

〈BulletPro〉

1. 한 줄 요약 / 핵심 장점

- BulletPro는 Unity에서 탄막슈팅(Bullet Hell, Danmaku) 패턴을 손쉽게 구현할 수 있도록 하는 상용 에셋입니다.
- 프로그래밍 없이 비주얼 에디터 + 프리셋 패턴 시스템으로 수천 발의 탄막을 최적화하여 표현 가능하며, RPG Maker처럼 직관적인 패턴 구성과 Unity 친화적인 API를 함께 제공합니다.



2. 적용대상

- 탄막슈팅/보스전 패턴 제작이 많은 슈팅 게임, 액션 RPG, 보스레이드 연출
- 수천 발의 탄을 발사해도 퍼포먼스를 유지해야 하는 경우
- 디자이너/레벨 디자이너가 직접 노코드 방식으로 패턴을 만들고 싶은 프로젝트

A practical workspace for your bullets

The image displays the Bullet Pro software interface. On the left, the 'Bullet Hierarchy' panel shows a tree structure with 'Root Emitter', 'Pattern', 'Shot', and 'Bullet'. Below it is the 'Recycle Bin' panel. The main 'Pattern' editor is open, showing a 'Play at Start' checkbox, a 'Pattern Tags' field, and an 'Instruction List' with various actions like 'Play Audio', 'Begin Loop', 'Shoot', 'Fade', 'Wait', and 'End Loop'. The central workspace shows a grid of colored squares representing a pattern. To the right, a large preview window shows a complex bullet pattern of many small, colorful arrows. Below the main workspace, a smaller preview window shows a game scene with a character and a large, colorful, swirling bullet pattern. The text 'Suitable for any 2D gameplay!' is written in a stylized font. The 'Bullet Pro' logo is in the bottom right corner.

Suitable for any 2D gameplay!

Bullet Pro

3. 설치 & 시작

3.1 Unity Asset Store> BulletPro 구매 후 Import

3.2 메뉴에 BulletPro 관련 툴/윈도우가 추가됨

3.3 샘플 씬('Demo Scenes/') 실행> 기본 탄막 패턴 확인 가능

The easiest possible workflow


Step 1 - Store your work in profile assets

Step 2 - Use only two components: Bullet Emitter and Bullet Receiver

The image displays the BulletPro Unity package interface. It features a dark background with a colorful logo on the left. The main content area shows two steps in the workflow. Step 1, 'Store your work in profile assets', is illustrated with a screenshot of the Unity Package Manager showing the 'Bullet Pro' package installed. Step 2, 'Use only two components: Bullet Emitter and Bullet Receiver', is illustrated with two screenshots: one of the 'Bullet Emitter' component in the Unity Inspector and another of the 'Bullet Receiver' component in the Unity Inspector. The 'Bullet Emitter' component shows settings for 'Emitter Profile' and 'Pattern Origin'. The 'Bullet Receiver' component shows settings for 'Collider Type', 'Hitbox Size', 'Hitbox Offset', 'Kill Bullet On Collision', 'Max Simultaneous Collisions Per Frame', 'Collision Tags', and 'On Hit By Bullet' events. The 'Bullet Pro' logo is visible in the bottom right corner.

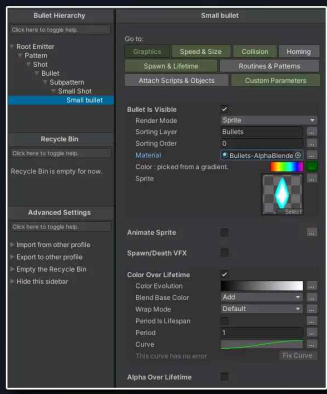
4. 핵심 구성요소

컴포넌트	설명
Bullet Emitter	탄을 발사하는 발사체 생성기
Bullet Pattern	발사 각도, 속도, 곡선, 가속도, 지연 등 패턴 정의
Bullet Pool	수천 발의 탄을 풀링(Pooling)으로 관리
Bullet Behaviour	각 탄의 개별 동작(곡선, 유도, 충돌 후 반응) 제어
Bullet Collision Handler	피격 판정 및 충돌 처리
Pattern Editor	GUI 기반 시각적 에디터로 패턴 구성





Freely editable bullet graphics

Easily find every parameter for how your bullet looks:



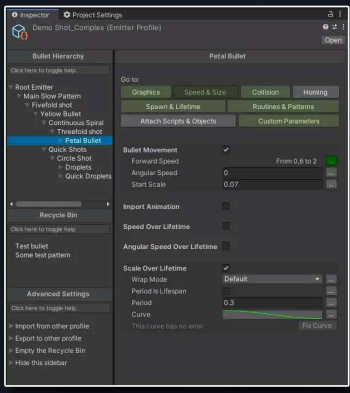
- Sprite
- Material
- Particle Effects
- Animations
- Color, Alpha, Gradients, Curves...






Elaborate behaviours, without coding

Unique **Bullet Hierarchy** system: easily locate any parameter




- Fully editable in **Play Mode**
- Randomize all parameters
- Bind them to game difficulty
- Infinite emitter nesting
- And so much more...




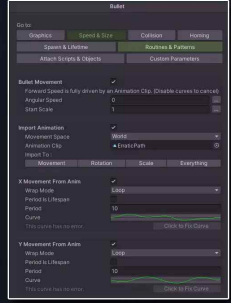
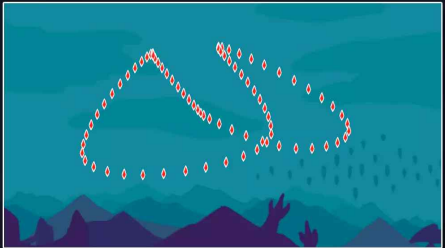
5. 주요 기능


- 비주얼 패턴 에디터: 드래그&드롭 방식으로 패턴(방사형, 웨이브, 스파이럴, 유도탄 등) 설정
- 수학 기반 파라미터: 각도, 속도, 회전, 가속, 주기적 파형 등을 수치로 제어
- Bullet Pooling: GPU/CPU 부담 최소화, 수천 발 동시 처리
- 애니메이션/이펙트 연동: Shader나 VFX Graph과 연계 가능
- API 연동: C#에서 패턴 시작/중지/변경 가능
- 데모 패턴 제공: 보스전/스테이지 예제 다수 포함



Convert AnimationClips to bullet behaviours!

1 - Animate any object in Unity
2 - Drag the clip, click "Import"
3 - Profit!

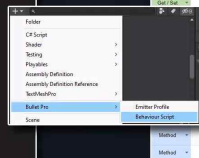





Fully documented, and scriptable API

Has quickstart guides + full manual + online script reference.

```
// Use this for initialization (instead of Start)  
public override void OnBulletBirth ()  
{  
    base.OnBulletBirth();  
    // Your code here  
}  
  
// Update is (still) called once per frame  
public override void Update ()  
{  
    base.Update();  
    // Your code here  
}  
  
// This gets called when the bullet dies  
public override void OnBulletDeath ()  
{  
    base.OnBulletDeath();  
    // Your code here  
}
```



Access	Type	Name	Use
Read	float	baseSpeed	The base forward speed of the bullet
Read	float	baseAngularSpeed	The base angular speed of the bullet, in degrees per second
Read	float	baseScale	The base scale of the bullet, directly applied to its Transform
Read	float	angularOffset	Evolution of the bullet's angular speed
Read	float	scaleOffset	Evolution of the bullet's scale
Read	float	rotationOffset	How the bullet moves along the X axis if an AnimationClip has been loaded into it. If it is enabled, baseSpeed isn't used at all
Read	float	translationOffset	How the bullet moves along the Y axis if an AnimationClip has been loaded into it. If it is enabled, baseScale isn't used at all
Read	float	rotationOffset	How the bullet moves along the Z axis if an AnimationClip has been loaded into it. If it is enabled, baseScale isn't used at all
Read	float	translationOffset	How the bullet moves along the X axis if an AnimationClip has been loaded into it. If it is enabled, baseSpeed isn't used at all
Read	float	rotationOffset	How the bullet moves along the Y axis if an AnimationClip has been loaded into it. If it is enabled, baseScale isn't used at all
Read	float	translationOffset	How the bullet moves along the Z axis if an AnimationClip has been loaded into it. If it is enabled, baseScale isn't used at all
Read	float	rotationOffset	How the bullet moves along the X axis if an AnimationClip has been loaded into it. If it is enabled, baseSpeed isn't used at all
Read	float	translationOffset	How the bullet moves along the Y axis if an AnimationClip has been loaded into it. If it is enabled, baseScale isn't used at all
Read	float	rotationOffset	How the bullet moves along the Z axis if an AnimationClip has been loaded into it. If it is enabled, baseScale isn't used at all
Read	float	translationOffset	How the bullet moves along the X axis if an AnimationClip has been loaded into it. If it is enabled, baseSpeed isn't used at all
Read	float	rotationOffset	How the bullet moves along the Y axis if an AnimationClip has been loaded into it. If it is enabled, baseScale isn't used at all
Read	float	translationOffset	How the bullet moves along the Z axis if an AnimationClip has been loaded into it. If it is enabled, baseScale isn't used at all



6. 실무 설정 / 워크플로우

6.1 Bullet Emitter 생성

- 씬에 'BulletEmitter' 프리팹 배치
- Inspector에서 Pattern 선택

6.2 패턴 편집

- Pattern Editor 열기
- "Radial Shot" 선택 후 탄 개수=36, 주기=0.5

초 설정〉원형탄막

6.3 패턴 재생

- 실행 시 자동 발사
- 필요 시 스크립트에서

'StartPattern("RadialShot")' 호출

7. 코드 예제

7.1 패턴 실행

```
using BulletPro;
using UnityEngine;

public class BossPatternTrigger : MonoBehaviour
{
    public BulletEmitter emitter;

    void Start()
    {
        // 보스 시작 시 원형 탄막 패턴 실행
        emitter.Play("RadialPattern");
    }

    void Update()
    {
        // 특정 키로 패턴 전환
        if (Input.GetKeyDown(KeyCode.Alpha1))
            emitter.Play("SpiralPattern");
    }
}
```

7.2 동적 파라미터 변경

```
// 실행 중 각도를 회전시켜 탄막을 나선형으로
emitter.currentPattern.angle += Time.deltaTime * 30f;
```

8. 베스트 프랙티스

- 폴링 사용: BulletPro 자체 폴링 시스템 사용으로 메모리 누수 방지
- 패턴 이름 관리: “Boss1_Phase1_Spiral” 처럼 체계적으로 관리하면 유지보수 용이
- 피드백 연동: MMFeedbacks, Cinemachine 카메라 셰이크 등과 연결> 보스 패턴의 임팩트 극대화
- Difficulty Curve: 패턴 속도/탄 개수/간격을 변수화해 난이도 조정 가능

9. 확장성

- API 활용: 특정 HP/Phase 조건에서 패턴 전환> 보스 전 시퀀스 구현
- 커스텀 Bullet Behaviour: 탄의 이동/분열/사라짐 로직을 직접 스크립트로 확장 가능

[요약정리]

BulletPro는 탄막 패턴 제작의 복잡성을 대폭 줄여주는 상용 툴입니다.

특히 퍼포먼스 문제 없이 수천 발의 탄막 구현이 가능하다는 점이 강점입니다.