 유사 폴더)에서 플레이어 캐릭터 프리팹을 찾습니다. 이 프리팹에는 플레이어 모델, 애니메이션, CharacterController 또는 NavMeshAgent, 무기 장착 위치, 피격 판정 collider 등이 포함됩니다. 에디터에서 이 프리팹을 열어 **모델을 교체**하거나 **속성 값을 조절**할 수 있습니다. 예를 들어 다른 3D 모델로 교체하려면 프리팹의 자식 오브젝트인 모델을 대체하고, 애니메이션 Avatar 세팅을 새 모델에 맞게 적용합니다 ([Enemies | Notion](https://wmelongames.notion.site/Enemies-cc061ab4daf04a05ab9c1ef6a3631716#:~:text=How%20to%20replace%20the%20existing,enemy%20model)).
* **캐릭터 데이터베이스 수정**: Project Files/Data/Characters System/Character Database 에셋을 검사합니다. 여기에는 사용 가능한 캐릭터(스킨)의 목록과 각 캐릭터의 속성(이름, 모델 참조, 필요한 플레이어 레벨 등)이 저장되어 있습니다 ([Experience System | Notion](https://wmelongames.notion.site/Experience-System-f88c15e9f92d4cf8a0dea0b2a5035ecd#:~:text=Image%3A%20Untitled)). 예를 들어 두 번째 캐릭터 스킨은 “필요 플레이어 레벨 = 5”로 설정되어 있다면, 플레이어 레벨 5에 도달해야 해당 캐릭터를 선택할 수 있습니다 ([Experience System | Notion](https://wmelongames.notion.site/Experience-System-f88c15e9f92d4cf8a0dea0b2a5035ecd#:~:text=,player%20skin%20in%20our%20case)) ([Experience System | Notion](https://wmelongames.notion.site/Experience-System-f88c15e9f92d4cf8a0dea0b2a5035ecd#:~:text=Image%3A%20Untitled)). 이 값을 조정하면 캐릭터 잠금 해제 조건을 바꿀 수 있습니다. 새로운 캐릭터를 추가하려면 우선 **캐릭터 프리팹**을 만든 후, Character Database에 항목을 하나 추가하고 그 프리팹과 해금 조건 등을 설정합니다.
* **캐릭터 능력치 및 업그레이드**: 캐릭터의 체력, 이동속도 등의 기본 능력치는 Character 관련 스크립트나 데이터 에셋으로 관리됩니다. 예를 들어 CharacterStats ScriptableObject가 있다면 거기서 레벨업에 따른 체력 증가량 등을 정의할 수 있습니다. 또한 플레이어가 코인으로 **캐릭터 업그레이드**를 할 수 있는 경우, 해당 업그레이드 단계별 수치와 비용이 데이터로 정의되어 있을 것입니다. (이 부분은 Weapon System의 업그레이드와 유사하게 동작합니다).
* **게임 내 캐릭터 변경**: 템플릿은 게임 진행 중에 캐릭터를 바꾸는 기능도 제공합니다 (예: 특정 아이템 획득 시 다른 아군 캐릭터로 교체 등). 기본적으로는 메인 메뉴에서 사용 가능한 캐릭터 중 하나를 선택하여 게임을 시작하며, 플레이 도중에는 변경하지 않는 형태입니다. 그러나 필요하다면 **여러 캐릭터 동시 사용**도 구현 가능합니다 (스쿼드 구성). 해당 기능을 구현하려면 새로운 캐릭터를 스폰하고 컨트롤을 전환하는 로직을 작성하거나, 템플릿 제공 함수를 활용하면 됩니다. (자세한 내용은 Watermelon Games 문서의 "How to change character during gameplay" 부분을 참고하세요 ([Player Character | Notion](https://wmelongames.notion.site/Player-Character-9b7dc41755384acc8e65209e1f9a4f2f#:~:text=How%20to%20add%20characters)).)

**3. 맵 디자인 및 레벨 구성 (Map Design & Level Setup)**

Squad Shooter 템플릿은 **전용 레벨 에디터 도구**를 통해 맵을 디자인하고 레벨을 구성하는 워크플로우를 제공합니다. 이를 사용하여 프로그래밍 없이 시각적으로 스테이지를 제작할 수 있습니다. 주요 절차는 다음과 같습니다:

* **레벨 에디터 열기**: Unity 에디터 상단 메뉴의 **Tools > Level Editor**를 클릭하여 **레벨 편집기 윈도우**를 엽니다 ([World System | Notion](https://wmelongames.notion.site/World-System-a719e8ab7f5a446c8e25d4e35c687c80#:~:text=World%20assets%20are%20accessible%20through,the%20currently%20selected%20world%20here)). 레벨 에디터 창 상단에는 현재 편집 중인 **월드 선택** 부분이 표시됩니다. 좌우 화살표 (◀▶) 버튼으로 월드를 변경할 수 있으며, 중앙에는 현재 선택된 World\_XX 자산 이름이 표시됩니다 ([World System | Notion](https://wmelongames.notion.site/World-System-a719e8ab7f5a446c8e25d4e35c687c80#:~:text=World%20assets%20are%20accessible%20through,the%20currently%20selected%20world%20here)). 예를 들어 "World\_01" (챕터1: Desert 테마)이 선택되어 있다면, 해당 월드의 레벨들을 편집하게 됩니다.
* **월드 환경 설정 확인**: 레벨 에디터에는 두 개의 탭이 있습니다 ([World System | Notion](https://wmelongames.notion.site/World-System-a719e8ab7f5a446c8e25d4e35c687c80#:~:text=Down%20below%20you%20will%20see,related%20to%20the%20selected%20world)):
  + **Levels Creation 탭**: 현재 월드에 속한 레벨들의 목록이 표시됩니다. 각 레벨 항목에는 레벨 번호, 이름, 난이도 설정 등이 나오며, 새로운 레벨을 추가하거나 기존 레벨을 삭제/편집할 수 있는 UI가 있습니다 ([World System | Notion](https://wmelongames.notion.site/World-System-a719e8ab7f5a446c8e25d4e35c687c80#:~:text=Down%20below%20you%20will%20see,related%20to%20the%20selected%20world)).
  + **World Settings 탭**: 현재 월드에 대한 환경 설정과 등록된 오브젝트 목록이 표시됩니다 ([World System | Notion](https://wmelongames.notion.site/World-System-a719e8ab7f5a446c8e25d4e35c687c80#:~:text=Down%20below%20you%20will%20see,related%20to%20the%20selected%20world)) ([Obstacles | Notion](https://wmelongames.notion.site/Obstacles-bc6922b59fff4090ab9eb1ab35d40ff2#:~:text=Image%3A%20Untitled)). 여기에서 이 월드에서 사용 가능한 \*\*장애물(Obstacles)\*\*과 \*\*환경 오브젝트(Environment)\*\*의 리스트를 확인할 수 있으며, 월드의 배경 테마, 지형 프리팹, 조명 설정 등을 조정할 수도 있습니다.
* **레벨 추가 및 기본 구성**: 새로운 레벨을 만들기 위해 Levels Creation 탭에서 **Add Level**(+) 버튼을 클릭합니다. 그러면 해당 월드에 새 레벨 엔트리가 생성됩니다. 새 레벨을 선택하면 편집을 시작할 수 있습니다. 레벨 속성으로 **Required Upgrades(필요 업그레이드)** 등의 난이도 관련 설정을 입력할 수 있습니다 (이 값은 플레이어가 해당 레벨을 “보통 난이도”로 느끼기 위해 요구되는 업그레이드 수준이며, 난이도 밸런싱에 활용됩니다 ([Difficulty Balancing | Notion](https://wmelongames.notion.site/Difficulty-Balancing-08ea6e1f9a6e4d43a9e58618558a6be9#:~:text=Each%20level%20has%20a%20field,required%2C%20enemies%20will%20feel%20stronger))). 또한 레벨별 **경험치/코인 보상** 양도 설정할 수 있습니다. 새 레벨 생성 시 기본적으로 비어 있는 지형만 있는 상태이므로, 다음으로는 실제 지형과 장애물을 배치하게 됩니다.
* **지형 및 방 구조 디자인**: **환경(Environment)** 요소로 지면(바닥)과 벽, 장애물 등을 배치하여 레벨의 지형을 설계합니다 ([Environment | Notion](https://wmelongames.notion.site/Environment-991c663dc33443659795e9d6be81a6b0#:~:text=,like%20ground%2C%20walls%2C%20and%20gates)). Environment 항목에는 **Ground**(플레이어와 NPC가 걸어다니는 바닥), **Wall**(벽체), **Gate**(문/게이트) 등이 포함됩니다 ([Environment | Notion](https://wmelongames.notion.site/Environment-991c663dc33443659795e9d6be81a6b0#:~:text=,like%20ground%2C%20walls%2C%20and%20gates)). Ground로 설정된 오브젝트는 NavMesh 상 **보행 가능 영역**으로 간주되고, Wall은 시각적인 장치로서 배치됩니다 (벽은 기본적으로 이동을 막지 않고 장식용이며, **NavMeshObstacle**을 추가하면 실제 이동을 차단할 수도 있습니다 ([Environment | Notion](https://wmelongames.notion.site/Environment-991c663dc33443659795e9d6be81a6b0#:~:text=3,prevent%20aiming%20through%20the%20obstacles))). Level Editor의 World Settings 탭에서 해당 월드에 등록된 지형 프리팹들을 확인할 수 있고, 필요하면 추가 등록도 가능합니다 ([Environment | Notion](https://wmelongames.notion.site/Environment-991c663dc33443659795e9d6be81a6b0#:~:text=1.%20Right,Add%20into%20Level%20Editor)). Scene 뷰에서 **드래그 앤 드롭**이나 Level Editor 창 내 배치 기능을 통해 바닥 타일을 깔고 벽이나 장치를 배치하세요.
  + **방(Room) 구성**: 하나의 레벨은 한 개 이상의 **방**으로 이루어질 수 있습니다. 방은 벽과 게이트로 구획되어 있으며, 플레이어는 방의 적을 모두 처치하면 다음 방으로 진입하는 구조를 만들 수 있습니다. 첫 번째 방을 디자인한 후, 출구 지점에 **Gate 프리팹**을 배치하고 문 너머에 두 번째 방의 지형을 이어서 배치합니다. 이렇게 하면 게임 플레이 중 한 방의 적을 처치하면 Gate가 열리고 다음 방으로 진행하도록 스크립트가 제어할 수 있습니다. 방별로 테마나 구조를 다르게 주어 다양성을 높일 수 있습니다. (예: 방1은 장애물 많은 좁은 공간, 방2는 탁 트인 열린 공간 등)
* **장애물 배치**: \*\*장애물(Obstacle)\*\*은 플레이어와 적의 움직임을 제한하고 엄폐물을 제공하는 오브젝트입니다 ([Obstacles | Notion](https://wmelongames.notion.site/Obstacles-bc6922b59fff4090ab9eb1ab35d40ff2#:~:text=,movement%20and%20create%20hiding%20spots)). 예를 들어 바리케이드, 통, 바위 등이 이에 해당합니다. 장애물은 **Obstacle** 유형으로 Level Editor에 등록되어 있어야 하며, 월드별로 사용할 수 있는 장애물 프리팹 목록이 World Settings 탭에 나와 있습니다 ([Obstacles | Notion](https://wmelongames.notion.site/Obstacles-bc6922b59fff4090ab9eb1ab35d40ff2#:~:text=Image%3A%20Untitled)). 장애물을 배치할 때에는 NavMeshObstacle 및 물리 충돌 설정이 자동 적용되어, 탄환이 통과하지 못하고 적/플레이어가 둘러가게 됩니다 ([Obstacles | Notion](https://wmelongames.notion.site/Obstacles-bc6922b59fff4090ab9eb1ab35d40ff2#:~:text=3,prevent%20aiming%20through%20the%20obstacles)). 씬 뷰에서 원하는 위치에 장애물 프리팹을 드래그하거나 Level Editor의 **Obstacle** 카테고리에서 선택 후 씬을 클릭하여 배치합니다. 적절하게 회전/위치를 조정해 엄폐 커버를 형성하고 길목을 설계합니다. 장애물 배치로 인해 만들어지는 **진행 경로**를 고려하여, 플레이어가 일방 통로에 갇히지 않도록 디자인합니다.
* **적 배치**: **Enemies** 카테고리에서 적 캐릭터 스폰 포인트를 배치합니다. 템플릿에서는 레벨 에디터를 통해 적 자체를 놓는 대신 **스폰 위치**나 **웨이브 설정**을 사용하는 경우가 일반적입니다. 예를 들어 방마다 적 출현 지점을 몇 곳 지정하고, 각 방이 시작될 때 해당 지점들에 적을 생성하도록 설정할 수 있습니다. Level Editor UI에서 특정 방에 대한 **Enemy Wave 설정** 옵션이 있다면 (예: 방1에 적 5명, 방2에 적 8명 + 보스1명 등), 이를 활용해 입력합니다. 그렇지 않고 수동 배치를 허용한다면, 적 프리팹을 직접 씬에 배치할 수도 있습니다. 배치한 적 프리팹에는 **Enemy AI 스크립트**가 붙어 있으며, 게임 시작 시 자동으로 활성화될 것입니다. 적 유형(총잡이, 돌격병 등)을 다양하게 섞고 배치 위치를 전략적으로 선택하여, 플레이어가 진입했을 때 다양한 방향에서 교전이 발생하도록 디자인합니다.
* **내비메시 및 경로찾기**: 레벨 디자인을 완료했다면, 적들의 AI 이동을 위해 **NavMesh**를 업데이트해야 합니다. 템플릿에서는 **NavMeshSurface**가 미리 구성되어 있을 수 있으며, Play 시 동적으로 NavMesh를 빌드하거나, 에디터 상에서 **Bake**해야 할 수도 있습니다. 장애물과 지형이 변경되었다면 **Navigation 창**에서 베이크(Bake) 버튼을 눌러 NavMesh 데이터를 갱신하세요. Obstacle로 지정된 오브젝트는 NavMesh에서 통로가 없는 **차단 영역**으로 인식되고, Ground 레이어는 **이동 가능 영역**으로 남게 됩니다 ([Environment | Notion](https://wmelongames.notion.site/Environment-991c663dc33443659795e9d6be81a6b0#:~:text=3,prevent%20aiming%20through%20the%20obstacles)). 이 과정을 통해 적 AI가 새로운 지형에 맞게 정상적으로 경로를 찾고 이동합니다.
* **플레이 테스트**: 레벨 구성 후에는 게임을 실행하여 해당 레벨을 테스트합니다. 만약 Level Editor에서 특정 레벨을 선택한 상태에서 **Play**를 누르면 그 레벨이 시작되도록 구현되어 있다면 바로 테스트할 수 있고, 그렇지 않다면 임시로 해당 레벨을 1번으로 만들어 게임을 시작하거나, 디버그 옵션으로 원하는 레벨을 불러오도록 할 수 있습니다. 게임 내에서 적들이 의도대로 스폰되고 이동하는지, 장애물이 잘 배치되어 플레이에 재미를 주는지 확인합니다. 필요한 경우 에디터로 돌아가 오브젝트 위치를 조정하고 다시 테스트합니다.

이러한 과정을 통해 **레벨 디자인**을 반복하며 원하는 수만큼 스테이지를 구성합니다. 템플릿의 Level Editor 도구 덕분에 개발자는 **씬 직관적으로 레벨을 제작**할 수 있고, 각 월드별로 색다른 도전을 설계할 수 있습니다 ([World System | Notion](https://wmelongames.notion.site/World-System-a719e8ab7f5a446c8e25d4e35c687c80#:~:text=Worlds%20group%20levels%20of%20the,assets)) ([World System | Notion](https://wmelongames.notion.site/World-System-a719e8ab7f5a446c8e25d4e35c687c80#:~:text=Down%20below%20you%20will%20see,related%20to%20the%20selected%20world)).

**4. UI 구성 (UI Setup)**

Squad Shooter 템플릿에 포함된 UI를 자신의 게임에 맞게 구성하는 방법입니다. 기본 UI는 모바일 환경을 염두에 두고 제작되었으며, 주요 화면으로 **메인 메뉴**, **게임 HUD**, **일시정지 및 결과 화면**, **상점 화면(UI Store)** 등이 있습니다.

* **메인 메뉴(Main Menu)**: Game 씬을 처음 실행하면 표시되는 화면입니다. 여기에는 게임 타이틀, 시작 버튼 (Play), 설정, 상점, 현재 플레이어 레벨/경험치 표시, 캐릭터/무기 선택 UI 등이 포함될 수 있습니다. Hierarchy에서 MainMenu 관련 오브젝트(예: Canvas\_MainMenu)를 찾아 UI 구성요소를 수정합니다. 예를 들어 게임 타이틀 텍스트를 변경하거나 배경 이미지를 교체할 수 있습니다. **애니메이션**도 함께 제공되어 있다면 (예: 시작 버튼 깜빡임 등), Animation 또는 Animator 창에서 조정할 수 있습니다.
* **게임 HUD(HUD: Head-Up Display)**: 게임 플레이 중에 화면에 겹쳐서 나타나는 UI 요소들입니다. 체력 바, 탄약/무기 아이콘, 일시정지 버튼, 스코어/코인 표시 등이 해당됩니다. Canvas\_HUD 하위에 이러한 UI들이 배치되어 있으니, 디자인적으로 바꾸고 싶다면 UI 스프라이트 교체나 배치 조정을 합니다. 기본적으로 모바일 플레이를 위해 **가상 조이스틱** 또는 간단한 터치 컨트롤 UI가 있을 수 있습니다 (만약 수동 사격 모드가 있다면 사격 버튼 UI 등). 필요에 따라 PC 플랫폼 테스트 시에는 키보드 입력으로 동작하지만, 모바일 빌드 시에는 이 터치 UI가 사용됩니다.
* **일시정지 및 게임오버 화면**: 게임 중 일시정지 메뉴(Pause)와 레벨 클리어 또는 실패 시 나오는 결과 화면이 구현되어 있습니다. 이들도 UI Canvas 형태로 존재하며, Canvas\_Pause, Canvas\_LevelComplete, Canvas\_GameOver 등으로 되어 있을 수 있습니다. 각 화면에 표시되는 문구("Level Clear!", "Game Over", 획득 코인/XP 등)를 수정하거나, 버튼 (재시작, 메인 메뉴로 등) 동작을 연결합니다. 기본 동작은 이미 스크립트로 구현되어 있지만, UI 텍스트는 **다국어 대응**을 위해 텍스트 에셋으로 분리되어 있을 수도 있습니다.
* **상점 화면(UI Store)**: 인앱 구매 및 보상 광고 관련 UI입니다. Canvas\_Store 혹은 해당 UI Prefab을 찾아 열어봅니다. 상점 UI에는 구매 가능한 상품 리스트가 표시됩니다. 기본 상품으로는 앞서 언급한 **Currency Pack (코인 구매)**, **No Ads (광고 제거)**, **Starter Pack** 등이 있고, 각 항목에 가격(IAP 결제 금액)이나 필요 재화가 표시됩니다 ([UI Store | Notion](https://wmelongames.notion.site/UI-Store-18bc0f2fe155808193e4d5d7fd72bdfd#:~:text=Rewards%20Implemented%20in%20this%20Template)). 예를 들어 No Ads 상품에는 "$4.99" 등의 레이블이, Currency Pack에는 "1000 Coins"와 해당 가격이 표시될 것입니다. 이 UI에서 **상품 항목 Prefab**을 복제하여 새로운 상품을 추가하거나, 기존 상품을 제거할 수 있습니다 ([UI Store | Notion](https://wmelongames.notion.site/UI-Store-18bc0f2fe155808193e4d5d7fd72bdfd#:~:text=The%20UI%20Store%20displays%20various,Module%20and%20the%20Rewards%20System)) ([UI Store | Notion](https://wmelongames.notion.site/UI-Store-18bc0f2fe155808193e4d5d7fd72bdfd#:~:text=Rewards%20Implemented%20in%20this%20Template)). 새로운 상품을 추가한 경우, 뒷단의 **IAP 설정**에도 해당 상품을 추가해야 실제 구매가 동작합니다 (다음 단계 Ads/IAP 설정 참조). 상점 UI의 구성요소 (상품 설명, 아이콘 등)를 원하는 대로 변경하고, 레이아웃이나 스타일을 수정하여 게임 분위기에 맞게 꾸밉니다.
* **기타 UI 요소**: 게임 내 튜토리얼 팝업이나 특별 이벤트 배너 등을 추가하고 싶다면, 기존 Canvas에 새로운 패널을 만들거나 별도 Canvas를 추가할 수 있습니다. 템플릿의 UI Manager 스크립트가 UI를 제어하고 있을 것이므로, 새로 추가한 UI를 표시하려면 그에 맞는 스크립트 수정이 필요할 수 있습니다. 하지만 단순히 시각적인 변경이나 텍스트 수정, 버튼 연결 변경 등은 Unity 에디터 상에서 가능하므로, 큰 틀에서 벗어나지 않는 한 편집이 수월합니다.

UI 구성 변경 후에는 에디터 **플레이 모드**에서 화면 표시를 확인해보고, 버튼들이 정상적으로 동작하는지 (예: 상점 버튼 클릭 시 상점 화면 나타나는지 등) 테스트합니다. UI는 플레이어와 첫 상호작용 지점이므로, 게임 특성에 맞춰 충분히 수정해주는 것이 좋습니다.

**5. 적 및 오브젝트 배치 (Enemies & Objects Placement)**

이 단계는 맵 디자인과 맞물려 있지만, 주로 **새로운 적이나 오브젝트를 게임에 추가**하거나 **기존 요소를 배치/조정**하는 관점에서 설명합니다. 템플릿의 확장성을 활용하여 콘텐츠를 늘리는 방법이기도 합니다.

* **새 적 유형 추가**: 만약 기본 제공된 적 이외에 새로운 적 캐릭터를 추가하고 싶다면, 다음 과정을 따릅니다. 우선 새로운 적 모델(프리팹)을 준비합니다. 예를 들어 **보스 캐릭터**를 추가한다고 가정하면, 보스 3D 모델에 애니메이션을 세팅하고 적 AI 스크립트를 붙여 **보스 프리팹**을 만듭니다. 이 프리팹에 **콜라이더와 적 Health 컴포넌트, NavMeshAgent/NavMeshObstacle** 등을 필요한 대로 추가합니다 (기존 적 프리팹을 참고하면 설정해야 할 컴포넌트와 레이어를 쉽게 파악할 수 있습니다). 준비된 프리팹을 프로젝트 창에서 **우클릭 > Add into Level Editor** 옵션을 선택해 레벨 에디터에 등록합니다 ([Obstacles | Notion](https://wmelongames.notion.site/Obstacles-bc6922b59fff4090ab9eb1ab35d40ff2#:~:text=1.%20Right,Add%20into%20Level%20Editor)). 이때 **Type**을 "Enemy"로 선택하고, 사용할 **World**를 지정한 후 Add를 누르면 해당 적이 레벨 에디터의 Enemies 목록에 추가됩니다 ([Obstacles | Notion](https://wmelongames.notion.site/Obstacles-bc6922b59fff4090ab9eb1ab35d40ff2#:~:text=2,Add)). 만약 적 종류별로 여러 타입을 그룹화했다면 Type에 세분화된 카테고리가 있을 수 있습니다. 등록을 마치면 월드 설정의 적 리스트에 새로운 적이 나타나고, 이후 레벨 편집 시 해당 적을 배치/스폰 구성할 수 있게 됩니다 ([Obstacles | Notion](https://wmelongames.notion.site/Obstacles-bc6922b59fff4090ab9eb1ab35d40ff2#:~:text=Image%3A%20Untitled)). 이제 이 적이 게임 내에서 정상 동작하는지 (AI 이동, 공격 등) 스크립트를 조정하고 테스트합니다.
* **적 속성 조정**: 추가하거나 기존에 있는 적의 능력치를 조정하려면 Project Files/Data/Enemy… 관련 데이터(예: Enemy Database 혹은 개별 적 ScriptableObject)를 편집합니다. 여기에는 적의 기본 체력, 공격력, 이동속도 등이 정의되어 있고, 난이도에 따른 보정 수치가 들어갈 수도 있습니다. 템플릿의 **난이도 밸런싱 시스템**에 의해 적의 실제 스탯은 플레이어 업그레이드 상황에 따라 자동 조절되므로, 기본 수치를 정할 때 이를 염두에 둡니다 ([Difficulty Balancing | Notion](https://wmelongames.notion.site/Difficulty-Balancing-08ea6e1f9a6e4d43a9e58618558a6be9#:~:text=,them%20stronger%2C%20normal%2C%20or%20weaker)) ([Difficulty Balancing | Notion](https://wmelongames.notion.site/Difficulty-Balancing-08ea6e1f9a6e4d43a9e58618558a6be9#:~:text=To%20%E2%80%9Cexplain%E2%80%9D%20the%20game%20how,want%20the%20player%20to%20be)). 즉, "Required Upg" 대비 낮을 때 얼마나 강해지는지 등은 템플릿이 계산하므로, 개발자는 기본 능력과 각 레벨의 Required Upg만 신경 쓰면 됩니다. 또한 적의 **무기 장착 위치**나 **발사 패턴**을 바꾸고 싶다면 적 프리팹의 자식 오브젝트 (총구 위치 등)와 Weapon/Bullet 설정을 조정합니다 ([Enemies | Notion](https://wmelongames.notion.site/Enemies-cc061ab4daf04a05ab9c1ef6a3631716#:~:text=How%20to%20replace%20the%20existing,enemy%20model)). 문서에 따르면 "How to adjust enemy weapon position" 부분에서 이러한 세부사항을 다루고 있으니 참고하면 좋습니다 ([Enemies | Notion](https://wmelongames.notion.site/Enemies-cc061ab4daf04a05ab9c1ef6a3631716#:~:text=How%20to%20replace%20the%20existing,enemy%20model)).
* **장애물/환경 오브젝트 추가**: 새로운 **장애물**이나 **환경** 오브젝트를 추가하는 방법은 앞서 레벨 디자인 단계에서 다룬 바와 같습니다. 핵심은 **프리팹 준비**와 **Level Editor 등록**의 두 단계입니다. 예를 들어 새로운 장식 오브젝트(환경 요소)를 추가한다고 하면:
  1. **Prefab 준비**: 3D 모델 프리팹을 만들고, Collider를 추가합니다 (없을 경우 Box, Sphere 등 적절한 형태로) ([Obstacles | Notion](https://wmelongames.notion.site/Obstacles-bc6922b59fff4090ab9eb1ab35d40ff2#:~:text=1,to%20turn%20into%20an%20obstacle)). 그리고 그 오브젝트에 **Obstacle 레이어**를 지정합니다 (Ground로 동작해야 하는 바닥 오브젝트라면 Ground 레이어로 설정) ([Environment | Notion](https://wmelongames.notion.site/Environment-991c663dc33443659795e9d6be81a6b0#:~:text=3,prevent%20aiming%20through%20the%20obstacles)). 필요 시 NavMeshObstacle 컴포넌트를 붙여서 경로 차단 역할을 부여합니다 ([Obstacles | Notion](https://wmelongames.notion.site/Obstacles-bc6922b59fff4090ab9eb1ab35d40ff2#:~:text=4,or%20NPC%20to%20go%20through)) ([Environment | Notion](https://wmelongames.notion.site/Environment-991c663dc33443659795e9d6be81a6b0#:~:text=3,prevent%20aiming%20through%20the%20obstacles)).
  2. **Level Editor 등록**: 프로젝트 창에서 해당 프리팹을 우클릭하고 **Add into Level Editor**를 선택합니다. **World**는 이 오브젝트를 사용할 월드를 고르고, **Type**은 장애물이라면 "Obstacle", 배경 장식이라면 "Environment" 등으로 선택합니다 ([Environment | Notion](https://wmelongames.notion.site/Environment-991c663dc33443659795e9d6be81a6b0#:~:text=1.%20Right,Add%20into%20Level%20Editor)). 그리고 Add를 누릅니다. 만약 필요한 컴포넌트나 설정이 빠졌다면 에러 메시지가 나타나므로 지시에 따라 보완합니다 ([Obstacles | Notion](https://wmelongames.notion.site/Obstacles-bc6922b59fff4090ab9eb1ab35d40ff2#:~:text=3,will%20get%20an%20error%20message)). 완료되면 월드 설정의 해당 카테고리에 새로운 오브젝트가 추가되고, 이제 레벨 디자인 시 사용할 수 있게 됩니다 ([Obstacles | Notion](https://wmelongames.notion.site/Obstacles-bc6922b59fff4090ab9eb1ab35d40ff2#:~:text=Image%3A%20Untitled)).

기존 제공 오브젝트들 역시 이러한 데이터베이스에 등록되어 있으며, **월드별 전용 폴더**에 정리되어 있습니다 (예: World\_01 전용 장애물 폴더 등) ([Obstacles | Notion](https://wmelongames.notion.site/Obstacles-bc6922b59fff4090ab9eb1ab35d40ff2#:~:text=5,are%20stored%20in%20dedicated%20folders)). 참고로, Obstacle과 Environment 타입 구분은 레벨 에디터 내 탭/필터링 용도로 쓰이며 동작상의 큰 차이는 없습니다 ([Obstacles | Notion](https://wmelongames.notion.site/Obstacles-bc6922b59fff4090ab9eb1ab35d40ff2#:~:text=2,Add)) ([Environment | Notion](https://wmelongames.notion.site/Environment-991c663dc33443659795e9d6be81a6b0#:~:text=2,Add)).

* **아이템/파워업 배치**: 템플릿에 **드랍 아이템**이나 **파워업** 오브젝트가 있다면 (예: 체력 회복 아이템, 일시 무적 등), 이를 배치하거나 추가할 수도 있습니다. 이러한 오브젝트 역시 Level Editor에 등록하여 사용하거나, 별도의 스폰 논리로 동작할 수 있습니다. 만약 존재하지 않지만 구현하고자 한다면, 새로운 프리팹을 만들고 위와 같은 절차로 등록한 뒤, **게임 플레이 중 특정 조건에 생성**되도록 GameManager나 Enemy 죽음 로직 등에 코드를 추가해야 합니다. 이는 고급 커스터마이즈에 속하므로, 필요 시 템플릿 구조를 분석하여 구현합니다.

**6. 광고 및 IAP 설정 (Ads & IAP Setup)**

이 단계에서는 **Monetization(수익화) 모듈**을 실제 퍼블리싱할 게임에 맞게 설정하는 방법을 설명합니다. 템플릿에는 광고와 인앱결제가 통합되어 있지만, 이를 사용하려면 각 플랫폼별로 약간의 설정이 필요합니다.

* **광고 네트워크 설정**: 프로젝트 창에서 **Project Files/Data/Monetization Settings** 에셋을 찾습니다. 이 파일은 광고/결제 관련 설정을 모두 담고 있습니다 ([Monetization (Ads & IAP) | Notion](https://wmelongames.notion.site/Monetization-Ads-IAP-13bc0f2fe155801f9db2d7b4f0e43b74#:~:text=)). 인스펙터에서 열어보면 **Mobile Monetization 활성화 토글**, **지원 광고 네트워크 옵션 (AdMob, Unity Ads, LevelPlay 등)**, 그리고 각 네트워크별 **광고 단위 ID 입력 필드**가 있을 것입니다 ([Monetization (Ads & IAP) | Notion](https://wmelongames.notion.site/Monetization-Ads-IAP-13bc0f2fe155801f9db2d7b4f0e43b74#:~:text=,developers%20looking%20to%20generate%20revenue)) ([Monetization (Ads & IAP) | Notion](https://wmelongames.notion.site/Monetization-Ads-IAP-13bc0f2fe155801f9db2d7b4f0e43b74#:~:text=)). 우선 사용할 광고 네트워크를 고릅니다:
  + *Unity Ads*: Unity에서 기본 제공하는 광고 네트워크로, **Unity Services**에 프로젝트를 등록하고 **Game ID**를 발급받아야 합니다. Monetization Settings의 Unity Ads 섹션에 해당 Game ID와 Placement ID (배너, 전면, 리워드)들을 입력하세요. 또한 **Services** 창이나 **프로젝트 설정**에서 Ads가 활성화되어야 합니다.
  + *AdMob*: Google AdMob을 사용하려면 **Google Mobile Ads SDK**를 프로젝트에 추가해야 합니다. (일부 템플릿에는 패키지가 포함되어 있을 수도 있습니다. 없다면 Google이 제공하는 Unity 패키지를 임포트합니다.) 그리고 Monetization Settings에서 AdMob을 활성화하고, **AdMob App ID**와 각 광고 형식별 \*\*광고 단위 ID(ad unit ID)\*\*를 입력합니다. Android와 iOS ID를 각각 지정할 수 있을 것입니다. AdMob의 경우 AndroidManifest나 plist 설정도 필요할 수 있으므로, Google 공식 문서를 참고해 세팅합니다.
  + *LevelPlay (아이언소스)*: Unity LevelPlay(구 ironSource)를 사용할 경우, 역시 설정 창에 App Key 등을 입력하는 란이 있을 것입니다. LevelPlay를 사용하면 AdMob이나 Unity Ads 등을 **중재(Mediation)** 형태로 함께 쓸 수 있으므로 고급 옵션입니다. 필요 시 Watermelon Games 문서의 **“Add your custom”** 부분을 참고하여 통합합니다 ([Squad Shooter | Template + Editor | Packs | Unity Assets - Codeintra](https://www.codeintra.com/items/1143/squad-shooter-template-editor-packs-unity-assets?srsltid=AfmBOorIJ3ALhSAcenBUZ1cBiU0tBoe7z0iznulZST8BYoAYzDXnaCb-#:~:text=%E2%80%8E%20%E2%80%8E%20%E2%80%8E%20%E2%80%8E%20%F0%9F%94%B8,Unity%20Ads)).
  + 그 외 네트워크: 템플릿은 **추가 커스텀 광고 네트워크**도 지원하도록 설계되었으나, 기본적으로 상기 3가지를 통해 커버됩니다 ([Squad Shooter | Template + Editor | Packs | Unity Assets - Codeintra](https://www.codeintra.com/items/1143/squad-shooter-template-editor-packs-unity-assets?srsltid=AfmBOorIJ3ALhSAcenBUZ1cBiU0tBoe7z0iznulZST8BYoAYzDXnaCb-#:~:text=%E2%80%8E%20%E2%80%8E%20%E2%80%8E%20%E2%80%8E%20%F0%9F%94%B8,Unity%20Ads)).

필요한 광고 ID들을 입력한 후 **테스트 모드**로 게임을 실행하면 (또는 실제 기기 빌드), 배너나 전면 광고, 리워드 광고가 호출되는지 확인합니다. UI Store 등에서 **“Watch Ad”** 버튼이 리워드 광고를 호출하며, 광고 시청 완료 후 보상이 제대로 지급되는지 (코인 증가 등) 테스트하세요. 광고가 나타나지 않으면 설정 파일에 ID가 제대로 들어갔는지, 해당 네트워크 SDK가 포함되었는지 확인해야 합니다.

* **인앱 구매(IAP) 설정**: 인앱 구매를 활성화하려면 Unity IAP 패키지를 설정해야 합니다. Watermelon Games는 **IAP Quick Start 가이드**를 통해 이를 안내하고 있습니다 ([UI Store | Notion](https://wmelongames.notion.site/UI-Store-18bc0f2fe155808193e4d5d7fd72bdfd#:~:text=,IAP%20Purchases%20in%20the%20project)). 일반적인 절차는 다음과 같습니다:
  + **서비스 설정**: Unity 에디터의 Services (혹은 프로젝트 설정의 Services 탭)에서 \*\*In-App Purchasing (UDP가 아닌 Unity IAP)\*\*를 활성화합니다. 필요한 패키지가 없다면 자동으로 임포트합니다.
  + **제품 정의**: 판매할 아이템들을 Unity IAP Catalog 또는 각 스토어의 Developer Console에 등록합니다. 예를 들어 Google Play 콘솔과 Apple App Store Connect에 **상품 ID**를 생성하세요. 상품 ID는 템플릿에서 사용중인 것과 일치시켜야 합니다. 기본적으로 템플릿은 "no\_ads", "coin\_pack1" 등 몇 가지 상품 ID를 사용하고 있을 것입니다 (예: No Ads 상품의 ID가 no\_ads로 코드에 정의됨).
  + **Monetization Settings 확인**: 앞서 열린 Monetization Settings 에셋에서 IAP 관련 설정을 찾습니다. 아마 **상품 목록**이나 **리워드 종류** 정의가 있을 수 있습니다. 템플릿 문서에 따르면, 기본적으로 **No Ads**, **Currency Reward(코인)**, **Starter Pack**의 세 가지 **Reward** 타입을 구현했다고 합니다 ([UI Store | Notion](https://wmelongames.notion.site/UI-Store-18bc0f2fe155808193e4d5d7fd72bdfd#:~:text=Rewards%20Implemented%20in%20this%20Template)). 이와 실제 IAP 상품을 연결해야 합니다. 예를 들어 No Ads는 비영구 소비형 상품이므로 구매 시 광고를 제거하고 버튼을 숨기는 로직이 이미 구현되어 있습니다 ([UI Store | Notion](https://wmelongames.notion.site/UI-Store-18bc0f2fe155808193e4d5d7fd72bdfd#:~:text=1,currency%20and%20no%20ads%20rewards)). Currency 구매는 구매 즉시 해당 양의 코인을 통화로 추가하는 형태입니다.
  + **코드 연결**: 템플릿의 IAP 매니저 (예: IAPManager 스크립트)가 Unity IAP API를 통해 구매 처리를 하고 있을 것입니다. 여기서 기본 상품 ID와 위에서 등록한 ID가 일치해야 하므로 확인합니다. 일반적으로 Watermelon Games 템플릿에서는 **제품 ID 문자열을 한 곳에서 관리**하도록 해두었을 가능성이 높습니다 (예: Constants나 ScriptableObject로). 이를 맞추면 별도 코드 수정 없이도 IAP가 동작합니다.
  + **테스트**: 에디터의 **IAP Simulation** 기능이나 각 플랫폼의 샌드박스 테스트를 통해 구매 플로우를 시험합니다. 예를 들어 에디터에서 Unity IAP Simulator를 켜고 No Ads 상품을 구매해보면, 결제 성공 콜백 후 **게임 내 Banner/Interstitial 광고가 사라지고**, 상점 UI에서 No Ads 상품이 "구매 완료"로 비활성화되는지 확인합니다 ([Monetization (Ads & IAP) | Notion](https://wmelongames.notion.site/Monetization-Ads-IAP-13bc0f2fe155801f9db2d7b4f0e43b74#:~:text=Image%3A%20image)) ([Monetization (Ads & IAP) | Notion](https://wmelongames.notion.site/Monetization-Ads-IAP-13bc0f2fe155801f9db2d7b4f0e43b74#:~:text=Important%3A%20In%20most%20cases%2C%20if,monetization%2C%20you%20will%20need%20to)). 코인 상품의 경우 코인 증가를 체크합니다. 모든 플로우가 문제 없으면 인앱 구매 설정 완료입니다.
* **보상 시스템 연동**: 광고 시청 및 IAP 구매에 따른 \*\*보상(Rewards)\*\*은 템플릿에서 자동 처리됩니다. 예를 들어 **리워드 광고**를 시청하면 UI Store에 정의된 **Currency Reward**로서 특정량의 코인을 지급하도록 설정되어 있습니다 ([UI Store | Notion](https://wmelongames.notion.site/UI-Store-18bc0f2fe155808193e4d5d7fd72bdfd#:~:text=Rewards%20Implemented%20in%20this%20Template)). 이러한 보상 종류 (Currency, NoAds, StarterPack 등)는 UI Store 시스템과 Monetization 모듈이 연계되어 동작합니다 ([UI Store | Notion](https://wmelongames.notion.site/UI-Store-18bc0f2fe155808193e4d5d7fd72bdfd#:~:text=The%20UI%20Store%20displays%20various,Module%20and%20the%20Rewards%20System)). 개발자는 필요한 경우 **새로운 보상 상품**을 추가할 수 있는데, 이를 위해서는 UI Store에 상품 추가 + Monetization 보상 종류 추가 + (옵션) IAP 설정 추가 의 단계를 모두 해주어야 합니다 ([UI Store | Notion](https://wmelongames.notion.site/UI-Store-18bc0f2fe155808193e4d5d7fd72bdfd#:~:text=Rewards%20Implemented%20in%20this%20Template)) ([UI Store | Notion](https://wmelongames.notion.site/UI-Store-18bc0f2fe155808193e4d5d7fd72bdfd#:~:text=More%20info%20about%20reward%20types)). 예를 들어 "Premium Gun" 같은 영구 아이템을 판매하고 싶다면, 게임 내 해당 무기를 잠금해제하는 로직을 추가하고, UI Store에 상품을 만들고, Monetization Settings에 새로운 Reward 타입을 정의하여 IAP 구매 시 그 아이템을 언락하도록 코드를 작성해야 합니다.

이로써 광고와 인앱 구매의 설정이 완료되었습니다. 요약하면, **Monetization Settings** 자산에서 손쉽게 광고/IAP를 켜고 끌 수 있고, 필요한 ID만 넣으면 별도 코드 작업 없이 **바로 수익화 기능을 활용**할 수 있다는 것이 큰 장점입니다 ([Monetization (Ads & IAP) | Notion](https://wmelongames.notion.site/Monetization-Ads-IAP-13bc0f2fe155801f9db2d7b4f0e43b74#:~:text=,developers%20looking%20to%20generate%20revenue)).

**Tip:** 개발 중에 모든 광고를 일시적으로 끄고 싶다면 Monetization Settings에서 **Mobile Monetization** 토글을 해제하면 됩니다 ([Monetization (Ads & IAP) | Notion](https://wmelongames.notion.site/Monetization-Ads-IAP-13bc0f2fe155801f9db2d7b4f0e43b74#:~:text=You%20can%20disable%20the%20module,and%20IAPs%20won%E2%80%99t%20be%20available)). 그러면 광고 모듈이 초기화되지 않고, 게임 내 광고 관련 버튼들도 누를 시 동작하지 않게 됩니다. 배너 자리 등이 비어있게 되므로 완전히 제거하려면 UI 요소 자체를 삭제해야 합니다 ([Monetization (Ads & IAP) | Notion](https://wmelongames.notion.site/Monetization-Ads-IAP-13bc0f2fe155801f9db2d7b4f0e43b74#:~:text=Image%3A%20image)).

**7. 빌드 및 배포 (Build & Deployment)**

게임 개발을 마쳤다면, 실제 기기에서 게임을 구동할 수 있도록 프로젝트를 빌드합니다.

* **플랫폼 전환**: 우선 Unity **Build Settings**에서 대상 플랫폼을 선택합니다. 모바일 배포를 위해 **Android**와 **iOS** 플랫폼으로 전환하고, 필요한 모듈(Android SDK, iOS Xcode 프로젝트)을 설정합니다. (PC/WebGL 배포도 가능하지만, 광고/IAP는 모바일에 한정되므로 모바일 테스트를 권장합니다.)
* **Android 빌드 설정**: Android의 경우 **Player Settings**에서 패키지명, 최소 SDK 버전 등을 설정합니다. AdMob을 사용했다면 **AndroidManifest**에 AdMob App ID를 추가해야 할 수 있습니다 (GoogleMobileAdsSettings에서 자동으로 추가 가능). IL2CPP + .NET 4.x로 설정하고, ARM64 체크 등 요즘 스토어 요구에 맞춥니다. 또한 Unity Ads/LevelPlay를 쓴다면 **Internet permission** 등이 자동 설정됩니다.
* **iOS 빌드 설정**: iOS는 Xcode로 내보낸 뒤 Apple 개발자 설정(앱 ID, 프로비저닝 등)을 해야 합니다. IAP를 쓴다면 **Capabilities**에서 In-App Purchase를 켜고, Advertising ID 접근을 위해 **Tracking Usage Description**도 넣어야 할 수 있습니다. Unity에서 Xcode 프로젝트를 만들 때 이 부분을 확인합니다.
* **최적화**: 모바일에서 원활히 돌아가도록 품질 설정이나 불필요한 디버그 로그를 정리합니다. Squad Shooter 템플릿은 모바일 최적화를 염두에 두고 설계되어 비교적 가볍지만, 만약 추가한 콘텐츠로 인해 성능이 떨어졌다면 **Occlusion Culling**이나 **LOD**, 품질 설정 등을 조정합니다. 또, **IL2CPP 빌드**를 통해 성능과 보안 향상을 기대할 수 있습니다.
* **빌드**: **File > Build Settings**에서 선택한 플랫폼에 대해 **Build** 혹은 **Build and Run**을 수행합니다. APK (Android)나 Xcode 프로젝트(iOS)가 생성될 것입니다. Android APK를 기기에 설치하거나, iOS는 Xcode에서 빌드하여 테스트 기기에 설치합니다. 초기 실행 시 모든 시스템(광고, IAP 등)이 잘 동작하는지, 해상도 및 화면비율에 UI가 잘 맞는지 확인합니다.
* **테스트**: 여러 기기에서 플레이 테스트를 진행합니다. **광고 노출 테스트** (테스트 모드 광고로), **인앱 구매 테스트** (샌드박스 계정으로) 등을 수행하고, **리더보드나 푸시 같은 부가 기능**을 붙였으면 함께 점검합니다. 몬스터가 의도한 대로 강해지는지, 업그레이드 진행이 적절한지 등도 실제 플레이하면서 최종 밸런싱 피드백을 얻습니다.
* **배포 준비**: 스토어에 출시하기 전 필요한 에셋 (아이콘, 스크린샷, 프로모션 이미지) 작업과, **패키지 서명**(Android), **앱 심사**(iOS)를 준비합니다. 템플릿에는 없지만 **앱 인증서**나 **번들 ID 설정**은 잊지 말아야 합니다.

이로써 Squad Shooter 템플릿을 활용한 게임 개발의 전 과정을 마치고, 게임을 릴리스할 준비가 완료되었습니다. 전체적인 과정 동안 템플릿이 제공하는 구조와 도구를 적극 활용함으로써 개발 시간을 크게 단축하고 안정적인 결과물을 얻을 수 있습니다.

**코드 구조 및 설명**

Squad Shooter 템플릿은 **깨끗한 폴더 구조와 모듈화된 코드**로 구성되어 있어, 개발자가 프로젝트를 이해하고 확장하기 용이합니다 ([Squad Shooter | Template + Editor by Watermelon Games](https://wmelongames.itch.io/squad-shooter-template-editor#:~:text=Customization%20Guides)). 이번 장에서는 프로젝트의 주요 폴더 구성과 핵심 스크립트의 역할, 그리고 게임 실행 흐름에 대해 설명합니다.

**폴더 구조 (주요 디렉토리)**

프로젝트 임포트 후 **Project Window**에서 확인할 수 있는 주요 폴더는 다음과 같습니다:

| **폴더 경로** | **내용 및 역할** |
| --- | --- |
| **Project Files/Game/Scenes** | 게임 씬 파일들이 위치한 폴더입니다. 기본으로 Game.unity (메인 게임 씬)가 포함되며, 필요에 따라 추가 씬(예: Tutorial.unity)을 둘 수도 있습니다 ([Overview |
| **Project Files/Game/Prefabs** | 게임에서 사용하는 주요 프리팹(prefab) 리소스들이 모여 있습니다. 하위에 **Characters**, **Enemies**, **Obstacles**, **Environment**, **Weapons** 등의 폴더로 세분화되어 있고, 각각 플레이어 캐릭터 프리팹, 적 AI 프리팹, 장애물/환경 프리팹, 무기 및 탄환 프리팹 등이 저장됩니다. 또한 **World별로 프리팹을 구분**하여, 예를 들어 Prefabs/Obstacles/World\_Desert/… 식으로 월드 테마별 장애물을 관리합니다 ([Obstacles |
| **Project Files/Game/Scripts** | 게임 논리를 담당하는 C# 스크립트 파일들이 위치합니다. 코드는 기능별로 구성되어 있어 가독성이 좋습니다 ([Squad Shooter |
| **Project Files/Data/Characters System** | **캐릭터 데이터** 관련 자산들이 모여 있습니다. **Character Database.asset**가 여기에 포함되며, 플레이어 캐릭터 및 스킨들의 목록과 unlock 조건 레벨 등이 이곳에 정의됩니다 ([Experience System |
| **Project Files/Data/Weapon System** | **무기 및 탄환 데이터** 관련 자산 폴더입니다. 무기 종류별 속성, 업그레이드 단계 정보, 탄환의 종류와 이펙트 등이 정의된 ScriptableObject들이 있습니다. 예를 들어 **Weapon Database**나 각 무기별 **데이터 에셋**(데미지, 연사속도, 탄창 크기 등)과 **업그레이드 비용 테이블**이 포함됩니다. 또한 탄환 프리팹 레퍼런스, 이펙트 프리팹 레퍼런스 등도 이 데이터 에셋들이 들고 있을 수 있습니다. |
| **Project Files/Data/Enemies** (또는 Enemy System) | **적 AI 데이터**를 담은 폴더입니다. 적 유형별 능력치 (체력, 공격력, AI 속성)를 정의한 ScriptableObject나 **Enemy Database**가 위치합니다. 적의 종류가 많다면 체계적으로 관리하기 위해 데이터화해 두었을 것입니다. 난이도 계수나 보상(죽을 때 드롭 코인 양) 등의 정보도 포함될 수 있습니다. |
| **Project Files/Data/Experience System** | **경험치 및 레벨링 시스템** 데이터가 있는 폴더입니다. **Experience Database.asset**가 핵심으로, 플레이어 레벨 업에 필요한 누적 경험치 값을 배열로 가지고 있습니다 ([Experience System |
| **Project Files/Data/Currency & Store** | 게임 내 **통화 및 상점** 관련 데이터 폴더입니다. 코인 등의 재화 설정과, 상점(UI Store)에 표시될 상품 목록과 속성이 정의됩니다. 예를 들어 **Product Database**나 **Store Items.asset**에 각 판매 상품의 이름, 종류(코인팩, NoAds 등), 가격(화폐 또는 필요 코인)이 기록되어 있습니다. 또한 **Rewards System** 관련 설정도 이쪽에 있거나 Monetization 폴더에 연계되어 있습니다 ([UI Store |
| **Project Files/Data/Monetization Settings** | **수익화 설정 파일**로, 광고와 IAP 전반의 설정을 포괄합니다. Monetization Settings.asset를 인스펙터에서 보면 광고 SDK 선택 및 IDs, IAP 제품 ID 등이 모두 포함돼 있어 **원스톱으로 수익화 설정**을 관리합니다 ([Monetization (Ads & IAP) |
| **Project Files/Art/**, **Audio/**, **Effects/** 등 | 게임의 시각/청각 리소스들이 저장된 폴더입니다. 예를 들어 2D UI 스프라이트, 아이콘 이미지는 Art/UI 폴더에, 3D 모델 및 텍스처는 Art/3D에, 사운드 이펙트와 배경음악은 Audio 폴더에, 총알 발사 파티클 등의 이펙트 프리팹은 Effects 폴더에 있을 것입니다. 이런 에셋 폴더들은 템플릿 공통으로 사용되며, 필요시 추가 리소스를 여기에 넣어 관리합니다. |

이 외에도 **Plugins** 폴더(예: 광고/IAP SDK), **Editor** 폴더(에디터 확장 스크립트, Level Editor 관련 코드), **Resources** 폴더(런타임 로드용 에셋) 등이 있을 수 있습니다. 예컨대 Level Editor가 내부적으로 **룸 프리셋**을 저장할 때 Resources를 사용할 수도 있고, Localization 기능이 있다면 언어별 리소스가 있을 수도 있습니다. 전반적으로 **폴더 구조가 논리적으로 구분**되어 있어, 각 시스템별 자원을 한눈에 찾아볼 수 있습니다 ([Squad Shooter | Template + Editor by Watermelon Games](https://wmelongames.itch.io/squad-shooter-template-editor#:~:text=Customization%20Guides)).

**핵심 스크립트와 게임 로직 흐름**

Squad Shooter 템플릿의 코드는 모듈화되어 있으나, 전형적인 게임 루프 구조를 갖습니다. 주요 스크립트와 게임 진행 흐름을 살펴보면 다음과 같습니다:

* **GameManager (또는 LevelManager)**: 이 스크립트는 게임 전체 흐름을 관리하는 **매니저 클래스**입니다. 싱글톤으로 구현되어 있을 가능성이 높으며, 게임 시작 시 초기화 작업을 합니다. 예를 들어 **첫 월드와 첫 레벨 로드**, 플레이어 캐릭터 생성, UI 초기 세팅 등을 수행합니다. GameManager는 각 레벨의 종료 조건을 체크하여 **레벨 완료** 또는 **게임 오버** 시 결과 처리(경험치, 코인 부여 등)를 하고 다음 단계(예: 다음 레벨 로드 또는 메인 메뉴로 복귀)를 결정합니다. 또한 일시정지 제어, 전역 게임 상태(Playing, Paused, GameOver 등) 관리도 담당합니다.
* **PlayerController (또는 PlayerMovement, PlayerCombat)**: 플레이어 캐릭터에 부착된 스크립트로, **이동 및 공격 제어**를 담당합니다. 입력(Input)을 처리하여 캐릭터를 움직이고 회전시키며, 공격 버튼 입력 시 총을 발사합니다. 모바일 환경에서는 조이스틱/버튼 UI와 연동되고, PC에서는 키보드/마우스로 제어될 것입니다. 이 컨트롤러는 **애니메이터**와 연계되어 움직임에 따라 달리는 애니메이션을 재생하거나, 공격 시 공격 애니메이션을 재생합니다. 또한 플레이어의 체력 및 피격 판정, 무적 시간 처리 등의 로직도 포함될 수 있습니다. (Template이 **자동 조준**이나 **자동 발사** 모드를 제공한다면, 이 스크립트에서 그 로직을 구현했을 것입니다).
* **WeaponController & Bullet**: 플레이어 (또는 적)의 무기를 관리하는 스크립트입니다. 총알 발사 지점(총구)에서 **Bullet 프리팹**을 생성하여 발사하며, 연사 속도, 탄환 속도를 제어합니다. Bullet 스크립트는 발사 후 직선 이동하거나 일정 시간이 지나면 사라지고, 충돌 시 피해를 주고 파괴되는 역할을 합니다. 무기 변경 기능이 있을 경우 WeaponController는 현재 장착 무기를 바꾸는 메서드를 갖고 있으며, 플레이어 입력이나 게임 이벤트로 호출됩니다. 예를 들어 무기 교체 UI를 눌렀을 때 PlayerController를 통해 WeaponController의 SwitchWeapon(int weaponId)가 호출되는 식입니다. **탄창/재장전** 개념이 있다면 탄약 수를 관리하고 HUD에 업데이트하는 부분도 해당 스크립트에서 담당합니다.
* **EnemyController (또는 AIController)**: 적 캐릭터마다 붙어있는 **AI 제어 스크립트**입니다. 이 스크립트는 NavMeshAgent를 이용해 플레이어를 추적하고 이동하며 ([Obstacles | Notion](https://wmelongames.notion.site/Obstacles-bc6922b59fff4090ab9eb1ab35d40ff2#:~:text=4,or%20NPC%20to%20go%20through)), 사정거리 안에 들어오면 공격(총알 발사 혹은 근접공격)을 수행합니다. 행동 패턴은 적 종류에 따라 다르게 구성되는데, 이를 한 클래스 내 분기 처리하거나, 적 타입별로 상속 구조를 가졌을 수 있습니다 (예: MeleeEnemy : EnemyController, RangedEnemy : EnemyController 등). EnemyController는 **체력 관리**도 포함하며, 체력이 0이 되면 사망 처리를 합니다. 죽을 때 Experience나 코인을 드롭하는 연출이 있다면 GameManager나 별도 DropManager를 통해 처리하지만, 수치를 결정하는 것은 EnemyController/데이터에 있을 것입니다. 또한 적이 플레이어를 발견하는 범위, 시야각 등이 설정되어 있고, 아군 NPC나 장애물과 상호작용(피해입기, 길막힘 등)을 처리합니다.
* **SpawnManager/WaveManager**: 특정 레벨에서 다수의 적을 한꺼번에 또는 순차적으로 생성하는 로직이 존재합니다. Level 데이터에 **웨이브 정보**가 있다면, SpawnManager가 이를 읽어들여 시간차로 적을 소환하거나, 방 이동 시 다음 웨이브를 활성화합니다. 예컨대 "방1: 적 5마리 즉시 스폰, 방2: 처음엔 3마리 스폰 후 5초 뒤 2마리 추가" 같은 패턴을 구현할 수 있습니다. SpawnManager는 GameManager와 연계되어 레벨이 시작될 때 웨이브를 세팅하고, 모든 웨이브 종료시 레벨 클리어를 GameManager에 알립니다.
* **UIManager**: 여러 UI 캔버스를 관리하는 스크립트로, UI 간 전환과 업데이트를 담당합니다. 예를 들어 게임이 시작되면 MainMenu UI를 비활성화하고 HUD를 표시하거나, 일시정지 시 Pause 메뉴 패널을 활성화하는 식입니다. 또한 **플레이어 상태에 따른 HUD 갱신**(체력바, 탄약 수 등)과 **획득한 코인/XP 표시 업데이트**, **레벨 클리어 시 결과 팝업에 데이터 채우기** 등을 합니다. UI 버튼 이벤트(OnClick)는 이 매니저에 연결되어 있어서, 상점 버튼을 누르면 Store UI를 띄우고 게임을 일시정지시키는 등의 로직을 수행합니다. UIManager는 MonetizationManager와도 연동되어, 예를 들어 "Watch Ad for Reward" 버튼 누르면 MonetizationManager를 호출하고 그 결과를 받아와 UI를 업데이트합니다.
* **MonetizationManager**: 광고 및 인앱결제를 실제 처리하는 전용 스크립트입니다. 초기화 시 선택된 광고 SDK들을 불러오고, **광고 이벤트 콜백**과 **보상 지급 로직**을 설정합니다 ([Monetization (Ads & IAP) | Notion](https://wmelongames.notion.site/Monetization-Ads-IAP-13bc0f2fe155801f9db2d7b4f0e43b74#:~:text=,developers%20looking%20to%20generate%20revenue)). 이 매니저는 UIManager나 GameManager로부터 광고/결제 요청을 받아 해당 함수를 실행합니다. 예를 들어 UIStore에서 "코인 x500 획득 광고보기"를 누르면 MonetizationManager의 ShowRewardedAd("CoinReward")를 호출하고, 광고 완료 콜백이 오면 GiveReward("CoinReward")를 실행하여 코인을 지급합니다 ([UI Store | Notion](https://wmelongames.notion.site/UI-Store-18bc0f2fe155808193e4d5d7fd72bdfd#:~:text=The%20UI%20Store%20displays%20various,Module%20and%20the%20Rewards%20System)). IAP의 경우 Unity IAP의 PurchaseSucceeded 이벤트에서 OnPurchaseComplete(productId)를 호출하며, 여기서 제품 ID별로 처리 (예: "no\_ads" 구매 시 광고제거 활성화, "coin\_pack1" 구매 시 코인 증가)를 수행합니다. 이러한 구현을 통해 **코드 한 곳에서 광고/IAP와 게임 데이터를 연결**하고 있어 유지보수가 쉽습니다.
* **데이터 관리 및 기타**: 그 밖에 **UpgradeManager**나 **SaveManager**가 있을 수 있습니다. UpgradeManager는 플레이어의 무기/캐릭터 업그레이드를 적용하고 관리하는 클래스입니다. 예컨대 플레이어가 업그레이드 메뉴에서 무기 공격력을 올리면 Weapon 데이터의 현재 레벨을 증가시키고, 코인을 차감하며, 그 정보를 SaveManager를 통해 영구 저장할 것입니다. SaveManager(또는 GameSave)는 플레이어 진행 데이터를 저장/불러오기 위한 것으로, 현재 플레이어 레벨, 획득 코인, 업그레이드 상태, 광고 제거 구매여부 등을 보관합니다. PlayerPrefs나 JSON 직렬화를 사용할 수 있고, 클라우드 연동은 별도 구현사항입니다.

**게임 실행 흐름**: 위 스크립트들이 어떻게 상호작용하며 게임이 돌아가는지 요약하면 다음과 같습니다:

1. **초기화** – Game 씬이 로드되면, GameManager가 시작됩니다. MonetizationManager가 초기화되어 광고/IAP를 준비하고, UIManager가 메인 메뉴를 표시합니다. 이때 GameManager는 기본 World와 Level을 설정하지만, 메인 메뉴 상태에서 게임을 대기시킵니다.
2. **게임 시작** – 플레이어가 메인 메뉴에서 Play를 누르면, GameManager는 현재 월드의 1레벨 데이터를 로드하고, 해당 레벨에 맞게 **환경과 장애물 배치**를 설정합니다 (이미 에디터에서 디자인된 레벨이라면 바로 사용하거나, 동적으로 생성될 부분이 있으면 여기서 생성) ([World System | Notion](https://wmelongames.notion.site/World-System-a719e8ab7f5a446c8e25d4e35c687c80#:~:text=Down%20below%20you%20will%20see,related%20to%20the%20selected%20world)). 플레이어 캐릭터(프리팹)가 스폰되고 위치가 초기화됩니다. HUD가 활성화되고, 게임이 진행 상태로 전환됩니다.
3. **레벨 진행** – 이제 EnemyController 및 SpawnManager에 의해 적들이 등장하고 움직입니다. 플레이어는 PlayerController로 캐릭터를 조작하여 적을 처치합니다. EnemyController는 적 사망 시 GameManager에 적 카운트를 보고하고, 모든 적 처치 시 SpawnManager/GameManager는 **방 클리어** 처리를 합니다. 만약 여러 방이 있는 레벨이라면, 다음 방의 게이트를 열고 플레이어가 이동하면 그 시점에 새로운 웨이브를 스폰합니다. 이렇게 **모든 방의 적을 처치하면 레벨 클리어** 상태가 됩니다.
4. **레벨 완료** – GameManager는 레벨 클리어를 감지하면 HUD를 숨기고 **Level Complete UI**를 띄웁니다. 이 UI에는 획득한 코인, 경험치, 별 점수 등이 표시되고 다음 단계로 진행 가능한 버튼이 나타납니다. 동시에 GameManager는 **경험치와 코인 보상**을 계산하여 플레이어 데이터에 추가합니다 (경험치 추가 후 Experience System에서 레벨업 여부를 체크하여 캐릭터 레벨을 올리고, 새 스킨이 언락되면 UI에 표시) ([Experience System | Notion](https://wmelongames.notion.site/Experience-System-f88c15e9f92d4cf8a0dea0b2a5035ecd#:~:text=,player%20skin%20in%20our%20case)). 코인은 SaveManager를 통해 저장되고, UIManager는 메인 메뉴에 현재 코인/레벨을 갱신합니다.
5. **다음 레벨/월드** – 플레이어가 계속하기(Next) 버튼을 누르면 GameManager는 다음 레벨 데이터를 로드하고 위 과정(3)을 반복합니다. 월드의 마지막 레벨을 클리어했다면 월드2를 해금하고, 월드 선택 UI 등을 통해 다음 월드로 이동할 수 있게 할 것입니다. 또는 다음 월드 첫 레벨을 바로 시작할 수도 있습니다.
6. **게임 종료 조건** – 플레이어 체력이 0이 되거나 목표를 달성하지 못하면 GameManager는 게임 오버를 처리합니다. 모든 적 처치 전에 사망하면 **Game Over UI**를 표시하고 재시도 옵션을 줍니다. 재시도 시 해당 레벨을 처음부터 다시 시작하거나, 메인 메뉴로 돌아갈 수 있습니다.
7. **상점 및 업그레이드** – 플레이어는 메인 메뉴에서 상점(UI Store) 버튼을 눌러 **IAP 구매**나 **광고 시청 보상**을 받을 수 있습니다. UIManager는 Store 패널을 활성화하고 MonetizationManager와 연결된 구매 버튼들을 보여줍니다. 예를 들어 "Watch Ad – +100 Coins" 버튼 클릭 시 MonetizationManager가 리워드 광고를 실행하고, 성공 시 100코인을 지급하며 UI를 업데이트합니다 ([UI Store | Notion](https://wmelongames.notion.site/UI-Store-18bc0f2fe155808193e4d5d7fd72bdfd#:~:text=The%20UI%20Store%20displays%20various,Module%20and%20the%20Rewards%20System)). "No Ads $4.99" 구매 시 Unity IAP 흐름을 거쳐 MonetizationManager가 광고 제거 상태를 저장하고, 이후로 광고 배너/팝업을 끄고 No Ads 버튼을 숨깁니다 ([Monetization (Ads & IAP) | Notion](https://wmelongames.notion.site/Monetization-Ads-IAP-13bc0f2fe155801f9db2d7b4f0e43b74#:~:text=Image%3A%20image)). **업그레이드 메뉴**에서는 보유 코인을 사용해 무기/캐릭터를 강화할 수 있습니다. Upgrade 버튼 누르면 현재 코인과 가격을 체크하여 업그레이드 적용 (예: 무기 데미지 +10%) 및 코인 차감, UI 갱신을 합니다. UpgradeManager/PlayerStats가 이 역할을 하며, SaveManager에 업데이트된 상태를 저장합니다.
8. **난이도 자동 조절** – 게임 진행 중, 플레이어의 업그레이드 상태에 따라 적의 강함이 자동으로 조절되는 것이 눈에 띌 것입니다. 이는 Difficulty Balancing 시스템이 작동한 결과로, GameManager 또는 EnemyController에서 적 스폰 시점에 현재 레벨의 **RequiredUpg**와 플레이어의 총 업그레이드 수치를 비교하여 적의 체력/공격력을 보정하기 때문입니다 ([Difficulty Balancing | Notion](https://wmelongames.notion.site/Difficulty-Balancing-08ea6e1f9a6e4d43a9e58618558a6be9#:~:text=Each%20level%20has%20a%20field,required%2C%20enemies%20will%20feel%20stronger)). 예를 들어 RequiredUpg=5인데 플레이어가 무기+캐릭터 합쳐 업그레이드 3단계만 했다면, 적에게 +20% 스탯 버프를 주어 도전하도록 합니다. 반대로 플레이어가 업그레이드가 매우 높으면 적 스탯을 너프하여 한방에 죽진 않도록 합니다. 이 계산은 EnemyController 초기화 시 이루어지며, 밸런스 데이터는 Weapon/Character 업그레이드 설정에 지정된 **Key Upgrade Number**를 참조합니다 ([Difficulty Balancing | Notion](https://wmelongames.notion.site/Difficulty-Balancing-08ea6e1f9a6e4d43a9e58618558a6be9#:~:text=To%20%E2%80%9Cexplain%E2%80%9D%20the%20game%20how,want%20the%20player%20to%20be)).

* **기타**: 이 밖에 **특수 이벤트** (예: 특정 레벨에서만 나타나는 연출), **사운드 매니저** (배경음악과 효과음 재생), **오디오 볼륨 설정**, **진동 피드백** 등이 있을 수 있습니다. 사운드 매니저는 Sound 폴더의 오디오 클립을 미리 로드해두고 이벤트 발생 시 재생하며, 게임 설정에서 볼륨을 조절하면 PlayerPrefs 등에 저장합니다. 또한 **애널리틱스/로그 수집**도 붙일 수 있지만 템플릿 기본에는 포함되지 않습니다. 필요하면 Unity Analytics 패키지를 추가해 이벤트를 보내도록 커스터마이즈합니다.

以上의 스크립트들이 유기적으로 동작하여 게임이 진행됩니다. **코드 구조는 확장성을 고려**하여 작성되어 있어, 예를 들어 새로운 무기나 적을 추가할 때 해당 클래스만 추가하면 나머지 시스템과 쉽게 통합될 수 있습니다 ([Squad Shooter | Template + Editor by Watermelon Games](https://wmelongames.itch.io/squad-shooter-template-editor#:~:text=Customization%20Guides)). 또한 **데이터 주도 설계**를 활용하고 있어, 각종 밸런스 수치나 콘텐츠 구성은 ScriptableObject 데이터 자산으로 분리되어 있습니다. 개발자는 게임 디자인 변경 시 코드 수정보다는 데이터 자산 수정을 통해 신속하게 반영할 수 있습니다 (예: 경험치 테이블 조정, 아이템 가격 변경 등). 이것이 템플릿이 **클린 코드 & 구조**를 표방하는 큰 장점입니다 ([Squad Shooter | Template + Editor by Watermelon Games](https://wmelongames.itch.io/squad-shooter-template-editor#:~:text=Customization%20Guides)).

마지막으로, 템플릿의 코드에는 풍부한 **주석**과 **문서**가 제공되어 있어 (함수마다 활용 방법 등이 기술됨), 필요할 때 참고하면 이해에 도움이 됩니다. Watermelon Games는 Discord 등을 통한 지원도 제공하므로, 고급 커스터마이즈를 시도할 때 궁금한 점이 있으면 커뮤니티 도움을 받을 수도 있습니다.

**레벨 제작 (심층 분석: Level Creation)**

이 장에서는 Watermelon Games의 공식 노션 문서 중 **Level Creation** 페이지의 내용을 토대로, 템플릿에서 **레벨을 제작하는 워크플로우와 도구, 구성 요소**를 심층 분석합니다 ([Level Creation | Notion](https://wmelongames.notion.site/Level-Creation-729328e4fcd74f6ab2f87b46b4bfbcc9#:~:text=Home%20Page)) ([Level Creation | Notion](https://wmelongames.notion.site/Level-Creation-729328e4fcd74f6ab2f87b46b4bfbcc9#:~:text=UI%20Store)). 앞서 *맵 디자인* 섹션에서 개괄적으로 다루었지만, 여기서는 에디터 툴의 구조와 레벨 데이터 구성 방식, 그리고 방(Room) 프리셋 개념에 대해 구체적으로 설명합니다.

**레벨 편집기 구조 (Level Editor Structure)**

**Level Editor**는 Squad Shooter 템플릿에 포함된 **커스텀 Unity 에디터 창**으로, 개발자가 월드와 레벨을 시각적으로 관리할 수 있는 도구입니다. 이 편집기의 주요 인터페이스 구성은 다음과 같습니다:

* **월드 선택 영역**: Level Editor 상단에는 현재 편집 중인 \*\*월드(World)\*\*를 표시하고 선택하는 UI가 있습니다 ([World System | Notion](https://wmelongames.notion.site/World-System-a719e8ab7f5a446c8e25d4e35c687c80#:~:text=World%20assets%20are%20accessible%20through,the%20currently%20selected%20world%20here)). 좌우 화살표 버튼(◀︎◀︎ / ▶︎▶︎)을 눌러 이전/다음 월드로 전환할 수 있으며, 가운데에는 현재 월드의 이름 (예: World\_01 (Desert))이 링크 형태로 표시됩니다 ([World System | Notion](https://wmelongames.notion.site/World-System-a719e8ab7f5a446c8e25d4e35c687c80#:~:text=Image%3A%20Untitled)). 이 링크를 클릭하면 해당 World 자산을 프로젝트에서 하이라이트하거나 인스펙터에서 열 수 있습니다. 새로운 월드를 추가하려면 (+) 버튼 또는 메뉴를 통해 **Add World** 기능을 사용하고, World Settings 탭에서 세부 설정을 해야 합니다 (아래 참조) ([World System | Notion](https://wmelongames.notion.site/World-System-a719e8ab7f5a446c8e25d4e35c687c80#:~:text=modify%2C%20or%20remove%20levels,dive%20deeper%20into%20this%20tab)).
* **레벨 목록 (Levels Creation 탭)**: Level Editor의 왼쪽 혹은 상단 탭 중 "Levels" 관련 탭을 선택하면, 현재 월드에 속한 **레벨 리스트**가 계층 구조로 나타납니다 ([World System | Notion](https://wmelongames.notion.site/World-System-a719e8ab7f5a446c8e25d4e35c687c80#:~:text=Down%20below%20you%20will%20see,related%20to%20the%20selected%20world)). 일반적으로 월드 안에 Level 1, Level 2, ... 식으로 나열되며, 각 레벨 엔트리 옆에는 편집/삭제 버튼이 있습니다. 레벨 엔트리를 클릭하면 우측 패널에 해당 레벨의 상세 설정과 편집 UI가 표시됩니다. 레벨 리스트 아래에는 **Add Level** 버튼이 있어 새로운 레벨을 생성합니다 ([World System | Notion](https://wmelongames.notion.site/World-System-a719e8ab7f5a446c8e25d4e35c687c80#:~:text=Down%20below%20you%20will%20see,related%20to%20the%20selected%20world)). 레벨 순서는 곧 게임 진행 순서를 의미하므로, 리스트에서 드래그로 순서를 변경할 수 있을 수도 있습니다 (혹은 생성 순서대로 고정). 또한 레벨을 우클릭하여 \*\*복제(Duplicate)\*\*나 **삭제(Delete)** 같은 옵션이 제공될 것입니다.
* **레벨 상세 패널**: 레벨을 선택했을 때 나타나는 패널로, **General Level Settings**과 **Level Editor Canvas**로 구성됩니다.
  + 상단에는 **레벨 이름**(식별자)과 간략 설명을 입력할 수 있고, **난이도 파라미터**로서 *Required Upgrades* 값 등을 설정합니다 ([Difficulty Balancing | Notion](https://wmelongames.notion.site/Difficulty-Balancing-08ea6e1f9a6e4d43a9e58618558a6be9#:~:text=Each%20level%20has%20a%20field,required%2C%20enemies%20will%20feel%20stronger)). 또한 해당 레벨에서 **획득 가능한 코인/XP** 양이나, **제한 시간**(타임어택 요소가 있다면) 등을 지정하는 필드가 있을 수 있습니다.
  + 패널의 본체는 **레벨 편집 캔버스**로, 말하자면 현재 레벨의 지형/오브젝트 배치를 시각적으로 편집하는 공간입니다. 여기에는 **Grid**나 **배경 타일**이 보이며, 개발자가 오브젝트를 배치할 수 있는 **에디터 뷰**가 나타납니다. Unity의 Scene 뷰를 그대로 사용하는 대신, Level Editor 창 내에 별도의 미니 Scene처럼 구현했을 가능성도 있지만, 일반적으로는 **Scene 뷰를 활용**합니다. Level Editor에서 어떤 레벨을 선택하면 내부적으로 그 레벨의 지형과 오브젝트를 Scene에 배치하여 미리보기/편집 가능하게 해주고, 편집이 끝나면 저장하는 방식입니다. (기술적으로는 편집용 임시 씬을 열거나, HideFlags를 이용해 Hierarchy에 보이지 않는 편집 모드를 쓸 수도 있습니다).
* **오브젝트 팔레트 패널**: Level Editor 창의 한쪽에는 \*\*배치 가능한 오브젝트 목록(팔레트)\*\*이 있습니다. 이는 **Type별**(카테고리) 탭으로 구분되어, 예를 들어 **Environment**, **Obstacles**, **Enemies**, **Collectibles** 등의 탭이 존재합니다. 각 탭을 선택하면 해당 타입에 등록된 프리팹들의 썸네일 리스트가 나타납니다 ([Obstacles | Notion](https://wmelongames.notion.site/Obstacles-bc6922b59fff4090ab9eb1ab35d40ff2#:~:text=2,Add)). 이 목록은 **World Settings**에서 관리되는 리스트를 기반으로 합니다. 즉, 현재 선택된 월드에서 사용가능하도록 등록된 오브젝트만 팔레트에 표시됩니다 ([Obstacles | Notion](https://wmelongames.notion.site/Obstacles-bc6922b59fff4090ab9eb1ab35d40ff2#:~:text=1.%20World%20,during%20level%20creation%20for%20clarity)). 예를 들어 Desert 월드에는 선인장 장애물이 등록되어 있지만, Mines 월드에는 지뢰 장애물이 등록되어 있다면, Desert 선택 시 선인장 프리팹이 팔레트에 나오고 Mines 선택 시 지뢰 프리팹이 나옵니다. 팔레트의 오브젝트를 클릭한 후 **Scene 뷰**를 클릭하면 해당 위치에 오브젝트가 배치되거나, 또는 드래그 앤 드롭으로 배치합니다. 이때 일정한 **그리드 스냅**이 적용될 수 있어 정렬된 배치가 가능합니다 (Grid 크기는 환경 타일 사이즈와 일치하게 설정됨). 팔레트 상단에는 검색창이나 필터가 있어 오브젝트를 이름으로 찾을 수도 있습니다.
* **월드 설정 (World Settings 탭)**: Level Editor의 다른 탭으로, 현재 월드에 대한 전반적인 설정을 편집합니다 ([World System | Notion](https://wmelongames.notion.site/World-System-a719e8ab7f5a446c8e25d4e35c687c80#:~:text=%2A%20Levels%20Creation%20,dive%20deeper%20into%20this%20tab)). 여기서는 **해당 월드 테마의 리소스**들을 지정하는데, 예컨대:
  + **Environment Prefabs**: 그 월드에서 쓸 지형/장식 프리팹 리스트 (팔레트에 표시될 항목들) ([Environment | Notion](https://wmelongames.notion.site/Environment-991c663dc33443659795e9d6be81a6b0#:~:text=1.%20Right,Add%20into%20Level%20Editor)).
  + **Obstacle Prefabs**: 그 월드에서 쓸 장애물 프리팹 리스트 ([Obstacles | Notion](https://wmelongames.notion.site/Obstacles-bc6922b59fff4090ab9eb1ab35d40ff2#:~:text=1.%20World%20,during%20level%20creation%20for%20clarity)).
  + **Enemy Prefabs**: 등장시킬 적 유형 리스트.
  + **Background/Lighting**: 월드 특유의 배경 색상이나 하늘박스, 조명 세기 등을 조정.
  + **음향(BGM)**: 해당 월드에서 재생할 배경음악이나 환경음 설정.

World Settings 탭에서 **Add New Item** 버튼을 통해 이 월드에 새로운 프리팹을 등록할 수도 있습니다 (사실상 Level Editor 창에서 직접 "Add into Level Editor" 기능을 하는 버튼일 것입니다). 예를 들어 Desert 월드에 새로운 선인장 종류 장애물을 추가하려면 여기서 + 버튼으로 목록에 추가하고 프리팹을 연결하면 됩니다 (혹은 앞서 Project 창 우클릭으로 이미 추가한 경우 여기에 나타나겠죠). 또한 **월드 이름**이나 **월드 순서**, **잠금 조건**(월드2는 월드1 클리어시에만 잠금 해제 등)도 설정할 수 있습니다. "How to add a new world" 절차에 따르면, 새로운 World 자산을 만들고 여기서 테마에 맞게 이러한 설정을 지정하면 된다고 안내하고 있습니다 ([World System | Notion](https://wmelongames.notion.site/World-System-a719e8ab7f5a446c8e25d4e35c687c80#:~:text=modify%2C%20or%20remove%20levels,dive%20deeper%20into%20this%20tab)).

* **룸 프리셋 관리**: Level Editor UI 어딘가에는 **Room Presets**와 관련된 인터페이스도 있을 수 있습니다. (문서의 "How to create a room preset" 섹션에 해당) 이는 방 단위로 미리 설계된 레이아웃을 저장하고 불러오는 기능입니다. Level Editor 상단 메뉴나 레벨 상세 패널 내에 **Save as Preset** 버튼이 존재할 수 있습니다. 방 하나를 모두 디자인한 뒤 이 버튼을 누르면 해당 방에 있는 **모든 오브젝트 배치 구성이 하나의 프리셋**으로 저장됩니다. 저장된 프리셋은 **프리셋 목록**에 추가되고, 이후 다른 레벨에서 새로운 방을 만들 때 이 프리셋을 불러와 동일한 구조를 재사용할 수 있습니다. 이렇게 하면 유사한 방 구조를 반복해서 만들 필요 없이 한 번의 디자인으로 여러 레벨에 활용할 수 있습니다 (예: "원형 경기장 형태 방", "미로형 방" 등의 프리셋). 프리셋은 월드 공용으로 쓰이거나, 월드별로 따로 관리될 수도 있습니다. 아마 **Resources/Presets** 폴더 등에 프리셋 데이터(배치 좌표 목록 형태)가 저장될 것입니다. Level Editor에서는 **Apply Preset** 기능으로 빈 방에 프리셋을 적용하거나, 기존 방 레이아웃을 덮어쓸 수 있을 것입니다.

요약하면, Level Editor는 **월드-레벨-방/객체**의 계층적 구조를 한 화면에서 편집할 수 있게 해주는 도구입니다. 덕분에 개발자는 **여러 스테이지를 손쉽게 생성, 관리**할 수 있으며, 디자인 변경이 있을 때도 실시간으로 수정하여 게임에 반영할 수 있습니다 ([World System | Notion](https://wmelongames.notion.site/World-System-a719e8ab7f5a446c8e25d4e35c687c80#:~:text=Down%20below%20you%20will%20see,related%20to%20the%20selected%20world)). 이러한 구조화된 편집 툴은 완성형 템플릿의 큰 장점으로, 코드를 수정하지 않고도 콘텐츠 제작을 가능케 합니다.

**레벨의 일반 구조 (General Level Structure)**

Squad Shooter의 각 \*\*레벨(Level)\*\*은 내부적으로 \*\*여러 개의 방(Room)\*\*으로 구성될 수 있는 구조입니다. 일반적인 레벨 구조와 그 데이터를 구성하는 요소들을 살펴보겠습니다:

* **방(Room)**: 하나의 레벨은 1개 이상의 방을 포함합니다. 방은 전투가 일어나는 단위 공간으로, 보통 사방이 벽으로 둘러싸여 있고 입구/출구에 \*\*문(Gate)\*\*이 있습니다 ([Environment | Notion](https://wmelongames.notion.site/Environment-991c663dc33443659795e9d6be81a6b0#:~:text=,like%20ground%2C%20walls%2C%20and%20gates)). 처음 방의 입구는 플레이어 시작 위치, 마지막 방의 출구는 레벨 종료 지점으로 통합니다. 각 방에는 특정 수의 적이 배치되며, 플레이어가 모든 적을 처치하면 출구 Gate가 열려 다음 방으로 이동할 수 있게 됩니다. 방 사이에는 로딩 없이 이어지며, 논리적으로만 구분됩니다.
* **방 연결(Gates)**: Gate는 일종의 문 또는 차단막 역할을 하는 오브젝트입니다 ([Environment | Notion](https://wmelongames.notion.site/Environment-991c663dc33443659795e9d6be81a6b0#:~:text=,like%20ground%2C%20walls%2C%20and%20gates)). 플레이어가 한 방의 적을 모두 처치하기 전까지는 닫혀 있어 다음 방으로 진행을 막다가, 방 클리어 시 열리는 식입니다. 구현상 Gate 오브젝트는 **Obstacle 레이어**로 작동하여 닫혀있을 때는 플레이어/적 이동과 탄환을 막고, 열리면 콜라이더를 비활성화하거나 투명화하여 통과 가능하게 합니다. Level Editor에서 Gate는 Environment 목록에 속하며 필요한 위치(두 방 사이 통로)에 배치됩니다. Gate가 열리는 조건은 GameManager 또는 SpawnManager가 해당 방의 **적 웨이브 상태**를 모니터링하여 결정합니다. (예: if(currentRoomEnemiesCleared) OpenGate(currentRoomIndex) 같은 로직)
* **적 웨이브(Wave)**: 각 방마다 하나 이상의 **적 웨이브**가 있을 수 있습니다. 간단한 경우 방 진입 시 모든 적이 동시에 나타나지만, 난이도가 높아지면 **연속 웨이브**를 넣을 수 있습니다. 예를 들어 "방2에서는 처음에 적 5명 -> 모두 쓰러뜨리면 즉시 보강병 3명 추가 스폰" 같은 식입니다. 이런 웨이브 구성은 레벨 데이터의 일부로 저장됩니다. Level Editor에서 특정 방을 편집할 때 **Wave 설정** UI가 있어, 웨이브 수와 웨이브별 적 종류/수를 정의할 수 있습니다. 또는, 웨이브까지 세부 설정하지 않고 단순히 방마다 적 목록만 정해두면, SpawnManager가 내부 규칙(난이도 곡선 등)에 따라 자동으로 웨이브 분배를 할 수도 있습니다. 템플릿 문서에서는 주로 Required Upgrades 등 난이도에 중점을 두고 있으므로, 웨이브 디자인은 개발자 재량에 맡긴 것으로 보입니다.
* **레벨 데이터 저장**: 각각의 Level 정보는 **World 자산**에 포함되어 관리됩니다. 예를 들어 World\_01.asset 내에 Level 1 ~ 10의 설정이 계층적으로 들어가 있을 수 있습니다. 또는 별도의 Level asset들을 생성하여 World 자산이 참조하는 방식일 수도 있습니다. 어떤 방식이든, 레벨 하나에는 다음과 같은 데이터 요소들이 있습니다:
  + *레벨 ID/이름*: (예: "1-3", "Desert-3") 표시용이나 디버그용 식별자.
  + *방 구성*: 방의 수 및 각 방의 오브젝트 배치 정보(프리팹 타입과 좌표). 이는 룸 프리셋 적용 여부에 따라 프리셋 참조 또는 개별 객체 나열로 저장됩니다.
  + *적 구성*: 각 방에 등장하는 적 종류와 수. (및 등장 타이밍/웨이브 정보)
  + *난이도 지표*: Required Upgrades 값 ([Difficulty Balancing | Notion](https://wmelongames.notion.site/Difficulty-Balancing-08ea6e1f9a6e4d43a9e58618558a6be9#:~:text=Each%20level%20has%20a%20field,required%2C%20enemies%20will%20feel%20stronger)), 그리고 권장 플레이어 레벨 또는 예상 업그레이드 상태. 이 수치는 Difficulty Balancing에 쓰입니다. 예를 들어 Level5에 RequiredUpg=3으로 설정하면, 시스템이 플레이어 업그레이드와 비교해 적 능력치를 조정합니다.
  + *보상*: 클리어 시 획득하는 기본 코인, 경험치 양 ([Experience System | Notion](https://wmelongames.notion.site/Experience-System-f88c15e9f92d4cf8a0dea0b2a5035ecd#:~:text=After%20completing%20a%20level%20player,here)). (추가로 별 3개 평가 시스템이 있다면, 별 개수에 따른 보너스 보상 등도)
  + *특수 조건*: 시간 제한, 특정 무기 사용 강제 등의 특수 조건이 있다면 기록.
  + *다음 레벨*: 레벨 완료 후 진행될 다음 레벨 참조. (순차적이라 World/Level 리스트 순서로 하므로 필요 없을 수도 있음)

이러한 데이터는 ScriptableObject 형태로 직렬화되어 프로젝트에 저장됩니다. 빌드된 게임에서는 이 데이터를 읽어 레벨을 로드하게 됩니다. ScriptableObject 활용으로 **로딩이 빠르고** 필요한 데이터만 메모리에 유지할 수 있으며, **수정 시 바로 에디터에 반영**되는 장점이 있습니다 ([Obstacles | Notion](https://wmelongames.notion.site/Obstacles-bc6922b59fff4090ab9eb1ab35d40ff2#:~:text=,the%20game%27s%20quick%20loading%20system)).

* **NavMesh 및 경로**: 일반 구조 측면에서, 모든 레벨(방들)은 하나의 NavMesh 맵을 공유합니다. Game 씬 기준으로 보면, 여러 방으로 이루어진 하나의 큰 NavMesh 지형이 있다고 볼 수 있습니다. NavMeshAgent(적 AI)는 방 구분 없이 전체 NavMesh에서 길을 찾지만, 게이트가 닫혀있을 때는 NavMeshObstacle로 막혀있으므로 다른 방의 적이 오지 못할 뿐입니다. 따라서 플레이어가 문을 열면 NavMeshObstacle이 꺼지고 적들이 추적해올 수 있습니다. 이런 구조는 **씬 전환 없이 연속된 전투 공간**을 구현하기에 적합하며, 모바일 게임에서 자주 쓰이는 방-별 진행 구조를 잘 뒷받침합니다.
* **난이도 & 업그레이드 상관관계**: 일반 구조에서 특이한 점은, **Difficulty Balancing** 시스템이 레벨 데이터의 일부인 RequiredUpg와 전역 데이터인 업그레이드 현황을 연계하여 동작한다는 것입니다. 이는 앞서 설명했듯 **동적인 레벨 난이도 조정**을 실현합니다. 결과적으로 어떤 레벨은 플레이어의 성장 정도에 따라 체감 난이도가 달라지지만, 개발자가 의도한 “적정 난이도” 경험은 유지됩니다 ([Difficulty Balancing | Notion](https://wmelongames.notion.site/Difficulty-Balancing-08ea6e1f9a6e4d43a9e58618558a6be9#:~:text=Each%20level%20has%20a%20field,required%2C%20enemies%20will%20feel%20stronger)) ([Difficulty Balancing | Notion](https://wmelongames.notion.site/Difficulty-Balancing-08ea6e1f9a6e4d43a9e58618558a6be9#:~:text=To%20%E2%80%9Cexplain%E2%80%9D%20the%20game%20how,want%20the%20player%20to%20be)). 예를 들어 Level 10의 RequiredUpg = 5, Level 11의 RequiredUpg = 6으로 설정해두면, 플레이어가 Level 11쯤엔 무기나 캐릭터 업글을 1회 더 했을 거라 기대하고 난이도를 설계하는 것입니다. 만약 플레이어가 업글을 안 하고 왔다면 11레벨의 적들은 강하게 느껴질 테고, 업글을 많이 하고 왔다면 쉽게 느껴지는 대신 곧 업글 없이 어려운 레벨이 나와 밸런스를 맞춥니다. 이 시스템 덕분에 레벨 디자인 시 정확한 수치 튜닝 부담을 덜 수 있습니다.
* **레벨 간 변수**: 어떤 레벨들은 새로운 기믹(함정, 장애물 움직임 등)이 있을 수 있습니다. 예를 들어 특정 레벨에서만 맵 위를 가로지르는 레이저 함정이 존재한다면, 그것도 레벨 데이터에 플래그를 넣어둘 수 있습니다 (e.g., hasLaserTrap = true). GameManager는 해당 레벨 시작 시 그 함정을 활성화하는 식입니다. 또, 보스전 레벨이라면 isBossLevel = true로 표시하고, EnemySpawnManager가 이를 보고 보스 적 1명만 생성하도록 특별 처리할 것입니다. 이러한 특수 상황 데이터도 레벨 구조의 일부로 포함될 수 있습니다.

**새 레벨 생성 방법 (How to Create a Level)**

**새로운 레벨을 만드는 방법**을 단계적으로 정리하면 다음과 같습니다. (이는 Level Editor 사용법의 일환으로, 일부 내용은 앞선 *맵 디자인* 절과 중복될 수 있습니다.)

1. **월드 선택/생성**: 새 레벨을 어느 월드에 추가할지 결정합니다. 기존 월드에 추가하려면 Level Editor에서 해당 월드를 선택하고 진행합니다. 새로운 월드를 먼저 만들고 그 안에 레벨을 만들 수도 있습니다. 새 월드는 Level Editor 메뉴의 **Add World** 기능을 통해 생성하며, World Settings에서 테마 설정을 완료합니다 ([World System | Notion](https://wmelongames.notion.site/World-System-a719e8ab7f5a446c8e25d4e35c687c80#:~:text=%2A%20Levels%20Creation%20,dive%20deeper%20into%20this%20tab)).
2. **레벨 추가**: Level Editor의 **Levels Creation** 탭에서 **Add Level** 버튼(+)을 클릭합니다 ([World System | Notion](https://wmelongames.notion.site/World-System-a719e8ab7f5a446c8e25d4e35c687c80#:~:text=Down%20below%20you%20will%20see,related%20to%20the%20selected%20world)). 레벨 리스트에 "New Level" 항목이 추가되고 자동으로 선택됩니다. 기본 이름은 임시로 부여되므로 인스펙터/패널에서 레벨 이름을 "Level X" 등으로 바꿉니다. (이름은 주로 편의용이며, 표시명은 다른 곳에서 관리될 수도 있습니다)
3. **레벨 속성 설정**: 새 레벨의 **Required Upgrades**, **보상 값(코인/XP)** 등을 입력합니다. 난이도 곡선 상 이 레벨이 어느 시점인지 고려하여 RequiredUpg 값을 정합니다 (예: 이전 레벨보다 1 높게, 혹은 동일하게). 기본 보상은 적당히 설정하고 나중에 조정 가능합니다.
4. **방 구성**: 처음에는 방이 1개 비어 있는 상태일 것입니다 (Level Editor가 자동 생성하거나, 새 레벨 추가 시 디폴트 방 1개를 넣어줄 수 있음). **방 추가/삭제** 버튼을 이용해 원하는 개수만큼 방을 만듭니다. 보통 선형 진행이라면 일렬로 방을 나열하면 되지만, 만약 분岐 구조 (예: 두 갈래 길) 같은 것을 만들려면 특별한 지원이 필요할 것입니다. 템플릿에서는 복잡한 분岐보다는 단순 직선 진행을 염두에 두었을 가능성이 큽니다.
5. **지형 배치**: 각 방별로 지면과 벽을 배치합니다. Level Editor 팔레트에서 Ground를 선택해 방 크기만큼 바닥 타일을 채웁니다. 벽 프리팹을 이용해 방 경계를 둘러쌉니다. 방과 방 사이 경로에는 Gate 프리팹을 놓습니다. (첫 방 입구 쪽 Gate는 플레이어 시작 위치이므로 열려있는 형태, 중간 방 게이트는 닫힘 상태로 시작). 필요한 경우 방 내부에 기둥이나 장식(Environment)을 넣어 다양성을 줍니다.
6. **장애물 배치**: 팔레트의 Obstacles에서 엄폐물/장애물을 골라 각 방에 배치합니다. 전투의 전략성을 높일 수 있도록, 방 크기에 맞춰 몇 개의 엄폐물을 배치하고, 플레이어와 적의 길목을 형성합니다. 장애물 배치 후에는 Collider와 NavMeshObstacle 세팅이 제대로 되었는지 확인합니다 (대부분 자동이지만 혹 수정했다면). **NavMesh Bake**를 수행하여 새 장애물에 대한 경로 차단이 반영되도록 합니다.
7. **적 스폰 설정**: **Enemies** 팔레트/탭에서 적을 배치합니다. 방당 적 출현 수를 결정하고, 각 적의 종류를 지정합니다. 만약 Level Editor가 **적 프리팹 직접 배치** 방식을 취한다면, Scene 뷰 해당 방 영역에 적 프리팹을 놓습니다. 그러나 이렇게 놓으면 에디터에서 플레이 모드로 테스트할 때만 보이고, 실제 게임에서는 GameManager가 Scene에 배치된 적을 참조하여 진행할 수도 있습니다. (일부 템플릿은 씬에 배치된 적들을 초기화 시 읽어서 사용하기도 합니다.) 다른 방식으로, **Spawn Point**나 **Spawn Zone** 오브젝트를 배치하는 형태일 수도 있습니다. 예를 들어 방 안에 몇 군데 SpawnPoint를 떨어뜨리고, 각 SpawnPoint에 어떤 적이 몇 마리 나올지 숫자를 붙여넣는 식입니다. Level Editor UI에서 방을 선택하면 우측에 Enemy Waves 설정표가 나타나고, 거기서 "Wave1: Grunt x5, Wave2: Sniper x2" 등 입력하게 할 수도 있습니다.
   * 간단한 경우라면 "이 방 총 적 수: N, 동시 등장: M" 정도만 설정하고, 내부적으로 알아서 두 웨이브로 나누는 자동화를 해둘 수도 있습니다.
   * 방이 많지 않은 초반 월드는 한 방에 한 웨이브로 끝내고, 후반 월드는 두 웨이브 이상 주는 디자인을 생각해볼 수 있습니다. 설정이 완료되면 SpawnManager가 그 데이터를 제대로 읽는지, 플레이 모드 테스트를 통해 확인합니다.
8. **테스트 및 조정**: 에디터에서 **Play** 모드로 들어가 새로 만든 레벨을 테스트합니다. (특정 레벨 바로 테스트 기능이 없으면, 임시로 GameManager 코드에서 startLevel를 우리가 만든 ID로 설정하거나, 기존 레벨과 교체하여 시험). 플레이하면서 방 크기나 장애물 위치가 어색하면 에디터로 돌아와 수정합니다. 적 수나 배치는 너무 쉽거나 어려운지 판단해 조절합니다. RequiredUpg 난이도 설정이 적절했는지도 느껴볼 수 있습니다 (예: 플레이어가 업그레이드 없이도 너무 쉽게 깼다면 RequiredUpg를 낮춰야 할지 등). 이러한 **반복 테스트와 수치 조정**을 거쳐 레벨을 완성합니다.
9. **레벨 데이터 저장**: Level Editor에서는 작업 내용이 실시간으로 ScriptableObject 자산에 반영되거나, 별도 **Save** 버튼으로 저장하게 할 수 있습니다. 보통 즉시 저장되지만, 혹시 Save Level/World 버튼이 있다면 수시로 눌러 저장합니다. Unity 프로젝트도 저장(Ctrl+S)하여 변경된 에셋을 기록합니다.
10. **다른 월드 연계**: 만약 이 레벨이 특정 월드의 마지막 레벨이라면, **월드 전환**이 제대로 되는지도 생각해야 합니다. 예를 들어 World1 마지막 레벨 클리어 시 World2가 잠금 해제되고 월드 선택 UI가 뜨는지, 혹은 바로 World2 Level1로 넘어가는지 시나리오를 점검합니다. 이 로직은 GameManager나 UIManager 측이 담당하지만, Level/World 데이터에도 "isLastLevel" 등의 플래그가 있을 수 있습니다. 특별히 개발자가 건드릴 부분은 아니나, 테스트 중 흐름에 문제가 있으면 데이터 플래그를 확인해야 합니다.

이상으로 새로운 레벨 한 개를 제작하여 게임에 추가하는 절차를 살펴보았습니다. **Level Editor 툴** 덕분에 대부분의 작업이 **시각적 에디팅**으로 이뤄지며, 스크립트 수정 없이도 레벨 콘텐츠를 풍부하게 만들 수 있습니다 ([Obstacles | Notion](https://wmelongames.notion.site/Obstacles-bc6922b59fff4090ab9eb1ab35d40ff2#:~:text=,the%20game%27s%20quick%20loading%20system)). 특히 다수의 레벨을 만들 때 **룸 프리셋**과 **객체 재사용** 기능이 큰 도움이 됩니다. 몇 가지 방 구조를 프리셋으로 만들어 놓고 이를 조합하면, 짧은 시간에 다양한 스테이지를 효율적으로 설계할 수 있습니다.

**레벨에 커스텀 오브젝트 추가 (Adding Custom Objects to Levels)**

게임 디자이너가 기본 제공되지 않은 **특수 오브젝트**를 레벨에 추가하고자 할 때의 절차와 고려사항입니다. 예를 들어 **폭발성 드럼통**이나 **이동하는 함정** 같은 오브젝트를 넣고 싶다고 합시다. 이런 오브젝트를 성공적으로 레벨에 배치하려면, 앞서 *적 및 오브젝트 배치* 절에서 논의한 **프리팹 등록 절차**를 따라야 합니다:

1. **프리팹 준비**: 프로젝트에 해당 오브젝트의 프리팹을 만든다. 모델과 애니메이션, 스크립트 등을 설정하고, **Collider**를 붙인다 (총알 및 물리 충돌용) ([Obstacles | Notion](https://wmelongames.notion.site/Obstacles-bc6922b59fff4090ab9eb1ab35d40ff2#:~:text=1,to%20turn%20into%20an%20obstacle)). 움직이는 함정이라면 Rigidbody와 관련 스크립트도 추가한다. 오브젝트가 **플레이어/적 이동에 영향을 주는 경우**, Collider 외에 NavMeshObstacle을 붙여 경로계산에 반영시킨다 ([Obstacles | Notion](https://wmelongames.notion.site/Obstacles-bc6922b59fff4090ab9eb1ab35d40ff2#:~:text=4,or%20NPC%20to%20go%20through)). 또한 오브젝트의 **레이어 설정**도 한다. 총알에 피격되는 대상이라면 Obstacle 레이어로 두어 총알이 막히게 하고, Ground 성격이면 Ground 레이어로 설정하여 NavMesh에 포함시킨다 ([Environment | Notion](https://wmelongames.notion.site/Environment-991c663dc33443659795e9d6be81a6b0#:~:text=3,prevent%20aiming%20through%20the%20obstacles)).
2. **Level Editor 등록**: 준비된 프리팹을 Level Editor에 알려줘야 팔레트에 나타난다. 프로젝트창에서 프리팹을 우클릭 → **Add into Level Editor** 수행하고, **World**를 선택하며, **Type**은 용도에 맞게 지정한다 (예: 폭발 드럼통은 장애물처럼 취급하여 Type=Obstacle로 추가하거나, 별도 "Trap" 타입이 있다면 선택) ([Obstacles | Notion](https://wmelongames.notion.site/Obstacles-bc6922b59fff4090ab9eb1ab35d40ff2#:~:text=1.%20Right,Add%20into%20Level%20Editor)). Level Editor 창의 World Settings 탭에서 해당 항목이 리스트에 추가됐는지 확인한다 ([Obstacles | Notion](https://wmelongames.notion.site/Obstacles-bc6922b59fff4090ab9eb1ab35d40ff2#:~:text=Image%3A%20Untitled)).
3. **레벨 배치 및 기능 구현**: 이제 팔레트에 등장한 새로운 오브젝트를 원하는 레벨의 방에 배치한다. 배치 후에는 **게임 플레이 시 어떻게 동작하는지** 확인해야 한다. 예를 들어 폭발 드럼통이라면, Bullet이 맞았을 때 터지는 효과와 주변 데미지 스크립트가 프리팹에 있어야 한다. 이런 **게임플레이 로직**은 템플릿 기본에는 없을 확률이 높으므로, 개발자가 직접 스크립트를 작성해 프리팹에 붙여야 한다. OnCollisionEnter로 총알 태그를 감지해 폭발시키고, 폭발 반경 내 적들에게 데미지를 주는 등의 코드를 구현한다. 함정 오브젝트도 마찬가지로, 일정 시간마다 작동하거나 플레이어 접근 시 발동하는 코드를 만들어야 한다. 템플릿의 **코드 구조를 활용**하여, 예컨대 GameManager나 EnemyController에 훅을 걸 수도 있고, 독립적으로 동작시킬 수도 있다.
4. **테스트 및 조정**: 커스텀 오브젝트를 넣은 레벨을 테스트해서 제대로 작동하는지 본다. NavMeshObstacle이 너무 크게 설정돼 적이 지나가지 못하거나, 폭발 데미지가 과도하거나 할 수 있으니 값을 튜닝한다. 여러 번 반복 테스트해도 문제가 없다면 해당 오브젝트를 다른 레벨에도 활용하여 다양성을 줄 수 있을 것이다.

이 때 중요한 점은, **Level Editor에 Type으로 등록되지 않은 오브젝트는 팔레트에 나타나지 않기 때문에 절차를 꼭 거쳐야** 한다는 것입니다 ([Obstacles | Notion](https://wmelongames.notion.site/Obstacles-bc6922b59fff4090ab9eb1ab35d40ff2#:~:text=,the%20game%27s%20quick%20loading%20system)). 또한 Type은 팔레트 분류를 위한 것이므로, 완전히 새로운 타입을 만들 필요는 없이 유사한 역할의 기존 타입에 넣어도 무방합니다 (예: 함정도 장애물의 일종으로 간주). 만약 정말 새로운 카테고리를 만들고 싶다면, 템플릿의 Level Editor 코드 확장이 필요합니다. 예컨대 "Trap"이라는 Type을 추가하려면, 에디터 스크립트에서 해당 열거형/탭을 추가하고 저장 데이터 구조도 업데이트해야 하는 등 복잡하므로, 특별한 이유가 없다면 기존 Type을 재사용하는 것이 좋습니다 ([Obstacles | Notion](https://wmelongames.notion.site/Obstacles-bc6922b59fff4090ab9eb1ab35d40ff2#:~:text=2,Add)).

**레벨 로딩 최적화**: 문서에 따르면 Level Editor에 오브젝트를 등록하는 행위는 **게임의 빠른 로딩 시스템의 핵심**이라고 언급됩니다 ([Obstacles | Notion](https://wmelongames.notion.site/Obstacles-bc6922b59fff4090ab9eb1ab35d40ff2#:~:text=,the%20game%27s%20quick%20loading%20system)). 이는 아마도 레벨 데이터에 객체 참조를 일일이 저장하지 않고, 등록된 리스트의 인덱스만 저장하여 경량화하는 식으로 구현했기 때문일 것입니다. 예를 들어 "Obstacle 타입 3번 객체를 좌표 (x,y)에 배치" 이런 식으로만 기록하고, 실제 프리팹 참조는 World Settings의 리스트를 통해 해소하는 방식입니다. 그렇기 때문에 모든 배치 오브젝트는 사전에 등록되어 있어야 하며, 그렇지 않으면 로딩 시 알 수 없는 객체가 되어버립니다. 따라서 **임시로 등록 안 한 채 Scene에 배치**만 하고 테스트하면, 에디터 플레이에서는 보이더라도 빌드된 게임에서는 해당 오브젝트가 누락될 수 있으니 주의해야 합니다.

**룸 프리셋 생성 방법 (How to Create a Room Preset)**

\*\*Room Preset(방 프리셋)\*\*은 Level Editor가 제공하는 강력한 기능으로, **방 하나의 디자인을 저장하여 재사용**할 수 있게 해줍니다. 새로운 룸 프리셋을 만드는 방법과 활용 방법은 다음과 같습니다:

* **프리셋 생성**: 레벨 디자인을 어느 정도 완료한 상태에서, 특정 방의 레이아웃이 마음에 들면 그것을 프리셋으로 저장할 수 있습니다. Level Editor에서 해당 방을 선택한 후, **"Save Room as Preset"** 버튼을 클릭합니다. (이 버튼은 방 편집 패널 내 혹은 상단 툴바에 있을 것입니다.) 클릭 시 프리셋 이름을 입력하도록 요청할 수 있습니다 (예: "SmallSquareRoom" 또는 자동으로 World1\_RoomA 등). 확인하면 내부적으로 그 방에 배치된 **모든 오브젝트의 종류와 좌표** 정보를 하나의 프리셋 데이터로 저장합니다. 성공하면 콘솔에 프리셋 생성 로그를 남기거나, Preset 목록에 추가됐다는 UI 피드백을 줄 것입니다. 생성된 프리셋 데이터는 프로젝트의 **Presets 폴더** (혹은 World 자산 내) 에 저장됩니다.
* **프리셋 확인/관리**: Level Editor의 월드 설정이나 별도 **Presets 관리 창**에서 방금 만든 프리셋을 확인할 수 있습니다. 프리셋에는 미리보기가 가능하도록 작은 썸네일 이미지가 자동 생성되었을 수도 있습니다 (씬을 캡처해서). 프리셋 목록에서 불필요한 프리셋을 삭제하거나 이름을 변경하는 기능도 제공될 것입니다. 프리셋 데이터는 에셋 형태이므로 프로젝트에서 관리 가능합니다.
* **프리셋 적용**: 이미 저장된 프리셋을 불러와 레벨에 적용하려면, 새로운 방을 만든 후 **"Apply Preset"** 기능을 사용합니다. 예를 들어 월드1의 Level5에 방 하나를 추가하고, 그 방이 기존에 만든 SmallSquareRoom 프리셋과 동일한 구조로 하고 싶다면, 방 선택 후 **Apply Preset -> SmallSquareRoom**을 실행합니다. 그러면 해당 방에 미리 저장된 대로 바닥, 벽, 장애물 배치가 한꺼번에 생성됩니다. (주의: 이미 방에 배치된 것이 있다면 덮어쓸지 묻거나, 빈 방에서만 적용되도록 제한할 수 있습니다.) 이렇게 하면 몇 초 만에 복잡한 방 레이아웃을 만들어낼 수 있습니다. 이후 적 배치 등만 조정하면 되므로 매우 효율적입니다.
* **다른 월드에서 활용**: 기본적으로 프리셋은 특정 월드의 테마에 종속되지 않을 수도 있습니다. 예를 들어 Desert 월드에서 만든 방 프리셋을 Sci-fi 테마 World에서도 쓸 수 있을까요? 만약 두 월드가 사용하는 오브젝트 풀(팔레트)이 많이 다르다면, 프리셋 적용 시 해당 오브젝트를 찾지 못하는 문제가 생깁니다. 이를 회피하려면 **공용 프리셋**은 공통으로 존재하는 오브젝트만으로 설계하거나, 프리셋을 만들 때 월드 한정으로 표기합니다. 템플릿이 어떻게 처리하는지는 문서에 상세히 나와 있지 않지만, 안전하게는 **월드별 프리셋**으로 운용하는 것이 좋습니다. 즉 Desert 방 프리셋은 Desert 월드 내에서만 사용하고, 다른 월드에는 어울리게 새로 디자인하는 것입니다. (만일 공유하고 싶다면 프리셋 적용 후 해당 오브젝트들을 그 월드에 맞는 것으로 일괄 치환하는 기능 등이 필요하나, 이는 구현되지 않았을 수 있습니다.)
* **프리셋 활용 사례**: 프리셋은 특히 **레벨 수가 많을 때** 유용합니다. 예를 들어 게임에 100개의 스테이지가 있다면, 독자적으로 100개의 방 레이아웃을 만들기는 버거울 수 있습니다. 이때 10종 정도의 방 프리셋을 만들어 조합하면, 플레이어에게는 100개의 서로 다른 레벨처럼 느껴지게 할 수 있습니다. 예를 들어 Level1은 프리셋 A, Level2는 프리셋 B, Level3은 프리셋 A+B 연속, Level4는 프리셋 C(보스방) 이런 식으로 섞습니다. 그리고 적 구성만 달리하면 충분히 변화를 줍니다. Watermelon Games 템플릿은 이러한 **조립식 설계**를 염두에 두고 Level Editor를 만든 것으로 보이며, 따라서 프리셋 기능을 적극 활용하는 것이 개발 효율을 높입니다.
* **프리셋 업데이트**: 이미 여러 레벨에 사용된 프리셋을 **수정**하고 싶다면, 어떻게 해야 할까요? 예를 들어 프리셋 A 형태의 방들이 너무 어렵다고 느껴져서 장애물 하나를 빼고 싶다면, 단순히 프리셋 데이터를 고쳐도 그걸로부터 생성된 기존 레벨들은 자동으로 안 바뀔 것입니다 (프리셋은 일종의 청사진이어서, 적용 후에는 독립된 복사본이 됨). 따라서 이러한 경우 개별 레벨을 수정하거나, 또는 레벨 에디터에 "프리셋으로부터 리프레시" 기능이 없다면 일일이 조정해야 합니다. 템플릿에서는 아마 프리셋 동기화까지는 지원하지 않을 가능성이 높으므로, 프리셋 적용 후엔 해당 레벨들은 개별적으로 튜닝하는 것이 좋습니다. 프리셋은 어디까지나 초기 베이스를 빠르게 까는 용도로 생각하면 됩니다.

정리하면, **Room Preset**은 레벨 디자인의 생산성을 극대화하는 도구입니다. 이를 통해 **일관성 있는 디자인 패턴**을 유지하면서도 **개성 있는 조합**을 만들어낼 수 있습니다. 예를 들어 "원형 경기장 + X자 배치 장애물" 프리셋과 "십자형 복도 + 중앙 개방" 프리셋을 적절히 섞으면, 플레이어는 매번 새로운 전투 공간을 접하게 되지만 개발자는 기존 구성요소를 재활용한 것이 됩니다. 특히 모바일 슈팅 게임처럼 레벨 소모가 빠른 장르에서는 이런 프리셋 활용이 매우 중요합니다.

以上으로 Squad Shooter 템플릿의 **Level Creation**에 대한 심층 분석을 마칩니다. 정리하자면, 템플릿은 **전용 Level Editor**를 통해 **월드와 레벨, 방과 오브젝트**를 체계적으로 관리하며, **프리셋 시스템**으로 레벨 디자인을 효율화하고 있습니다. 이러한 툴과 워크플로우를 숙지함으로써 개발자는 방대한 양의 콘텐츠도 짧은 시간 내에 구현 가능하며, 나아가 자기만의 창의적인 레벨 아이디어를 시스템 제약 없이 반영할 수 있을 것입니다. 템플릿 제공 문서를 참고하면서 실제로 여러 레벨을 제작해보면 금세 익숙해지고, 추후 완성된 게임의 유지보수(밸런스 수정이나 신규 레벨 추가 등)도 수월하게 해낼 수 있을 것입니다.

**참고 자료**: Watermelon Games 공식 문서 - Level Creation ([Level Creation | Notion](https://wmelongames.notion.site/Level-Creation-729328e4fcd74f6ab2f87b46b4bfbcc9#:~:text=Home%20Page)) ([Level Creation | Notion](https://wmelongames.notion.site/Level-Creation-729328e4fcd74f6ab2f87b46b4bfbcc9#:~:text=UI%20Store)), World System ([World System | Notion](https://wmelongames.notion.site/World-System-a719e8ab7f5a446c8e25d4e35c687c80#:~:text=Worlds%20group%20levels%20of%20the,assets)) ([World System | Notion](https://wmelongames.notion.site/World-System-a719e8ab7f5a446c8e25d4e35c687c80#:~:text=Down%20below%20you%20will%20see,related%20to%20the%20selected%20world)), Obstacles/Environment 가이드 ([Obstacles | Notion](https://wmelongames.notion.site/Obstacles-bc6922b59fff4090ab9eb1ab35d40ff2#:~:text=,movement%20and%20create%20hiding%20spots)) ([Environment | Notion](https://wmelongames.notion.site/Environment-991c663dc33443659795e9d6be81a6b0#:~:text=,like%20ground%2C%20walls%2C%20and%20gates)), Experience System ([Experience System | Notion](https://wmelongames.notion.site/Experience-System-f88c15e9f92d4cf8a0dea0b2a5035ecd#:~:text=,player%20skin%20in%20our%20case)) ([Experience System | Notion](https://wmelongames.notion.site/Experience-System-f88c15e9f92d4cf8a0dea0b2a5035ecd#:~:text=To%20configure%20how%20much%20experience,folder)) 및 Monetization 가이드 ([Monetization (Ads & IAP) | Notion](https://wmelongames.notion.site/Monetization-Ads-IAP-13bc0f2fe155801f9db2d7b4f0e43b74#:~:text=,developers%20looking%20to%20generate%20revenue)) ([Monetization (Ads & IAP) | Notion](https://wmelongames.notion.site/Monetization-Ads-IAP-13bc0f2fe155801f9db2d7b4f0e43b74#:~:text=)) 등. 본 매뉴얼에서 인용된 내용들은 모두 해당 공식 문서의 공개 자료에 근거하였으며, 궁금한 점은 해당 레퍼런스를 확인하거나 Watermelon Games 커뮤니티에 문의하시기 바랍니다. ([Squad Shooter | Template + Editor by Watermelon Games](https://wmelongames.itch.io/squad-shooter-template-editor#:~:text=This%20fully%20customizable%20shooter%20template,point%20for%20your%20next%20hit)) ([Squad Shooter | Template + Editor](https://www.gameassetdeals.com/asset/274208/squad-shooter-template-editor#:~:text=TEMPLATE%20INCLUDES))

**'Squad Shooter' 템플릿에서 광고 및 IAP 수익화 설정 방법에 대해 다음 항목을 포함한 자세한 매뉴얼을**

1. 지원 광고 네트워크별 설정 방법 (AdMob, Unity Ads, LevelPlay)
2. 보상형 광고 및 전면 광고 설정
3. 인앱 상품(IAP) 추가 및 관리 방법
4. Unity 에디터 내 설정 절차 및 테스트 방법
5. Google Play / iOS 배포 시 주의사항

완료되면 PDF 형태로 정리하여 공유드릴게요.

**Unity Squad Shooter 템플릿 광고/IAP 수익화 설정 매뉴얼**

Unity 기반 **Squad Shooter** 템플릿에서 **광고**와 \*\*인앱 결제(IAP)\*\*를 설정하여 수익화를 구현하는 방법을 안내합니다. 이 매뉴얼은 중급 Unity 개발자를 대상으로 하며, PDF 문서로 정리하기 적합하도록 구성되었습니다. 각 섹션에서는 지원되는 광고 네트워크 연동부터 보상형 광고 버튼 설정, 인앱 상품 구성, 에디터 테스트 방법, 출시 전 점검 사항까지 순차적으로 설명합니다.

**1. 지원 광고 네트워크별 설정 방법**

Squad Shooter 템플릿은 **AdMob**, **Unity Ads**, **LevelPlay** (ironSource 기반) 등의 인기 광고 네트워크를 지원하며, 추가 커스텀 광고 네트워크도 연동 가능하도록 설계되어 있습니다 ([Monetization (Ads & IAP) | Notion](https://wmelongames.notion.site/Monetization-Ads-IAP-13bc0f2fe155801f9db2d7b4f0e43b74#:~:text=,developers%20looking%20to%20generate%20revenue)). 광고를 적용하기 전에 **반드시 해당 광고 네트워크의 SDK를 프로젝트에 통합**하고, **Monetization Settings** 자산에 관련 ID를 정확히 입력해야 합니다. 아래에서는 각 광고 네트워크별 **SDK 통합 및 ID 설정 절차**를 설명합니다.

**AdMob 연동 절차**

* **1) AdMob SDK 가져오기:** Google의 **Mobile Ads SDK** (Google Mobile Ads 플러그인)을 Unity 프로젝트에 추가합니다. 최신 버전의 Unity에서는 Package Manager의 Add Package 기능이나 .unitypackage 파일을 통해 **Google Mobile Ads** 패키지를 가져올 수 있습니다. SDK를 임포트한 후 **External Dependency Manager**(구 Google Play Services Resolver)를 실행하여 Android와 iOS용 의존성을 해결하세요 (보통 플러그인 임포트 시 자동 실행됨).
* **2) AdMob 애플리케이션 및 광고 단위 생성:** AdMob 웹사이트의 **앱 등록** 절차를 통해 Android와 iOS 앱을 각각 생성하고 **애드몹 앱 ID**를 발급받습니다. 또한 **배너**, **전면(Interstitial)**, **보상형(Rewarded)** 광고에 필요한 \*\*광고 단위 ID(Ad Unit ID)\*\*들을 AdMob에서 생성하세요. 이때 **테스트용 광고 단위**도 함께 만들어두면 개발 중 실제 광고 대신 테스트 광고가 나오게 할 수 있습니다.
* **3) Monetization Settings에서 AdMob 설정:** Unity 에디터에서 프로젝트의 Assets/Project Files/Data/Monetization Settings 자산을 찾아 Inspector로 엽니다. 이 **Monetization Settings** 에셋에는 광고 설정 섹션이 있으며, AdMob용 필드가 포함되어 있습니다 ([Monetization (Ads & IAP) | Notion](https://wmelongames.notion.site/Monetization-Ads-IAP-13bc0f2fe155801f9db2d7b4f0e43b74#:~:text=The%20Monetization%20module%20is%20already,file%20at%20the%20following%20path)). 여기에서 **AdMob App ID**와 각 광고 형식별 **Ad Unit ID**를 입력합니다:
  + Android App ID / iOS App ID: 각각 AdMob에서 발급받은 앱 ID를 입력합니다.
  + Banner Unit ID, Interstitial Unit ID, Rewarded Video Unit ID: 앞서 생성한 광고 단위 ID를 입력합니다.

*Tip:* AdMob 앱 ID는 Android의 경우 빌드 시 AndroidManifest에 자동으로 포함되지만, iOS에서는 Info.plist에 GADApplicationIdentifier 키로 추가해주는 것을 권장합니다. 최신 AdMob 플러그인은 자동으로 처리하지만, Xcode 빌드 후 Info.plist에 해당 키가 있는지 확인하세요.

* **4) AdMob 광고 초기화 및 테스트:** 템플릿의 Monetization 모듈은 설정된 AdMob ID를 바탕으로 자동 초기화되도록 구성되어 있습니다. 개발 단계에서는 **테스트 광고**가 나오도록 설정하는 것이 중요합니다. AdMob에서는 실제 광고 단위 ID 대신 Google에서 제공하는 **테스트용 광고 ID**를 입력하거나, AdMob 계정에서 개발 기기를 **테스트 기기**로 등록하면 됩니다. 이렇게 하면 디바이스에서 앱을 실행할 때 테스트 광고가 표시되어 광고 수익에 영향을 주지 않고 안전하게 동작을 검증할 수 있습니다.
* **5) AdMob 통합 확인:** Unity 에디터에서 **Play 모드**로 실행하고, 콘솔 로그를 확인하여 AdMob SDK 초기화 로그가 출력되는지 확인합니다. (에디터에서는 실제 광고가 표시되지 않지만, 초기화 과정 로그를 통해 연동 성공 여부를 알 수 있습니다.) 빌드한 앱을 실제 모바일 기기에서 실행했을 때 배너나 테스트 광고가 정상 노출되면 AdMob 연동이 완료된 것입니다.

**Unity Ads 연동 절차**

* **1) Unity Ads 패키지 활성화:** Unity Ads는 Unity 엔진에 내장된 광고 솔루션입니다. **Unity 2020 이상**에서는 Package Manager에서 **Advertisement** 패키지를 설치하거나, **Services** 창(또는 **Unity Dashboard**)에서 광고 서비스를 활성화해야 합니다. 최신 Unity 버전에서는 Unity Ads가 **LevelPlay**로 통합되었지만, 본 템플릿은 **단일 Unity Ads 사용**과 **LevelPlay(중재 플랫폼) 사용**을 모두 지원합니다. 우선 Unity Ads standalone으로 사용하려면 프로젝트에 \*\*Advertisement 패키지 (Unity Ads SDK)\*\*가 포함되어 있어야 합니다.
* **2) Unity Ads 게임 등록 및 Game ID 확인:** Unity **Developer Dashboard**에서 새로운 프로젝트를 만들고 **Unity Ads** 서비스를 설정합니다. 플랫폼별(Android, iOS) **Game ID**를 발급받을 수 있습니다. (Unity Ads의 Game ID는 해당 플랫폼 앱의 고유 식별자입니다.)
* **3) Monetization Settings에서 Unity Ads 설정:** Unity 에디터의 Monetization Settings 에셋을 열어 **Unity Ads 섹션**을 찾습니다. 여기서 \*\*Unity Ads Game ID (Android)\*\*와 \*\*Game ID (iOS)\*\*를 입력합니다. 또한 **Test Mode** 옵션이 있다면 개발 중에는 활성화하여 **테스트 광고 모드**로 초기화되도록 설정하세요. 일부 Monetization Settings에서는 별도의 **Enable Unity Ads** 토글이 있을 수 있으므로, 있다면 이를 체크하여 Unity Ads 사용을 명시적으로 활성화합니다.
* **4) Unity Ads 초기화 및 테스트:** 템플릿 모듈이 Game ID를 바탕으로 자동으로 Advertisement.Initialize()를 호출하여 Unity Ads를 초기화합니다. **Play Mode**에서는 Unity Ads가 실제 동작하지 않을 수 있으므로, 개발 중에는 Android/iOS 디바이스에 빌드하여 테스트하는 것이 좋습니다. Unity Ads의 **테스트 모드**를 켠 경우 기기에 실제 광고 대신 Unity의 테스트 광고 배너가 나타나므로 안전합니다. 디바이스 로그캣(logcat)이나 Xcode 콘솔을 통해 "Unity Ads initialization complete" 등의 로그를 확인하여 성공 여부를 점검하세요.
* **5) 광고 형식 설정:** Unity Ads에서 **광고 Placement**(광고 소재 슬롯) 설정은 기본적으로 banner, video(전면), rewardedVideo 등이 **기본 ID**로 사용됩니다. 템플릿은 이러한 기본 placement 이름을 사용해 광고를 호출하도록 구성되어 있으므로, Unity Dashboard에서 해당 Placement들이 활성화되어 있는지 확인하세요. (예: **Banner** 광고를 사용하려면 Dashboard에서 Banner 광고를 켜야 함)

**LevelPlay (ironSource) 연동 절차**

* **1) LevelPlay SDK 통합:** **LevelPlay**는 Unity와 ironSource의 통합 광고 중재 플랫폼입니다. LevelPlay를 사용하려면 \*\*ironSource SDK (LevelPlay SDK)\*\*를 프로젝트에 추가해야 합니다. ironSource 공식 사이트나 Unity LevelPlay 패키지를 통해 Unity 프로젝트에 import합니다. (Unity 2022+에서는 Package Manager에서 **Unity LevelPlay** 패키지를 받을 수 있습니다. 또는 ironSource Unity 패키지를 수동으로 다운로드해 임포트할 수도 있습니다.) SDK 임포트 후 **Android용 Gradle 의존성**과 **iOS Pod**가 제대로 설정되도록 **EDM4U**로 dependencies를 확인하세요.
* **2) ironSource(LevelPlay) 계정 설정:** ironSource 관리자 콘솔에서 새 앱을 생성하고 **App Key**를 받습니다. ironSource는 하나의 \*\*앱 키(App Key)\*\*로 해당 앱의 모든 광고를 관리하며, 각 플랫폼(Android, iOS)별 키가 별도로 주어집니다. 또한 ironSource Dashboard에서 **광고 매체(Network)** 설정 및 **광고 단위** 설정을 진행해야 합니다. (예: AdMob을 LevelPlay 중재에 포함하려면 AdMob 계정을 연결하고 AdMob의 Ad Unit ID를 ironSource에 입력하는 등 추가 설정 필요)
* **3) Monetization Settings에서 LevelPlay 설정:** Unity 에디터에서 Monetization Settings를 열고 **LevelPlay (또는 ironSource) 섹션**을 찾습니다. 거기에 \*\*App Key (Android)\*\*와 \*\*App Key (iOS)\*\*를 각각 입력합니다. 만약 Monetization Settings에 LevelPlay 사용 토글이 있다면 이를 활성화합니다. (일부 설정에서는 Unity Ads와 LevelPlay를 동시에 쓰지 않도록 하나만 선택하게 되어 있을 수도 있습니다.)
* **4) 광고 매체 어댑터 통합:** LevelPlay를 통해 여러 광고 네트워크를 \*\*중재(mediation)\*\*할 경우, 각 네트워크에 맞는 어댑터(Adapter)를 프로젝트에 추가로 임포트해야 합니다. 예를 들어 **AdMob을 LevelPlay 안에서 사용**하려면 **ironSource AdMob Adapter**를 추가해야 하고, Unity Ads를 함께 쓴다면 **ironSource Unity Ads Adapter**를 추가해야 합니다. 이러한 어댑터들은 ironSource Unity SDK 패키지에 포함되어 있거나 별도로 제공됩니다. 필요한 어댑터를 모두 추가하고 의존성 처리를 해주세요.
* **5) LevelPlay 초기화 및 테스트:** 템플릿의 Monetization 모듈은 입력된 App Key로 ironSource SDK를 초기화하여 LevelPlay 광고를 로드합니다. 개발 중에는 ironSource에서 제공하는 **Test Suite**(테스트 모드 도구)를 활용하여 다양한 광고 형식을 미리 테스트할 수 있습니다. 안드로이드 기기에서 앱을 실행한 후 Logcat을 보면 ironSource/LevelPlay 초기화 로그와 각 네트워크 초기화 로그가 순차적으로 출력됩니다. iOS에서도 Xcode 콘솔을 통해 초기화 로그를 확인하세요. **주의:** ironSource 기반 LevelPlay에는 별도의 "test mode" 토글은 없고, 대신 **테스트 광고**는 ironSource 계정에서 **테스트 기기 등록** 또는 **테스트용 앱** 설정을 통해 진행합니다. 출시 전에 실제 광고가 잘 나오는지 확인하려면, ironSource 대시보드의 **테스트 모드** 설정을 해제하고 실제 광고 요청이 일어나도록 해야 합니다.

**Note:** Monetization Settings 에셋에서 **Mobile Monetization** 토글을 끌 경우, 위에서 설정한 광고 네트워크 초기화가 수행되지 않고 게임 내 광고/IAP 기능이 모두 비활성화됩니다 ([Monetization (Ads & IAP) | Notion](https://wmelongames.notion.site/Monetization-Ads-IAP-13bc0f2fe155801f9db2d7b4f0e43b74#:~:text=You%20can%20disable%20the%20module,and%20IAPs%20won%E2%80%99t%20be%20available)). 개발 중 일시적으로 수익화 모듈을 끄고 싶다면 이 옵션을 활용할 수 있습니다. 단, 이 경우 **UI 상의 광고/IAP 버튼들은 여전히 보이지만** 동작하지 않으므로, 완전히 제거하려면 UI 요소와 관련 코드를 삭제해야 합니다 ([Monetization (Ads & IAP) | Notion](https://wmelongames.notion.site/Monetization-Ads-IAP-13bc0f2fe155801f9db2d7b4f0e43b74#:~:text=Image%3A%20image)).

**2. 보상형 광고 및 전면 광고 설정**

Squad Shooter 템플릿에는 \*\*보상형 광고(리워드 광고)\*\*와 \*\*전면 광고(Interstitial)\*\*를 손쉽게 활용할 수 있는 모듈이 내장되어 있습니다. **보상형 광고**는 사용자가 광고를 끝까지 시청하면 게임 내 보상을 제공하고, **전면 광고**는 게임 진행 흐름 사이에 전체 화면 광고를 노출합니다. 이 섹션에서는 **UI Store 내 보상형 광고 버튼** 구성과 **보상 지급 로직** 연결 방법, 그리고 전면 광고 표시 설정에 대해 설명합니다.

**보상형 광고 (Rewarded Ads) 설정**

* **UI Store의 리워드 광고 버튼:** Squad Shooter 템플릿에는 **상점(UI Store)** UI에 **"무료 보상"** 형태의 버튼이 기본 제공됩니다. 사용자는 이 버튼을 누르면 **보상형 광고 영상**을 시청하고, 시청 완료 시 \*\*게임 내 재화(코인 등)\*\*를 보상으로 받게 됩니다. 해당 버튼은 템플릿이 제공하는 Monetization 모듈과 연동되어 있으며, **OnClick 이벤트** 등을 통해 **보상형 광고 노출 함수**를 호출하도록 설정되어 있습니다. 별도로 코드를 작성하지 않아도 되도록 **Unity 이벤트**나 **전용 스크립트**로 연결되어 있으므로, 개발자는 **광고 ID만 설정**하면 기본 동작이 이루어집니다.
* **보상 지급 로직:** 보상형 광고가 끝까지 시청되면 **Monetization 모듈이 광고 성공 콜백을 감지**하여 **Rewards System**을 통해 보상을 지급합니다 ([UI Store | Notion](https://wmelongames.notion.site/UI-Store-18bc0f2fe155808193e4d5d7fd72bdfd#:~:text=The%20UI%20Store%20displays%20various,Module%20and%20the%20Rewards%20System)). 예를 들어, 기본 설정에서는 보상형 광고 1회를 성공적으로 시청하면 **일정량의 게임 코인**이 플레이어에게 지급됩니다. 이 로직은 템플릿에 내장되어 있어 별도 구현이 필요 없으며, **광고 시청 실패**나 **사용자가 광고를 건너뛰는 경우** 보상이 지급되지 않도록 처리되어 있습니다. 개발자는 필요에 따라 **지급되는 보상량**을 조정할 수 있습니다. 보상량은 Monetization Settings 혹은 Rewards System 설정에서 **Rewarded Video Reward Amount** 등의 필드로 지정되어 있으니 원하는 값으로 변경하세요. (예: 기본 100코인 지급을 200코인으로 변경)
* **보상형 광고 추가 활용:** UI Store 외에도 보상형 광고를 활용하고 싶다면, 예를 들어 \*\*게임 오버 화면에서 "광고 보고 부활"\*\*과 같은 기능을 구현할 수 있습니다. 이를 위해서는 **Monetization.ShowRewardedAd()** 메서드를 호출하고, 완료 콜백에서 원하는 보상 (부활 처리 등)을 수행하는 식으로 구현합니다. 템플릿의 Monetization 모듈이 제공하는 **간편 API**를 이용하면 동일한 광고 연동 로직을 재사용할 수 있습니다. (이 부분은 직접 코드를 다루는 작업이므로, 필요할 경우 Watermelon Games Discord나 문서를 참고하여 구현하시기 바랍니다.)

**전면 광고 (Interstitial Ads) 설정**

* **전면 광고 자동 노출:** 템플릿은 **전면 광고**가 적절한 게임 진행 시점에 자동으로 표시되도록 기본값이 설정되어 있습니다. 예를 들어, **스테이지 완료 후** 혹은 **몇 라운드마다** 전면 광고가 나타나도록 코드에 구현되어 있습니다. 이러한 로직은 게임의 흐름을 방해하지 않으면서 수익화를 극대화하기 위한 것입니다. 개발자는 Monetization Settings나 관련 스크립트에서 **전면 광고 빈도**를 조절할 수 있습니다. (예: Interstitial Interval = 2로 설정되어 있으면 두ステージ마다 전면 광고 표시) 기본 설정을 확인하여 필요하면 값을 변경하세요.
* **전면 광고 수동 호출:** 만약 특정 이벤트에 전면 광고를 수동으로 표시하고자 한다면, **Monetization 모듈의 ShowInterstitial() 함수**를 호출하면 됩니다. 이 함수 역시 이미 구현되어 있어, 다른 게임 매니저 스크립트에서 불러 쓰기만 하면 됩니다. 예를 들어 보상형 광고처럼 **UI 버튼**을 만들어 전면 광고를 보여주는 것도 가능하지만, 일반적으로 전면 광고는 UI 버튼보다는 **자동 노출**로 사용합니다.
* **노출 조건과 주의사항:** 전면 광고는 사용자가 **의도치 않게 너무 자주 보지 않도록** 간격을 두고 보여주는 것이 좋습니다. 템플릿 기본 설정도 이 점을 고려하여 플레이 경험을 해치지 않을 정도의 빈도로 광고가 나오도록 조정되어 있습니다. 출시 전에 실제 게임 플레이를 여러 차례 테스트하면서 전면 광고 등장 빈도가 적절한지 확인하세요. 과도한 광고는 **유저 이탈**을 유발하고 스토어 **리뷰 평가**에도 악영향을 줄 수 있으므로, 밸런스를 맞추는 것이 중요합니다.

**3. 인앱 상품(IAP) 추가 및 관리**

\*\*인앱 결제 (IAP)\*\*를 통해 사용자에게 **유료 아이템**을 판매함으로써 추가 수익을 올릴 수 있습니다. Squad Shooter 템플릿에는 기본적으로 **광고 제거**, **코인팩 판매**, **스타터 팩** 등의 IAP 상품 구조가 포함되어 있으며, 이를 바탕으로 새로운 상품을 추가하거나 기존 상품을 변경할 수 있습니다 ([UI Store | Notion](https://wmelongames.notion.site/UI-Store-18bc0f2fe155808193e4d5d7fd72bdfd#:~:text=Rewards%20Implemented%20in%20this%20Template)). 이 섹션에서는 **Unity IAP 설정 절차**, **상품 ID 생성 및 등록**, **Monetization Settings에서 상품 관리**, 그리고 **기본 제공 상품(광고 제거/코인팩 등)의 확장 방법**에 대해 설명합니다.

**Unity IAP 초기 설정 및 패키지 구성**

* **1) Unity IAP 패키지 설치:** Unity에서 IAP를 사용하려면 **In-App Purchasing** 패키지를 프로젝트에 임포트해야 합니다. Unity 2019 이후 버전에서는 Package Manager에서 **In-App Purchasing**을 추가하거나, 서비스 창 (Project Settings > Services)에서 **In-App Purchasing**를 켜면 자동으로 패키지가 설치됩니다. Squad Shooter 템플릿의 수익화 모듈은 Unity IAP에 맞춰 제작되어 있으므로, **IAP 패키지가 설치되지 않았다면 반드시 설치**해야 합니다. 설치 후, 에디터 재시작 혹은 재컴파일을 통해 IAP 관련 정의가 인식되는지 확인합니다.
* **2) 앱 스토어 설정 확인:** IAP를 테스트/출시하려면 **Google Play Console**이나 **Apple App Store Connect** 쪽에서도 해당 앱이 **인앱 구매 사용**으로 설정되어 있어야 합니다. Unity Editor에서 **Project Settings > Services > IAP** 탭이 있다면, 거기서 **Android**와 **iOS**의 스토어 설정 (Google Play, Apple App Store)을 확인합니다. 최신 Unity IAP 패키지는 이 과정 없이도 동작하지만, **구형 방식**에서는 Unity 프로젝트를 Unity Services에 연결하고 IAP를 켜야 했습니다. (현재는 **패키지만 임포트하면** 동작하므로 특별한 설정이 없을 수도 있습니다.)

**상품 ID 생성과 Monetization Settings에서 상품 등록**

* **1) 상품 ID 정의:** 판매할 인앱 상품마다 고유한 \*\*상품 ID(Product ID)\*\*가 필요합니다. 이 ID는 **앱 스토어 콘솔**에서 동일하게 생성되어야 하며, 앱 내 코드에서도 사용됩니다. 예를 들어, "Remove Ads" 상품의 ID를 remove\_ads로 정했다면, Google Play Console과 App Store Connect 모두에 동일한 ID로 아이템을 등록해야 합니다. 상품 ID는 보통 **영문 소문자 및 숫자**로 작성하며, **공백 없이** 지정합니다 (예: coin\_pack\_1, no\_ads\_upgrade 등).
* **2) Monetization Settings에서 상품 추가:** Unity 에디터에서 **Monetization Settings** 자산을 열고, **In-App Products** 관련 설정 영역을 찾습니다. 여기에는 기본적으로 템플릿에서 제공하는 몇 가지 상품들이 리스트 형태로 등록되어 있을 것입니다:
  + 예) **Remove Ads** – Product ID: remove\_ads (광고 제거, **비소모성/Non-Consumable**)
  + 예) **Coin Pack** – Product ID: coin\_pack (코인 500개 등, **소모성/Consumable**)
  + 예) **Starter Pack** – Product ID: starter\_pack (초기 패키지: 광고 제거 + 코인 묶음, **비소모성**, 1회 구매 한정)

이 리스트에서 **새로운 상품을 추가**하거나 기존 상품을 편집할 수 있습니다. 새로운 상품을 추가하려면 리스트에 빈 슬롯을 만들고 **Product ID**, **상품 유형(Type)**, **연결될 리워드(Reward)** 등을 입력합니다. 상품 유형은 \*\*소모성(consumable)\*\*인지 \*\*비소모성(non-consumable)\*\*인지 선택하며, Reward는 이 상품을 구매했을 때 제공할 보상 내용입니다. (예: 코인팩이라면 X만큼의 코인을 지급하는 reward를 연결, 광고 제거라면 게임 내 광고 표시를 끄는 reward를 연결)

* **3) Rewards System과 상품 연동:** 템플릿에는 **Rewards System**이 있어 IAP 상품 구매 시 실제 효과를 적용합니다 ([UI Store | Notion](https://wmelongames.notion.site/UI-Store-18bc0f2fe155808193e4d5d7fd72bdfd#:~:text=The%20UI%20Store%20displays%20various,Module%20and%20the%20Rewards%20System)) ([UI Store | Notion](https://wmelongames.notion.site/UI-Store-18bc0f2fe155808193e4d5d7fd72bdfd#:~:text=Rewards%20Implemented%20in%20this%20Template)). Monetization Settings에서 각 Product ID에 매핑된 Reward를 설정하면, Unity IAP 구매 성공 시 해당 Reward가 자동으로 적용됩니다. 예를 들어 remove\_ads 상품의 Reward로 "No Ads"를 지정하면, 구매 직후 게임 내 모든 배너/전면 광고가 비활성화됩니다. coin\_pack\_500 상품의 Reward로 "Currency +500"을 연결하면 구매 시 즉시 500 코인을 유저에게 적립해줍니다. 이처럼 **상품→보상 매핑**을 설정함으로써 **코드를 작성하지 않고도** IAP 구매 결과를 처리할 수 있습니다.
* **4) 상품 현지화 및 가격:** Monetization Settings에는 주로 Product ID와 보상만 설정하고, **가격 및 상품명** 등은 **스토어 콘솔**에서 관리됩니다. Google Play Console이나 App Store Connect에서 해당 Product ID로 아이템을 만들 때 \*\*가격(통화)\*\*과 **상품 표시 이름**을 지정할 수 있습니다. 앱 내부 UI에서 상품명을 표시하려면 별도의 현지화 처리가 필요할 수 있지만, 기본 템플릿 UI는 "Remove Ads", "500 Coins"처럼 영어로 표기되어 있을 것입니다. 필요한 경우 Unity의 Localization 시스템이나 간단히 텍스트 교체를 통해 한국어 등으로 변경할 수 있습니다.

**기본 제공 상품 (광고 제거/코인팩 등) 구성 및 확장 방법**

* **기본 상품 구성:** Squad Shooter 템플릿이 제공하는 **기본 IAP 상품**으로는 앞서 언급한 **광고 제거(No Ads)**, **코인팩(Coin Pack)**, \*\*스타터 팩(Starter Pack)\*\*이 있습니다. "광고 제거"는 사용자가 구매 시 이후부터 게임 내 모든 배너 및 전면 광고를 제거하며, "코인팩"은 인게임 화폐를 충전해주는 일회성 구매입니다. "스타터 팩"은 새로운 유저에게 유용한 패키지로, 한 번 구매하면 **광고 제거 효과와 일정량의 코인**을 함께 제공하는 상품입니다 ([UI Store | Notion](https://wmelongames.notion.site/UI-Store-18bc0f2fe155808193e4d5d7fd72bdfd#:~:text=1,currency%20and%20no%20ads%20rewards)). 이들은 **예시로 포함**된 것이므로, 원한다면 이름이나 내용을 변경할 수도 있습니다. 예를 들어 "Coin Pack"을 여러 개의 패키지(소, 중, 대)로 늘리거나, "Starter Pack"의 구성품(코인 수량 등)을 바꾸는 식으로 조정 가능합니다.
* **새로운 상품 추가:** 기본 상품 외에 **추가로 판매하고 싶은 아이템**이 있다면 자유롭게 상품을 확장할 수 있습니다. 예를 들어, **캐릭터 스킨**이나 **특별 무기**를 IAP로 판매하려는 경우, 다음과 같은 절차로 추가합니다:
  1. **Product ID 생성:** 판매 아이템에 대한 새로운 Product ID를 정합니다 (예: special\_skin\_1).
  2. **스토어 콘솔에 등록:** 해당 ID로 Google/Apple 스토어에 인앱 상품을 만듭니다 (가격 책정 포함).
  3. **Monetization Settings에 추가:** In-App Products 리스트에 새 항목을 추가하고 Product ID를 입력합니다. 상품 유형은 **비소모성**으로 하고, Reward는 해당 스킨을 **잠금 해제**해주는 로직과 연결해야 합니다.
  4. **보상 처리 구현:** 만약 스킨 해제처럼 템플릿의 기본 Rewards System에 없는 유형의 보상이라면, 코인이나 광고 제거와 달리 **직접 코드 구현**이 필요할 수 있습니다. 이 경우 IAP 구매 성공 콜백에서 해당 스킨을 언락하는 코드를 작성하거나, **PlayerPrefs** 등에 구매 상태를 저장하는 방식으로 처리합니다. (템플릿의 구조를 참고하여 유사한 방식으로 확장 가능)
  5. **UI Store에 아이템 표시:** 새로운 상품을 **UI Store 화면에 보여주기 위해**, UI Store의 상품 목록에 해당 아이템을 추가합니다. 템플릿은 상품 항목을 위한 프리팹이나 UI 요소를 가지고 있으므로, 이를 복제하여 새 상품에 맞게 수정합니다. 예를 들어 기존 코인팩 버튼을 복제하여 스킨 아이템 이미지를 넣고 가격 표시를 바꾸는 식입니다. 그런 다음 이 UI 요소를 Monetization Settings의 새 Product와 연결되도록 설정합니다 (예: 버튼의 구매 함수가 새로운 Product ID를 참조).
* **상품 제거/비활성화:** 기본 제공 상품 중 사용하지 않을 것이 있다면, **Monetization Settings에서 해당 Product 항목을 삭제**하거나 비활성화하고, UI Store 씬(또는 프리팹)에서 그 상품에 해당하는 UI 요소를 제거하면 됩니다. 예를 들어 광고 제거 기능을 쓰지 않을 계획이라면 remove\_ads 항목을 지우고, 상점 UI에서 "Remove Ads" 버튼을 없애면 됩니다 ([UI Store | Notion](https://wmelongames.notion.site/UI-Store-18bc0f2fe155808193e4d5d7fd72bdfd#:~:text=More%20info%20about%20reward%20types)). 이렇게 하지 않을 경우 사용자에게 노출되지만 동작하지 않는 버튼이 남을 수 있으므로, 사용하지 않는 상품은 정리하는 것이 좋습니다.

**4. Unity 에디터 내 설정 및 테스트**

광고와 IAP를 설정한 후에는 **Unity Editor**와 **테스트 기기**에서 올바르게 동작하는지 확인해야 합니다. 이 단원에서는 **Unity 에디터 내 Monetization Settings 에셋 편집** 방법과, **테스트 광고 표시** 및 **IAP 구매 시뮬레이션** 방법을 설명합니다. 철저한 테스트를 통해 출시 전에 문제를 발견하고 해결할 수 있습니다.

**Monetization Settings 에셋 설정 검토**

* **경로 및 열기:** Monetization Settings 에셋은 프로젝트 내 경로 Assets/Project Files/Data/Monetization Settings에 위치해 있습니다 ([Monetization (Ads & IAP) | Notion](https://wmelongames.notion.site/Monetization-Ads-IAP-13bc0f2fe155801f9db2d7b4f0e43b74#:~:text=The%20Monetization%20module%20is%20already,file%20at%20the%20following%20path)). 이 에셋을 선택하면 Inspector 창에서 다양한 설정 옵션이 나타납니다. 여기에는 **Mobile Monetization 활성화 토글**, **지원 광고 네트워크별 ID 필드 (AdMob, Unity Ads, LevelPlay)**, **광고 형식별 설정(배너/전면/리워드)**, **인앱 상품 리스트** 등이 포함됩니다.
* **일반 설정 확인:** Monetization Settings 상단의 **General Settings**에서 수익화 모듈이 제대로 활성화되어 있는지 확인합니다. Mobile Monetization 체크박스가 체크되어 있어야 광고/IAP 기능이 작동합니다. 또한 개발 단계에서는 **Test Mode** 관련 옵션이 있는지 찾아보고, 있다면 활성화합니다. (예: *Enable Test Ads*, *Debug Mode* 등의 옵션) 이러한 옵션은 테스트 시 편의를 위해 로그를 추가로 출력하거나, 광고를 테스트용으로 요청하도록 해줍니다.
* **ID 입력 확인:** 앞서 1번 섹션에서 입력한 **광고 App ID 및 Unit ID**, 그리고 3번 섹션에서 설정한 **Product ID**들이 올바르게 기재되었는지 다시 한번 확인하세요. 철자가 틀렸거나 잘못된 ID를 쓰면 광고가 표시되지 않거나 결제가 실패합니다. 특히 AdMob의 App ID와 광고 단위 ID, Unity Ads의 Game ID 등이 **정확하게 복사**되어야 합니다. IAP Product ID도 **대소문자까지 동일**하게 일치해야 스토어에서 인식합니다.
* **배너 광고 위치/크기 설정:** Monetization Settings에는 **배너(Banner) 광고의 위치**나 **크기**를 설정하는 옵션이 있을 수 있습니다 (예: 상단/하단, 스마트 배너 등). 템플릿 기본값으로 배너가 표시되도록 해놓았다면 그 위치를 확인하고, 필요하면 변경 가능합니다. 예를 들어 게임 화면 하단 중앙에 배너가 나오도록 할지, 상단에 나오도록 할지 등을 지정할 수 있습니다. 위치 설정 옵션이 없다면, 기본적으로 화면 하단에 배너가 나타날 것입니다.
* **광고 빈도/조건 설정:** Monetization Settings 또는 관련 스크립트에서 **전면 광고 표시 주기**나 **조건**을 설정할 수 있는 부분이 있는지 체크합니다. 예를 들어 "Show Interstitial After X Rounds" 같은 설정값이 존재할 수 있습니다. 이런 값이 있다면, 게임 디자인에 맞게 숫자를 조정할 수 있습니다. (숫자를 너무 작게 하면 광고 과다 노출, 너무 크게 하면 수익 저하에 유의)

**테스트 광고 출력 확인**

* **에디터 Play 모드 테스트:** Unity 에디터의 Play 모드에서 광고 기능을 간단히 테스트할 수 있지만, **실제 광고는 에디터에서 표시되지 않습니다**. 대신, 중요한 것은 **광고 초기화 과정에 오류가 없는지**를 확인하는 것입니다. Play 모드로 게임을 시작하고 **Console 창**을 보세요. AdMob SDK 또는 Unity Ads SDK 초기화 로그, IAP 초기화 로그 등이 출력되는지 확인합니다. 만약 필요한 SDK가 누락되었다면 콘솔에 오류가 뜰 수 있으므로 즉시 해결해야 합니다.
* **테스트 모드 확인:** 개발 환경에서 **실제 광고 대신 테스트 광고**를 보기 위해, 각 광고 네트워크별 테스트 설정을 확인합니다. 앞서 AdMob의 경우 테스트 광고 ID 사용을 권장했고, Unity Ads는 Monetization Settings의 Test Mode 옵션을 켜두었다면 자동으로 테스트 광고를 요청합니다. ironSource(LevelPlay)의 경우 별도의 테스트 모드가 없으므로, **테스트용 Application Key**를 썼다면 실제 광고가 안 나오므로 주의해야 합니다 (반드시 실제 App Key를 쓰되, 대시보드에서 테스트 기기로 지정).
* **실제 기기에서 광고 확인:** 광고는 최종적으로 **모바일 기기에서** 확인해야 합니다. Android의 경우 개발 PC에서 **Build and Run**하거나 생성된 APK/AAB를 기기에 설치하고, iOS의 경우 Xcode로 빌드하여 디바이스에 설치합니다. 게임을 몇 번 플레이하여 **배너 광고**가 화면에 나타나는지, **전면 광고**가 특정 이벤트 후 출력되는지, **보상형 광고 버튼**을 눌렀을 때 광고가 재생되는지를 확인합니다. 이때 **반드시 테스트 광고**로 나타나는지 확인하여, 실수로 실제 광고 클릭으로 인한 정책 위반이 없도록 합니다. AdMob은 디바이스 로그에 "You are using test ads"라는 메시지가 뜨는지 확인하고, Unity Ads는 광고 크리에이티브에 **테스트 모드 워터마크**가 표시되는지 확인하세요.
* **문제 해결:** 만약 광고가 표시되지 않는다면 몇 가지 점을 점검합니다:
  + 올바른 ID를 입력했는지 (특히 오타 여부)
  + 해당 광고 형식을 대시보드에서 활성화했는지 (Unity Ads Placements 등)
  + SDK가 제대로 초기화되었는지 (로그 확인)
  + 인터넷 연결이 되는지 (기본이지만 간혹 오프라인으로 테스트하면 광고 못 받아옴)
  + AdMob의 경우 AdMob 계정에 앱이 **승인 상태**인지 (초기엔 테스트 광고만 나오다가, 앱이 게시되고 광고 승인되면 실제 광고 등장)

이러한 항목을 모두 확인하고, 필요하면 Watermelon Games에서 제공한 문서나 커뮤니티에 문의하여 해결합니다.

**인앱 결제(IAP) 시뮬레이션 테스트**

* **Unity 에디터에서 IAP 테스트:** Unity IAP 패키지는 **Sandbox(모의) 결제 환경**을 제공합니다. 에디터 **Play 모드**에서 IAP 구매를 시도하면 **Unity IAP Fake Store 창**이 나타나며, 구매를 시뮬레이션할 수 있습니다. UI Store에서 코인팩 등의 구매 버튼을 눌러보세요. 그러면 팝업창이 뜨면서 "Purchase Complete", "Purchase Failed" 등을 선택할 수 있습니다. "결제 성공"을 선택하면, 에디터 콘솔에 구매 성공 콜백 로그가 출력되고 해당 상품의 Reward가 적용됩니다 (예: 코인 증가). "결제 취소/실패"를 선택하면 아무 일도 일어나지 않습니다. 이 Fake Store를 통해 코드상의 로직 (예: 구매 후 UI 업데이트)이 잘 작동하는지 검증합니다.
* **Android 기기 테스트 (내부 테스트):** Google Play 기반 Android에서는 실제로 IAP를 테스트하려면 **Google Play Console의 내부 테스트 트랙**을 이용해야 합니다. 우선 Play Console에 미리 Product ID들을 모두 등록하고, 앱을 **내부 테스트** 또는 **클로즈드 테스트**로 업로드합니다. 그런 다음 테스트 계정(자신의 Gmail 등)을 테스터로 초대하고 기기에 해당 테스트 버전을 설치하여 결제를 진행합니다. 테스트 결제는 실제 과금되지 않으며 **(테스트)** 표시와 함께 진행됩니다. 구매 성공 시 게임에서 보상이 제대로 적용되는지, 구매한 상품이 다시 구매되지 않도록 처리(non-consumable 경우)되는지 확인합니다.
* **iOS 기기 테스트 (Sandbox):** iOS의 IAP 테스트는 **Apple의 Sandbox 환경**에서 이루어집니다. App Store Connect에 상품들을 **승인 요청 상태로 등록**한 뒤, Xcode로 디바이스에 빌드합니다. iOS 디바이스의 설정에서 **Sandbox 테스트 계정**(Apple ID)을 로그인해두고, 앱에서 구매를 시도하면 **테스트용 결제 창**이 뜹니다. 여기서 Apple 테스트 계정으로 로그인하여 결제를 완료하면, 게임에서 동일하게 보상이 적용되는지 확인합니다. (Apple의 Sandbox 결제는 실제 결제와 화면이 유사하게 진행되므로, 테스트 계정이 아닌 일반 계정으로 실수로 결제하지 않도록 주의하세요.)
* **구매 후 처리 확인:** 플랫폼별 테스트에서 **구매 완료 후 시나리오**를 꼼꼼히 점검하세요. 예를 들어:
  + 광고 제거 상품 구매 후 즉시 배너/전면 광고가 사라졌는지 (또는 더 이상 나오지 않는지) 확인합니다.
  + 코인팩 구매 후 코인 잔액이 정확히 늘어나는지 UI를 확인합니다.
  + 이미 구매한 비소모성 상품(예: Remove Ads, Starter Pack)을 다시 구매 시도하면 어떻게 처리되는지 확인합니다. (보통 구매 버튼이 비활성화되거나, "이미 구매한 상품"이라는 메시지가 나오도록 구현)
  + 구매 이력이 앱 재시작 후에도 유지되는지 확인합니다. (비소모성의 경우 PlayerPrefs 등에 영구 저장, 또는 Apple의 경우 영수증 자동 복원 등)

이러한 테스트 과정을 통해 광고와 IAP가 의도대로 작동함을 확인한 후에 다음 단계로 넘어갑니다.

**5. 실제 배포 전 유의사항**

마지막으로, 앱을 Google Play 스토어나 Apple App Store에 **실제 배포하기 전에** 확인해야 할 사항들을 정리합니다. 광고/IAP 관련 설정은 앱 내에서 끝내는 것만으로 완료되지 않고, **스토어 콘솔 측의 설정** 및 **정책 준수 사항**도 챙겨야 합니다. 아래에 **Google Play Console**과 **Apple App Store Connect**에서의 설정, 그리고 **플랫폼별 요구사항** 및 **앱 심사 시 유의사항**을 안내합니다.

**Google Play 콘솔 설정**

* **광고 사용 고지:** Google Play에 앱을 게시할 때 **“광고가 포함되어 있습니다”** 옵션을 반드시 체크해야 합니다. Play Console의 앱 콘텐츠 섹션에 **"Ads"(광고)** 항목이 있으며, 여기서 앱에 광고를 표시하는지 여부를 묻습니다. Squad Shooter 템플릿은 광고를 사용하므로 "예"로 설정합니다. 이를 누락하면 출시 후 앱 정보와 실제 기능이 불일치하게 되어 문제될 수 있습니다.
* **인앱 상품 등록:** 앞서 만든 Product ID들은 **Google Play Console > Monetization > Products > In-app Products** 메뉴에서 모두 등록되고 **활성화(Active)** 상태여야 합니다. 각 상품의 **가격**을 설정하고, **출시** 상태로 변경합니다. 만약 상품을 등록만 하고 **활성화를 안 하면**, 앱에서 구매 시도 시 아이템을 찾지 못해 실패하게 됩니다. (테스트 단계에서 이를 확인했다면 문제없겠지만, 잊지 않도록 합니다.)
* **결제 라이센스 테스트:** 내부 테스트 단계에서 추가로 확인했다면 좋지만, **라이센스 테스트(License Testing)** 설정을 통해 결제 테스트 계정을 등록해두면 개발 중에 유용합니다. Play Console > Settings > License Testing에서 자신의 구글 계정을 추가하면, 결제 시 자동으로 테스트 결제로 처리됩니다.
* **Play 결제 정책 준수:** Google은 **결제 관련 정책**이 엄격합니다. 모든 디지털 상품은 반드시 Google Play 결제를 사용해야 하고, 다른 결제 수단을 앱 내에 안내해서는 안 됩니다. 또한 **환불 정책**이나 **구독(해당 시)** 등에 대한 명시가 요구될 수 있습니다. Squad Shooter 템플릿 기본 구조에서는 이러한 요소가 없지만, 커스터마이징하면서 정책 위반 요소가 생기지 않도록 합니다.
* **기타 권한 및 설정:** AdMob을 사용했다면 **com.google.android.gms.permission.AD\_ID** 권한이 AndroidManifest에 포함되어야 하며, 최신 Google Play 정책에 따라 이 권한을 선언해야 합니다. 최신 Google Mobile Ads SDK를 사용하면 자동으로 추가되지만, 빌드된 Android App Bundle(AAB)을 Play Console에 업로드할 때 **Privacy > Permission Declaration**에서 Advertising ID 사용에 대한 질문에 답하고 필요한 경우 설문을 제출해야 할 수 있습니다. (일반적으로 "광고 제공을 위해 사용"이라고 명시)
* **앱 서명 및 업데이트:** 광고/IAP를 통합한 후에는 **버전 코드를 증가**시키고 새로운 빌드를 준비하여 Play Console에 업로드해야 합니다. 출시 전 **Closed Testing**이나 **Open Testing** 트랙을 거쳐 실제 사용자 환경에서 문제없음을 확인하면 안정성을 높일 수 있습니다.

**App Store Connect 설정**

* **인앱 구매(IAP) 등록:** Apple App Store Connect의 **Features > In-App Purchases** 섹션에서 모든 Product ID를 등록해야 합니다. 각 아이템에 **참고용 이름 (Display Name)**, **Product ID**, **가격** 등을 입력하고, **App Store 리뷰용 스크린샷**을 업로드한 후 \*\*제출(Submit)\*\*하여 **승인**을 받아야 합니다. (인앱 구매 항목은 별도로 Apple의 심사를 거치며, 보통 앱 심사와 함께 진행됩니다.) 인앱 상품이 "Ready to Submit" 상태이고 앱과 연결되지 않은 경우, 앱 심사에 통과하더라도 해당 IAP는 작동하지 않으니 주의합니다.
* **앱 추적 투명성(ATT) 권한:** 만약 광고 SDK(특히 AdMob이나 Unity Ads)가 \*\*사용자 추적 (IDFA)\*\*을 활용하는 경우, iOS14+ 환경에서 **App Tracking Transparency** 권한 요청이 필요합니다. Apple 심사에서는 앱이 사용자에게 **추적 권한 팝업**을 표시하고, **NSUserTrackingUsageDescription** 문자열을 Info.plist에 포함하고 있는지를 확인합니다. 템플릿 자체에는 이 권한 요청 로직이 포함되지 않았을 수 있으므로, **광고 SDK의 문서**를 따라 ATT 요청을 구현하거나, **Info.plist에 적절한 설명문구**를 추가해야 합니다. 예: "이 앱의 광고 제공 개선을 위해 사용자의 기기 식별자를 활용할 수 있습니다." 등의 문구를 **NSUserTrackingUsageDescription**에 넣습니다.
* **Sign in with Apple 등 요구사항:** (해당사항이 있을 경우) Apple은 앱 내 로그인이나 계정 기능이 있을 때 **Sign in with Apple**을 요구하는 등 추가 가이드가 있습니다. Squad Shooter는 별도 계정 체계가 없으므로 직접적인 상관은 없지만, 만약 게임에 소셜 로그인 등을 추가했다면 이런 요구사항도 검토해야 합니다. **앱 권한** 측면에서는 일반적인 게임이 요구하는 카메라, 마이크 등이 없으므로 광고/IAP 관련해서는 ATT 권한 정도만 확인하면 됩니다.
* **앱 심사 노트 작성:** App Store에 앱을 제출할 때 \*\*심사팀에 제공하는 노트(Review Notes)\*\*에 광고 및 IAP에 대한 설명을 간략히 적어두는 것이 좋습니다. 예를 들어, "이 앱은 AdMob 및 Unity Ads를 통해 광고를 제공합니다. 또한 앱 내 구매로 광고 제거 및 코인팩을 판매합니다."라고 명시해두면, 심사 담당자가 앱 기능을 이해하는 데 도움이 됩니다. 심사에서는 **테스트 계정** 제공을 요구하기도 하는데, IAP의 경우 Sandbox로 충분하지만, 만약 게임 진행에 로그인이 필요하거나 특별한 조건이 있다면 데모 계정 정보를 제공해야 합니다.
* **테스트플라이트(TestFlight)로 최종 테스트:** 출시 전 마지막으로, Xcode에서 **TestFlight**에 빌드를 올려 실제 디바이스에서 **배포용 프로필**로 한번 더 테스트하는 것을 권장합니다. 이때 In-App Purchase가 Sandbox 환경에서 잘 작동하는지, 광고가 실제 사용자 환경에서 문제없이 나오는지 (테스트 광고 모드 해제 후) 확인합니다. TestFlight에서는 내부 테스터들에게 앱을 배포하여 다양한 기기에서 확인할 수 있으므로, 크래시나 레이아웃 이슈도 함께 점검할 수 있습니다.

**앱 리뷰 및 정책 유의사항**

* **광고 콘텐츠 제한:** 광고 플랫폼들은 자체 **콘텐츠 가이드라인**이 있습니다. 너무 폭력적이거나 선정적인 광고는 게재되지 않도록 필터링 옵션을 AdMob이나 Unity Ads 대시보드에서 설정할 수 있습니다. 또한 앱 스토어 심사에서는 **아동용 앱**인데 성인 대상 광고가 나오면 문제가 될 수 있으므로, 대상 연령에 맞는 광고 필터를 적용해야 합니다. (예: AdMob의 **DFP 설정**이나 Unity Ads의 **Content Rating** 설정 활용)
* **과도한 광고로 인한 거부:** Apple은 앱이 **과도하게 광고에 의존**하여 UX를 해친다고 판단되면 거부할 수 있습니다. 예를 들어, 사용자가 아무 행동도 못할 정도로 연속해서 전면 광고가 나온다거나 하면 4.0 지침 위반으로 거절될 수 있습니다. Squad Shooter 기본 설정은 이러한 기준을 만족하도록 적절히 광고 빈도를 조정했지만, 혹시 변경했다면 **심사 지침에 어긋나지 않게** 밸런스를 유지합니다.
* **인앱 구매 Restore 기능:** iOS의 경우 \*\*비소모성 상품(예: Remove Ads)\*\*은 **복원(restore)** 기능을 제공해야 합니다. 템플릿에 기본 제공되었을 수도 있지만, 없다면 구현을 고려해야 합니다. (예: 설정 화면에 "구매 복원" 버튼을 두어 RestorePurchases()를 호출) Apple 심사에서는 복원 기능이 없다고 크게 문제 삼지는 않지만, 사용자 경험상 필요합니다.
* **앱 내 구매 안내 문구:** Google Play와 Apple 모두, **유료 아이템**이 존재하면 이에 대한 안내를 앱 내에 명확히 해야 합니다. 예를 들어 튜토리얼이나 스토어 화면에 통화(코인)를 실제 돈으로 구매할 수 있다는 것을 명시하고, 가격 표시를 정확히 하는 것이 좋습니다. 특히 Apple은 **로컬 통화 기호와 단위**가 제대로 표시되는지를 보므로, Localization에 유의합니다 (예: "₩1,100" 또는 "$0.99" 등이 자동으로 표시되도록 함).
* **마켓 배포 전 최종 점검:** 모든 설정이 끝났다면, **버전명을 갱신**하고 **릴리즈 노트**를 작성한 후 스토어에 앱을 제출합니다. 제출 후 개발자로서 할 일은 앱 심사에 대응하는 것입니다. 광고/IAP와 관련해 추가로 문의가 오면 성실히 답변하고 필요한 자료를 제공하면 됩니다. (예: 저작권 관련 문서 등)

以上의 과정과 점검사항을 모두 마치면, Unity Squad Shooter 템플릿 기반 게임의 **광고 및 인앱 결제 수익화 설정**이 완료됩니다. 템플릿의 공식 문서(Watermelon Games Notion)와 가이드를 참고하여 진행했다면 큰 문제 없이 수익화 기능을 구현할 수 있을 것입니다. 이제 앱을 출시하고 유저의 반응을 모니터링하면서, 필요에 따라 광고 빈도나 상품 구성 등을 조정하여 **지속적으로 최적화**해 나가시기 바랍니다. Good luck!