Combat Graph Full Documentation

v1.0.0 by LissaGames

Welcome

Welcome to **Combat Graph**, a lightweight and flexible Unity library that lets you create and manage combat entities using a powerful graph-based interface.

Whether you're building an RPG, strategy game, or any combat-heavy experience, Combat Graph makes it easy to define combat logic, apply buffs, and execute attacks — all with a clean, modular design.

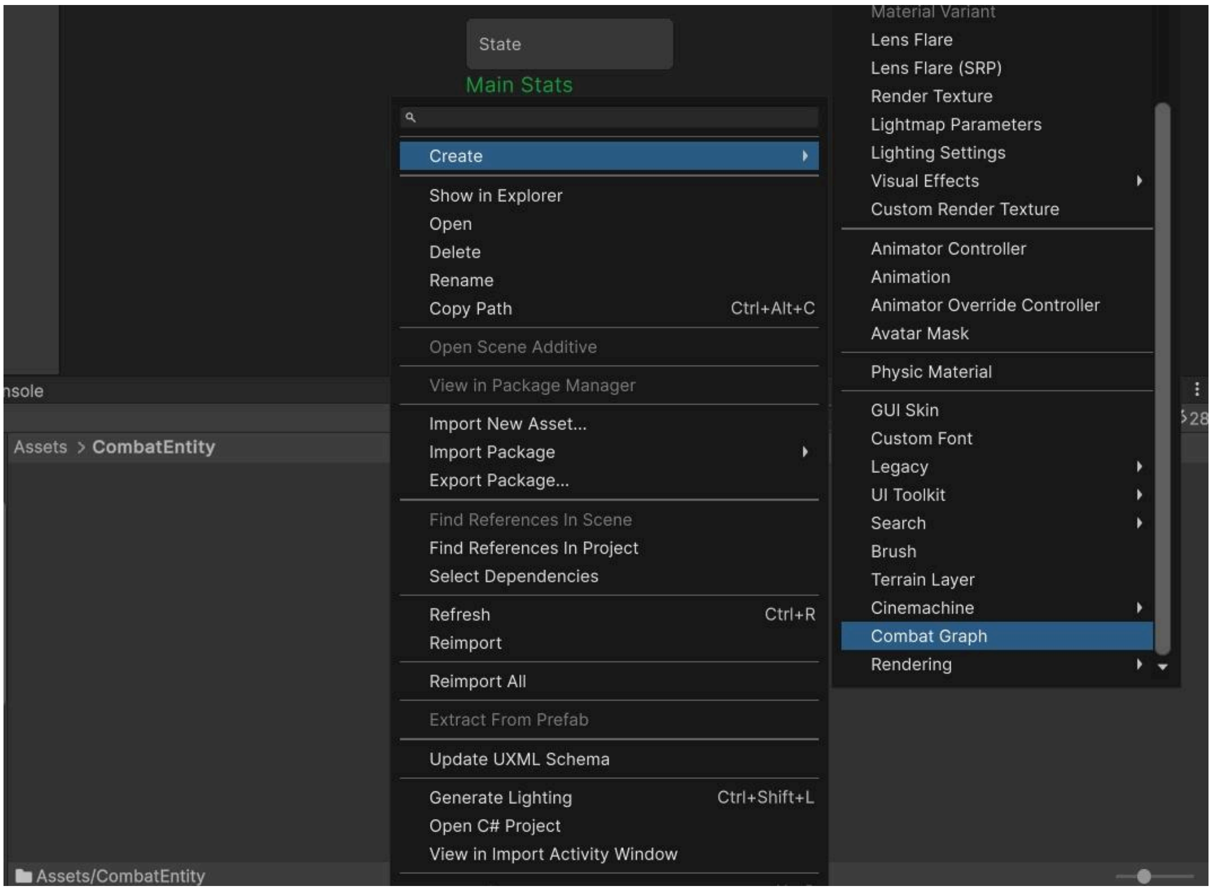
What is it?

Combat Graph allows you to:

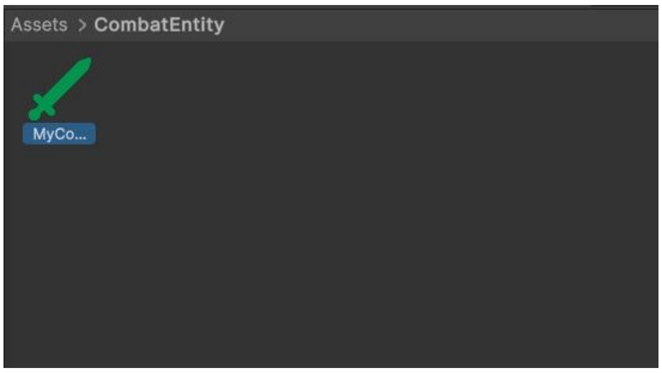
- **Design combat entities using a visual Graph Interface**. Create combat units by adding nodes for stats, abilities, buffs, and more.
- **Handle Combat Automatically**. Use simple methods like Attack or TakeDamage — the system takes care of all internal calculations.
- **Treat Entities as One Unified Component**. Once set up, a combat entity is just a single component — easy to use, clone, and extend.

Quickstart

Create your first Combat Entity



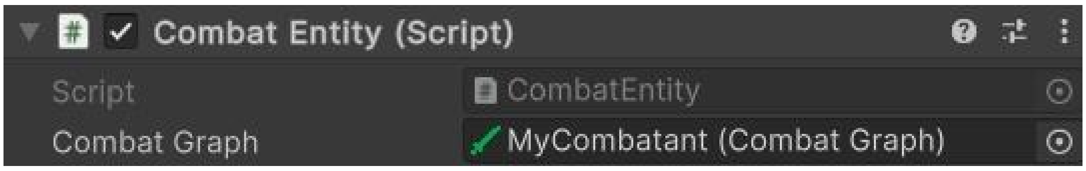
Start your journey with creating Combat Grath asset in project. You can rename it to MyCombatant.



Combat Entity Component

Add the Combat Entity component to your GameObject.

In the Inspector, drag your `MyCombatant` asset into the `Combat Graph` field.
That's it — your combat entity is now set up and ready.


Combat Entity (Script)

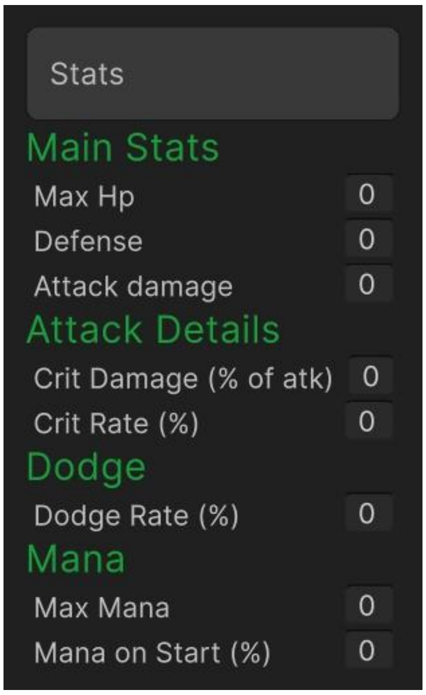
Script	CombatEntity
Combat Graph	MyCombatant (Combat Graph)

Combat Graph Editing

To edit your Combat Graph asset (`MyCombatant`), simply double-click it.

In the window that appears, you'll see the **Main node: Stats**, which represents the default stats of your entity — such as HP, Defense, Crit Rate, and more.

You can freely modify any of these fields to customize your entity.


Stats

Main Stats

Max Hp	0
Defense	0
Attack damage	0

Attack Details

Crit Damage (% of atk)	0
Crit Rate (%)	0

Dodge

Dodge Rate (%)	0

Mana

Max Mana	0
Mana on Start (%)	0

To add Attacks or Buffs right click on window plane and select required option. Create unique Attack names for all added Attacks.

Basic Scripting

To start interaction with your Entity add any C# script on the scene and give the reference of Combat Entity to it. A reference to the same object or to a different object.

Reference to the same object

```
[RequireComponent(typeof(CombatEntity))]
class MyCombatEntityController : MonoBehaviour
{
    private CombatEntity combatEntity;

    private void Awake()
    {
        combatEntity = GetComponent<CombatEntity>();
    }
}
```

Reference to a different object

class MyCombatEntityController : MonoBehaviour

```
{
    [SerializeField]
    // in inspector reference your Combat Entity
    private CombatEntity combatEntity;
}
```

The main methods of Combat Entity are Attack and TakeDamage this is all you need for simple game interaction between combatants.

To perform an attack, call the Attack method on the attacking entity.

You can also trigger any necessary animations at this point.

Once the attack is executed, determine which entities should receive damage, and call TakeDamage on each of them.

```
private void BlowAttack()
{
    var attackData = combatEntity.Attack("Blow");

    if (!attackData.isAvalible) return;

    animator.Play("Blow");

    foreach(var entity in blastedEnemies)
    {
        entity.TakeDamage(attackData.damageProduced,
combatEntity, "Blow");
    }
}
```

The Attack method returns data about the performed attack — including damage, critical hits, and other combat-related details.

The TakeDamage method returns information about the final damage taken after applying defense, shields, dodges, and other modifiers.

AutoAttack (Recommended for AI)

You also can use StartAutoAttack method for your Enemy AI systems. It triggers your auto attack holder function each time the entity performs an attack.

No-Damage Attacks

Since some attacks in games don't deal direct damage (e.g. summoning or triggering effects), you can use **No-Damage Attack** or **No-Damage Mana Attack** to handle these special events.

Graph Editing

Stats Node

Stats Node describes main parameters of Combat Entity.

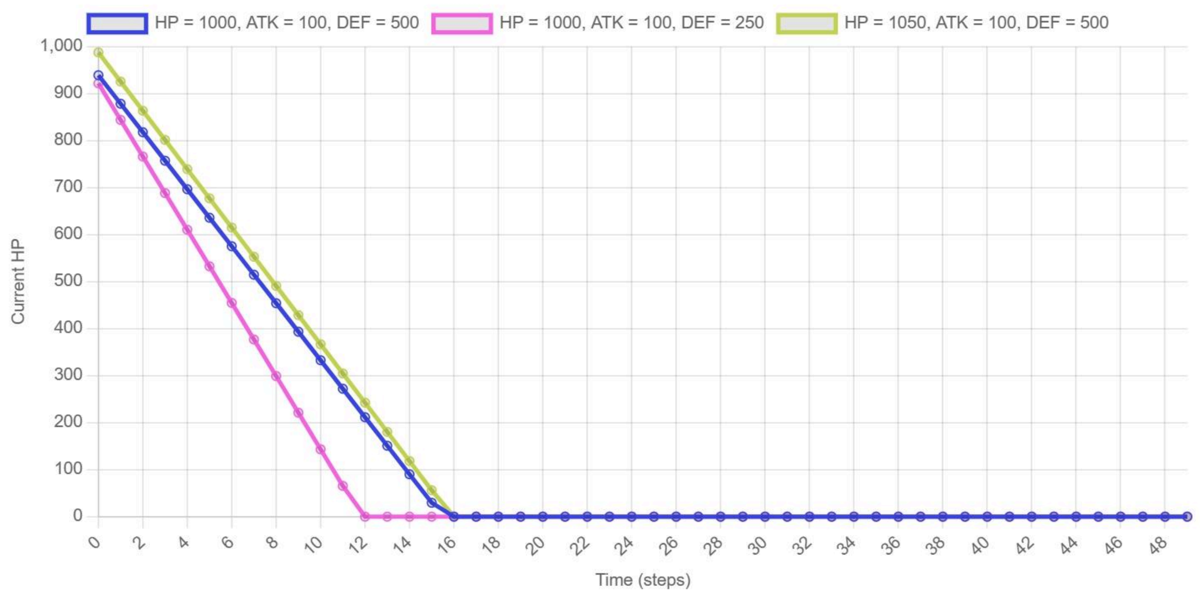
Field Name	Type	Description
MaxHp	int	Maximum Hp value.
Defence	int	Value that reduces the damage taken.
Attack Damage	int	Basic damage value.
Crit Damage	int	Percent of basic damage that added on crit.
Crit Rate	int	Chance to critically strike when casting an attack.
Dodge Rate	int	Chance to completely avoid incoming damage.
Max Mana	int	Maximum mana value.
Mana On Start	int	Percent of Max Mana the entity starts with.

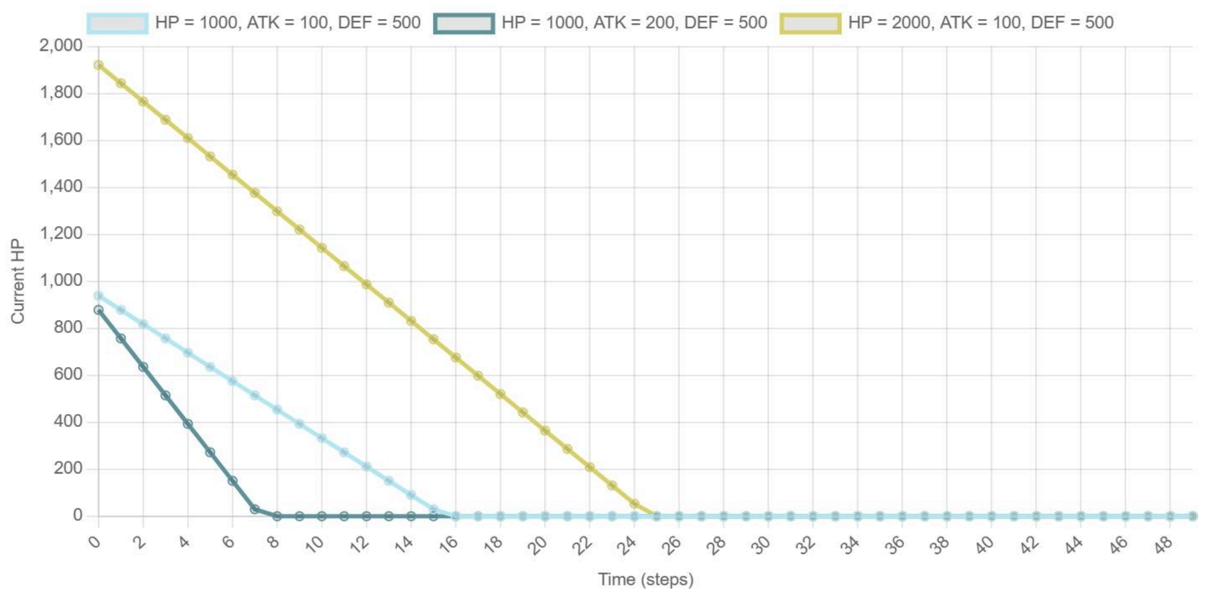
Defense Calculation

The model uses this defense formula.

$$TotalDamage = Atk \times e^{-Def \div MaxHP}$$

Below are graphics showing different values of HP, Defense, and Attack (in terms of damage over time), allowing you to compare the rate at which HP decreases for various setups.





Buffs

Buffs apply positive effects to the entity. Some buffs take effect immediately upon initialization, while others activate in response to specific actions. The buff will be summarized each time an action is triggered, until a specific limit is reached.

Multiple instances of the same buff **can exist** and function independently in the Combat Graph.

Field Name	**Description**
Attack Speed Increase	Boosts attack speed by a percentage.
Crit Damage Increase	Boosts critical hit damage by a percentage.
Crit Rate Increase	Boosts critical hit rate by a percentage.
Damage Increase	Increase damage, Percent or Liner mods.
Defense Increase	Increase defence, Percent or Liner mods.
Dodge Rate Increase	Increase dodge rate by a percentage.
Healing On Action	Increase current HP on selected action.
Healing Persistent	Constantly increase current HP.
Max HP Increase	Increase Max HP on selected action.
Invincibility	Add temporary invulnerability on selected action.
Mana Regeneration On Action	Increase current Mana on selected action.
Mana Regeneration Persistent	Constantly increase current Mana.
Set Damage Cap	Add a minimum damage limit.

Attack Speed Increase

Increases attack speed, which reduces all cooldowns inversely proportional to the boost.

Field Name	**Type**	**Description**
Event Type	Event Type	Trigger of boost.
Attack Speed Multiplier	int	For example, at 100%, attack speed is doubled.
Max Speed Boost Multiplier	int	If the current buff reaches this limit, it will remain at the limit until reset.
Effect Chance	int	Each time the triggering action occurs, the buff has this chance to be applied.

Crit Damage Increase

Increases the base Critical Damage stat by the Additional Critical Damage value. If the base Critical Damage is 20% and the Additional Critical Damage is 30%, the total Critical Damage will be 50%.

Field Name	**Type**	**Description**
Event Type	Event Type	Trigger of boost.
Additional Crit Damage	int	Boosts critical hit damage by a percentage.
Max Additional Crit Damage	int	If the current buff reaches this limit, it will remain at the limit until reset.
Effect Chance	int	Each time the triggering action occurs, the buff has this chance to be applied.

Crit Rate Increase

Increases the base Critical Rate state by the Additional Critical Rate value. If the base Critical Rate is 10% and the Additional Critical Rate is 60%, the total Critical Rate will be 70%.

Field Name	**Type**	**Description**
Event Type	Event Type	Trigger of boost.
Additional Crit Rate	int	Boosts critical hit rate by a percentage.
Max Additional Crit Rate	int	If the current buff reaches this limit, it will remain at the limit until reset.
Effect Chance	int	Each time the triggering action occurs, the buff has this chance to be applied.

Damage Increase

Increases base Attack Damage. If the Addition Type is *Linear*, the Additional Damage is directly added to the base Attack Damage. If the Addition Type is *PercentageBased*, the Additional Damage is calculated as a percentage of the base Attack Damage and added to it.

If there are multiple Damage Increase buffs, some *Linear* and some *PercentageBased*, the total damage will be calculated using the following formula.

$$totalDamage = (baseATK + linearBoost) \times (1 + percentBoost \div 100))$$

Field Name	**Type**	**Description**
Event Type	Event Type	Trigger of boost.
Addition Type	Addition Type	Method of increase: linear or percentage-based.
Additional Damage	int	Increases base ATK Damage based on the specified Addition Type.
Max Additional Damage	int	If the current buff reaches this limit, it will remain at the limit until reset.
Effect Chance	int	Each time the triggering action occurs, the buff has this chance to be applied.

Defense Increase

Increases Defence. If the Addition Type is *Linear*, the Additional Defense is directly added to the Defence. If the Addition Type is *PercentageBased*, the Additional Defense is calculated as a percentage of the default Defence.

If there are multiple Defense Increase buffs, some *Linear* and some *PercentageBased*, the total Defenses will be calculated using the following formula.

$$totalDefense = (baseDef + linearBoost) \times (1 + percentBoost \div 100))$$

Field Name	Type	Description
Event Type	Event Type	Trigger of boost.
Addition Type	Addition Type	Method of increase: linear or percentage-based.
Additional Defense	int	Increases base ATK Damage based on the specified Addition Type.
Max Additional Defense	int	If the current buff reaches this limit, it will remain at the limit until reset.
Effect Chance	int	Each time the triggering action occurs, the buff has this chance to be applied.

Dodge Rate Increase

Increases the base Dodge Rate state by the Additional Dodge Amount value. If the base Dodge Rate is 10% and the Additional Dodge Amount is 60%, the total Dodge Rate will be 70%.

Field Name	**Type**	**Description**
Event Type	Event Type	Trigger of boost.
Additional Dodge Amount	int	Boosts dodge rate by a percentage.
Max Additional Dodge Amount	int	If the current buff reaches this limit, it will remain at the limit until reset.
Effect Chance	int	Each time the triggering action occurs, the buff has this chance to be applied.

Healing On Action

Simply restores HP after an action occurs.

Field Name	Type	Description
Event Type	Event Type	Trigger of boost.
Healed HP Amount	int	Amount of HP that will be restored.
Effect Chance	int	Each time the triggering action occurs, the buff has this chance to be applied.

Healing Persistent

Adds persistent regeneration to the Entity for its entire lifetime.

Field Name	Type	Description
Event Type	Event Type	Trigger of boost.
Healed HP Amount	int	Amount of HP that will be restored.
Effect Chance	int	Each time the triggering action occurs, the buff has this chance to be applied.

Max HP Increase

Increases max and current HP. If the Addition Type is *Linear*, the Additional HP is directly added to the max and current HP. If the Addition Type is *PercentageBased*, the Additional HP is calculated as a percentage of the max HP.

If there are multiple HP Increase buffs, some *Linear* and some *PercentageBased*, the total HP will be calculated using the following formula.

$$totalHP = (baseHP + linearBoost) \times (1 + percentBoost \div 100))$$

Field Name	**Type**	**Description**
Event Type	Event Type	Trigger of boost.
Addition Type	Addition Type	Method of increase: linear or percentage-based.
Additional HP	int	Increases HP based on the specified Addition Type.
Max Additional HP	int	If the current buff reaches this limit, it will remain at the limit until reset.
Effect Chance	int	Each time the triggering action occurs, the buff has this chance to be applied.

Invincibility

Makes the Entity invincible for a period of time, preventing it from taking any damage.

Field Name	**Type**	**Description**
Event Type	Event Type	Trigger of boost.
Invincibility Duration	int	Duration of invulnerability.
Effect Chance	int	Each time the triggering action occurs, the buff has this chance to be applied.

Mana Regeneration On Action

Simply restores Mana after an action occurs.

Field Name	**Type**	**Description**
Event Type	Event Type	Trigger of boost.
Recovered Mana	int	Amount of Mana that will be restored.
Effect Chance	int	Each time the triggering action occurs, the buff has this chance to be applied.

Mana Regeneration Persistent

Adds persistent Mana restoration to the Entity for its entire lifetime.

Field Name	**Type**	**Description**
Recovery Rate	float	The duration of intervals at which Mana will be restored.
Recovered Mana	int	Amount of Mana that will be restored.
Effect Chance	int	Each time the triggering action occurs, the buff has this chance to be applied.

Set Damage Cap

The damage applied to the Entity is compared to the current cap. If the cap is greater than or equal to the incoming damage (after applying defense), the damage has no effect on the Entity.

☑️ If the Addition Type is PercentageBased, the cap is calculated as a percentage of HP.

Field Name	Type	Description
Event Type	Event Type	Trigger of boost.
Addition Typy	Addition Type	Method of increase: linear or percentage-based.
Cap Value	int	The Entity cannot take damage below the cap.
Max Cap Value	int	If the current buff reaches this limit, it will remain at the limit until reset.
Effect Chance	int	Each time the triggering action occurs, the buff has this chance to be applied.

Shield

The shield acts as additional HP. Shield points are consumed before HP when taking damage.

Field Name	**Type**	**Description**
Event Type	Event Type	Trigger of boost.
Addition Type	Addition Type	Method of increase: linear or percentage-based.
Additional Shield Amount	int	Shield points apples.
Effect Chance	int	Each time the triggering action occurs, the buff has this chance to be applied.

Attacks

Attacks – nodes that represent active skills an entity can use in combat. Each attack can be triggered manually or automatically.

☑️ Each attack node ***must contain a unique name*** to distinguish it in the script.

Node Name	**Description**
Basic Attack	Deals damage with a cooldown
Mana Attack	Deals damage with a cooldown, but also costs mana
No-Damage Attack	Some effect with cooldown
No-Damage Mana Attack	Some effect with cooldown and mana cost

Basic Attack

Node that contains data for attack that deals damage and becomes available again after a cooldown.

Field Name	**Type**	**Description**
Attack Name	string	Unique name.
Attack Damage	int	Percent of basic damage.
Ignore Defense And Shields	bool	Is it true damage or not.
Delay After Attack	float	Cooldown duration.
Attack Duration	float	The amount of time that passes after the attack is triggered. Useful if you need to play an animation, delay damage application, or time visual effects.
Ready On Start	bool	Indicates whether the attack is ready to use immediately when the entity is instantiated.

Mana Attack

A node that contains data for an attack which deals damage and becomes available again after its cooldown ends and sufficient Mana is available.

Field Name	**Type**	**Description**
Attack Name	string	Unique name.
Attack Damage	int	Percent of basic damage.
Ignore Defense And Shields	bool	Is it true damage or not.
Required Mana	int	Mana consumed when the attack is performed.
Delay After Attack	float	Cooldown duration.
Attack Duration	float	The amount of time that passes after the attack is triggered. Useful if you need to play an animation, delay damage application, or time visual effects.
Ready On Start	bool	Indicates whether the attack is ready to use immediately when the entity is instantiated.

No-Damage Attack

No-Damage Attacks useful for attacks that don't directly deal damage, such as summoning or triggering other effects.

Field Name	**Type**	**Description**
Attack Name	string	Unique name.
Delay After Attack	float	Cooldown duration.
Attack Duration	float	The amount of time that passes after the attack is triggered. Useful if you need to play an animation, delay damage application, or time visual effects.
Ready On Start	bool	Indicates whether the attack is ready to use immediately when the entity is instantiated.

No-Damage Mana Attack

No-Damage Attacks useful for attacks that don't directly deal damage, such as summoning or triggering other effects.

Field Name	**Type**	**Description**
Attack Name	string	Unique name.
Required Mana	int	Mana consumed when the attack is performed.
Delay After Attack	float	Cooldown duration.
Attack Duration	float	The amount of time that passes after the attack is triggered. Useful if you need to play an animation, delay damage application, or time visual effects.
Ready On Start	bool	Indicates whether the attack is ready to use immediately when the entity is instantiated.

Event Types

☑️ Difference Between OnAttack and OnDamageDealt

Timing:

1. OnAttack is called when the attack starts — before the attack delay.
2. OnDealDamage is called when the attack is actually performed — after the attack delay.

Call Frequency:

1. OnAttack is called once per attack.
2. OnDealDamage is called once for each entity hit by the attack.

Action Type Name	**Description**
OnAttack	Called when an attack is starting.
OnCriticalHitLanded	Called when a crit hit is performed and lended on target.
OnDealDamage	Called when damage is dealt to some Entity.
OnDeath	Called when the current HP value drops to zero.
OnDodge	Called when a dodge is performed.
OnHeal	Called when the current HP value increased.
OnHitLanded	Called when a hit is performed and lands on a target, not necessarily dealing damage.
OnHpChange	Called when the current HP value changes.
OnInitialize	Called once when the Entity is created.
OnKill	Called when an enemy's HP value drops to zero.
OnManaChange	Called when the current Mana value changes.
OnManaRestored	Called when the current Mana value increased.
OnManaSpent	Called when the current Mana value decreased.
OnTakeDamage	Called when the current HP value decreased.

Addition Types

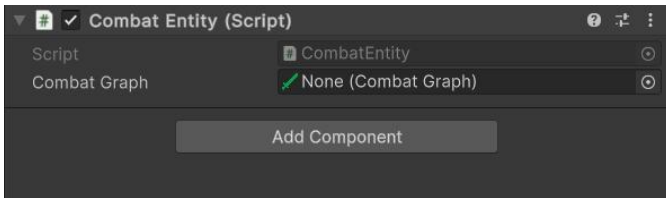
Modifies how the main value is increased:

- Linear – Adds the Additional Value directly to the base value.
- PercentageBased – Calculates the Additional Value as a percentage of the base value, then adds it to the base.

Scripting API

CombatEntity

A MonoBehaviour-based component that requires a CombatGraph to function. Using a Combat Entity allows you to define and execute various combat behaviors.



Combat Entity (Script)

Script	CombatEntity
Combat Graph	None (Combat Graph)

Public Methods

Method Name	Description
TakeDamage	Takes away hp.
ProhibitAttack	Prohibit attacks that should not be triggered by auto-attack.
AllowAttack	Allow previously prohibited attacks to be triggered by auto-attack.
StartAutoAttack	Starts auto attacks, time depends.
Attack	Performs attack, returns produced damage.
GetTimer	Returns timer, that contains time till selected attack reset.
AddNewBuff	Adds a new buff that wasn't part of the initial CombatGraph.
AddNewBuff	Dynamically adds new buffs that were not defined by default.
ReloadEntity	Sets default values of all stats and attacks.

Public Properties (readonly)

Property Name	Type	Description
CurrentHp	int	Return current hp.
MaxHp	int	Returns max hp after applying all boosters.
AttackDamage	int	Returns basic damage after applying all boosters.
CurrentMana	int	Return current mana.
MaxMana	int	Returns max mana after applying all boosters.
Defense	int	Returns defence after applying all boosters.
CurrentShield	int	Returns current shield value.
EventManager	EventManager	Field that gives access to inner combat events.

TakeDamage

method

Definition

```
DamageData TakeDamage(AttackData attackData, CombatEntity dealer)
```

Parameters

`attackData` AttackData - Description of an attack that can be obtained from StartAutoAttack, Attack, or constructed manually.

`dealer` CombatEntity - Entity that deals damage.

Returns

DamageData - Some data that describes the result of a performed attack, such as damage taken and status.

Description

The example below shows a variant of the implementation where the attack is triggered by the player. If the attack is available, the `AttackData` is passed to the `TakeDamage` function.

```
public class PlayerAttacks : MonoBehaviour
{
    [SerializeField] CombatEntity combatEntity;

    private void Update()
    {
        if (Input.GetKeyDown(KeyCode.X))
        {
            BasicAttack(combatEntity.Attack("Basic Attack"));
        }
    }

    [SerializeField] Weapon sword;
    void BasicAttack(AttackData data)
    {
        if (!data.IsAvalible)
        {
            Debug.Log("Attack is unavailable!");
            return;
        }

        Instantiate(sword, transform.position, transform.rotation, 
transform)
            .SubscribeOnHitAction((entity) => { 
entity.TakeDamage(data, combatEntity); });
    }
}
```

ProhibitAttack

method

Definition

```
public void ProhibitAttack(string attackName)
```

Parameters

`attackName` string - Unique name of an attack you want to prohibit from being called automatically.

Description

In the `Update` function below, the melee basic attack is prohibited when the player is too far from the enemy and allowed otherwise.

```
private void Update()
{
    if (Vector2.Distance(player.position, transform.position) > 6)
    {
        combatEntity.ProhibitAttack("Basic Attack");
    }
    else
    {
        combatEntity.AllowAttack("Basic Attack");
    }
}
```

AllowAttack

method

☑️ All attacks are allowed by default.

Definition

```
public void AllowAttack(string attackName)
```

Parameters

`attackName` string - Unique name of an attack you want to allow to be called automatically.

Description

In the `Update` function below, the melee basic attack is prohibited when the player is too far from the enemy and allowed otherwise.

```
private void Update()
{
    if (Vector2.Distance(player.position, transform.position) > 6)
    {
        combatEntity.ProhibitAttack("Basic Attack");
    }
    else
    {
        combatEntity.AllowAttack("Basic Attack");
    }
}
```

StartAutoAttack

method

Definition

```
void StartAutoAttack(AttackHandler action)
```

Parameters

`action` AttackHandler - Delegate that defines a method handling specific attacks for the current entity.

Description

The `AttacksHandler` method handles different attacks based on their names. It is passed as a parameter to the `StartAutoAttack` function at the `Start` method.

```
[SerializeField] CombatEntity combatEntity;

private void Start()
{
    combatEntity.StartAutoAttack(AttacksHandler);
}

void AttacksHandler(AttackData attackData)
{
    switch (attackData.AttackName)
    {
        case "Basic Attack":
            BasicAttack(attackData);
            break;

        case "Mana Attack":
            ManaAttack(attackData);
            break;
    }
}
```

AttackHandler

delegate

Definition

```
void AttackHandler(AttackData attackData)
```

Parameters

`attackData` AttackData - All data required to handle a specific attack, such as damage produced, attack name, status, etc.

Description

Delegate used to handle auto-attack events.

```
void AttacksHandler(AttackData attackData)
{
    switch (attackData.AttackName)
    {
        case "Basic Attack":
            BasicAttack(attackData);
            break;

        case "Mana Attack":
            ManaAttack(attackData);
            break;
    }
}
```

Attack

method
If you call this function and it returns zero damage, check the `isAvailable` field.

☑️ If you call this function and it returns zero damage, check the `isAvailable` field.
It will be `false` if the attack is on cooldown or if there isn't enough mana.

Definition

```
AttackData Attack(string attackName)
```

Parameters

`attackName` string - Unique attack name.

Returns

AttackData - Includes producedDamage, status etc.

Description

The example below shows a variant of the implementation where the attack is triggered by the player with the `Attack` function.

```
public class PlayerAttacks : MonoBehaviour
{
    [SerializeField] CombatEntity combatEntity;

    private void Update()
    {
        if (Input.GetKeyDown(KeyCode.X))
        {
            BasicAttack(combatEntity.Attack("Basic Attack"));
        }
    }

    void BasicAttack(AttackData data)
    {
        if (!data.IsAvalible)
        {
            Debug.Log("Attack is unavailable!");
            return;
        }

        Debug.Log("Attack was performed!");
    }
}
```

GetTimer

method

Definition

```
Timer GetTimer(string attackName)
```

Parameters

`attackName` string - The unique name of the attack whose timer should be retrieved.

Returns

Timer - Useful if you want to display in the UI how much time remains until the skill becomes available.

Description

In the example usage within the `Update` method, the slider is updated every frame using the `TimeTillReset` timer property.

```
class SkillSlider : MonoBehaviour
{
    [SerializeField] CombatEntity combatEntity;
    [SerializeField] TMPro.TMP_Text attackTimer;
    [SerializeField] UnityEngine.UI.Slider attackSlider;

    private void Update()
    {
        attackTimer.text =
combatEntity.GetTimer("Attack").TimeTillReset.ToString();

        attackSlider.value =
combatEntity.GetTimer("Attack").TimeTillReset;
        attackSlider.maxValue =
combatEntity.GetTimer("Attack").CooldownTime;
    }
}
```

Timer

class

Useful if you want to display in the UI how much time remains until the skill becomes available.

Name	Type	Description
TimeTillReset	int	Time until the skill becomes available, rounded up.
CooldownTime	int	Skill cooldown duration after applying all current time boosters.

EventManager

class

Class that gives access to inner combat events.

Public Methods

Method Name	**Description**
AddListener	Adds a method that will be called after the chosen event is performed.
RemoveListener	Removes a method that was previously added.

AddListener

method

Definition

There are two versions of this method: one for events that return data and one for those that do not. Each event type has a strictly defined return type, which you can find documented on the EventType page.

Overloads

AddListener(EventType type, UnityAction listener)
AddListener<T>(EventType type, UnityAction<T> listener)

AddListener(EventType type, UnityAction listener)

```
void AddListener(EventType type, UnityAction listener)
```

Parameters

`type` EventType - A special enum that represents the different types of events.

`listener` UnityAction - A delegate that allows you to pass your functions as arguments to other functions.

Description

Example of using this function to update the HP slider.

```
class HPSlider : MonoBehaviour
{
    [SerializeField] CombatEntity combatEntity;
    [SerializeField] Slider hpSlider;
    [SerializeField] TMP_Text hpText;

    private void Start()
    {
```

```
combatEntity.EventManager.AddListener(EventManager.EventType.OnHpC
hange, UpdateField);
          UpdateField();
      }

      private void UpdateField()
      {
          //also update MaxHp
          hpSlider.maxValue = combatEntity.MaxHp;
          hpSlider.value = combatEntity.CurrentHp;

          hpText.text =
$"{combatEntity.CurrentHp}/{combatEntity.MaxHp}";
      }
 }
```

AddListener<T>(EventType type, UnityAction<T> listener)

```
void AddListener<T>(EventType type, UnityAction<T> listener)
```

Parameters

`type` EventType - A special enum that represents the different types of events.

`listener` UnityAction<T> - A delegate that allows you to pass your functions as arguments to other functions. In `<>` you should specify the return type of the event.

Description

Example of using this function to send a message that includes the event's return data.

```
public class MessagesController : MonoBehaviour
{
    [SerializeField] CombatEntity combatEntity;
    [SerializeField] Message message;
    [SerializeField, ColorUsage(true, true)] Color color =
Color.white;

    private void Start()
    {

combatEntity.Actions.AddListener<DamageData>(Actions.EventType.OnT
akeDamage, (data) =>
        {
            Message newMessage;
            //check if damage was dodged
                if (data.Status == BlockStatus.Dodged)
                {
                    newMessage = Instantiate(message,
transform.position, Quaternion.identity);
                    newMessage.UpdateText("Dodged!", color);
                    return;
                }
                //if damage was not dodged, display it
                newMessage = Instantiate(message,
transform.position, Quaternion.identity);
                newMessage.UpdateText(data.DamageTaken.ToString(),
color);

        });
    }
}
```

RemoveListener

method

Definition

This function is used to prevent destroyed or deleted objects from triggering their callbacks. Call this function in the `OnDestroy` method of your GameObject to ensure proper cleanup.

RemoveListener(EventType type, UnityAction listener)
RemoveListener<T>(EventType type, UnityAction<T> listener)

RemoveListener(EventType type, UnityAction listener)

```
void RemoveListener(EventType type, UnityAction listener)
```

Parameters

`type` EventType - A special enum that represents the different types of events.

`listener` UnityAction - A delegate that allows you to pass your functions as arguments to other functions.

Description

```
private void OnDestroy()
{
    combatEntity?
        .EventManager
        .RemoveListener(EventManager.EventType.OnHpChange,
UpdateField);
}
```

RemoveListener<T>(EventType type, UnityAction<T> listener)

```
void RemoveListener<T>(EventType type, UnityAction<T> listener)
```

Parameters

`type` EventType - A special enum that represents the different types of events.

`listener` UnityAction - A delegate that allows you to pass your functions as arguments to other functions. In `<>` you should specify the return type of the event.

Description

```
private void OnDestroy()
{
    combatEntity?
        .EventManager

.RemoveListener<DamageData>(EventManager.EventType.OnTakeDamage,
MessageSend);
}
```

EventType

enum

☑️ These events are the same Event Types that can be selected in the graph editor.

This enumerator is used to identify callbacks for specific events. Some events may return data related to the action that triggered them. The type of that data is described by the Return Type field.

Action Name	**Return Type**	**Description**
OnAttack	AttackData	Called when an attack is starting.
OnCriticalHit	void	Called when a crit hit is performed.
OnDealDamage	void	Called when damage is dealt to some Entity.
OnDeath	void	Called when the current HP value drops to zero.
OnDodge	void	Called when a dodge is performed.
OnHeal	int	Called when the current HP value increased.
OnHpChange	int	Called when the current HP value changes.
OnHitLanded	DamageData	Called when a hit is performed and lands on a target, not necessarily dealing damage.
OnInitialize	void	Called once when the Entity is created.
OnKill	void	Called when an enemy's HP value drops to zero.
OnManaChange	int	Called when the current Mana value changes.
OnManaRestored	int	Called when the current Mana value increased.
OnManaSpent	int	Called when the current Mana value decreased.
OnTakeDamage	DamageData	Called when the current HP value decreased.

AddNewBuff

method

Definition

```
void AddNewBuff(BattleEffect effectsBattle)
```

Parameters

`effectBattle` BattleEffect - The base type from which all buffs inherit.

Description

The example below describes a function that adds different buffs to an entity. For instance, it can be used as a button-triggered action.

```
[SerializeField] CombatEntity playerEntity;
public void SelectBlessing(int id)
{
    if (id == 1)
    {
        playerEntity.AddNewBuff(new DodgeRateIncrease(
            EventManager.EventType.OnInitialize,
            40,
            40));
        Time.timeScale = 1;
        blessingSelection.SetActive(false);
        return;
    }

    if (id == 2)
    {
        playerEntity.AddNewBuff(new DamageIncrease(
            EventManager.EventType.OnManaSpent,
            AdditionType.Linear,
            5,
            20));

        Time.timeScale = 1;
        blessingSelection.SetActive(false);
```

```
            return;
        }
}
```

ReloadEntity

method

Definition

```
void ReloadEntity()
```

Description

Resets all parameters to their default values, including all buffs and attacks.

DamageData

struct

Public Fields (readonly)

FieldName	Type	Description
DamageTaken	int	Represents the final damage value after all modifiers (defense, dodge, cap, etc.) have been applied.
Status	DamageData	Represents the taken damage status for clearer display of events like dodges.

DamageStatus

enum

Status Name	**Description**
Dodged	The entity took no damage due to a dodge.
Landed	The entity took normal damage.
Caped	The entity took no damage due to a cap.
Blocked	The entity took no damage due to an invincibility.
Critical	The entity took critical damage.

AttackData

struct

Public Fields (readonly)

Field Name	Type	Description
IsAvalible	bool	Will be false if the attack is on cooldown or if there isn't enough mana.
DamageProduced	int	The final damage value after applying critical hit calculations.
Status	AttackStatus	Represents the attack status for clearer display of events like critical hits.

AttackStatus

enum

Status Name	Description
Normal	Simple attack.
Crit	Attack dealt extra damage.
NoDamage	A no-damage attack was performed.

BattleEffect

class

An abstract class that serves as the base for all buffs used in the **Combat Graph**.

Buffs

Buff Name	Related CombatGraph Node
AttackSpeedIncrease	Attack Speed Increase
CritDamageIncrease	Crit Damage Increase
CritRateIncrease	Crit Rate Increase
DamageCap	Damage Cap
DamageIncrease	Damage Increase
DodgeRateIncrease	Dodge Rate Increase
HealingOnAction	Healing On Action
HealingPersistent	Healing Persistent
Invincibility	Invincibility
ManaRegenerationOnAction	Mana Regeneration On Action
ManaRegenerationPersistent	Mana Regeneration Persistent
ManHPIncrease	Man HP Increase
Shield	Shield

AttackSpeedIncrease

class

Public Methods

Method Name	Description
Constructor	Creates an instance of AttackSpeedIncrease.

Constructor AttackSpeedIncrease

constructor

Definition

```
public AttackSpeedIncrease(
    EventManager.EventType eventType,
    int additionalSpeed,
    int cap,
    int effectChance = 100)
```

Parameters

`eventType` EventType - Event type that triggers a buff action.

`additionalSpeed` int - Additional value that is added to `attackSpeed` when the buff is triggered.

`cap` int - Once the sum of additional values exceeds the `cap`, buffs will no longer be triggered.

`effectChance` int - Percentage of effect chance.

CritDamageIncrease

class

Public Methods

Method Name	Description
Constructor	Creates an instance of CritDamageIncrease.

Constructor CritDamageIncrease

constructor

Definition

```
public CritDamageIncrease(
    EventManager.EventType eventType,
    int additionalCritDamage,
    int cap,
    int effectChance = 100)
```

Parameters

`eventType` EventType - Event type that triggers a buff action.

`additionalCritDamage` int - Additional value that is added to `critDamage` when the buff is triggered.

`cap` int - Once the sum of additional values exceeds the `cap`, buffs will no longer be triggered.

`effectChance` int - Percentage of effect chance.

CritRateIncrease

class

Public Methods

Method Name	Description
Constructor	Creates an instance of CritRateIncrease.

Constructor CritRateIncrease

constructor

Definition

```
public CritRateIncrease(
    EventManager.EventType eventType,
    int additionalCritRate,
    int cap,
    int effectChance = 100)
```

Parameters

`eventType` EventType - Event type that triggers a buff action.

`additionalCritRate` int - Additional value that is added to `critRate` when the buff is triggered.

`cap` int - Once the sum of additional values exceeds the `cap`, buffs will no longer be triggered.

`effectChance` int - Percentage of effect chance.

DamageCap

class

Public Methods

Method Name	Description
Constructor	Creates an instance of DamageCap.

Constructor DamageCap

constructor

Definition

```
public DamageCap(
    EventManager.EventType eventType,
    AdditionType additionType,
    int capValue,
    int maxCapValue,
    int effectChance = 100)
```

Parameters

`eventType` EventType - Event type that triggers a buff action.

`additionType` AdditionType - Way of additional value calculation.

`capValue` int - A value that represents the threshold below which an entity cannot take damage.

`cap` int - Once the sum of cap values exceeds the `cap`, buffs will no longer be triggered.

`effectChance` int - Percentage of effect chance.

DamageIncrease

class

Public Methods

Method Name	Description
Constructor	Creates an instance of DamageIncrease.

Constructor DamageIncrease

constructor

Definition

```
public DamageIncrease(
    EventManager.EventType eventType,
    AdditionType additionType,
    int additionalDamage,
    int cap,
    int effectChance = 100)
```

Parameters

`eventType` EventType - Event type that triggers a buff action.

`additionType` AdditionType - Way of additional value calculation.

`additionalDamage` int - Additional value that is added to `damage` when the buff is triggered.

`cap` int - Once the sum of additional values exceeds the `cap`, buffs will no longer be triggered.

`effectChance` int - Percentage of effect chance.

Defense Increase

class

Public Methods

Field Method	Description
Constructor	Creates an instance of DefenceIncrease.

Constructor DefenseIncrease

constructor

Definition

```
public DefenseIncrease(
    EventManager.EventType eventType,
    AdditionType additionType,
    int additionalDefense,
    int cap,
    int effectChance = 100)
```

Parameters

`eventType` EventType - Event type that triggers a buff action.

`additionType` AdditionType - Way of additional value calculation.

`additionalDefense` int - Additional value that is added to `defense` when the buff is triggered.

`cap` int - Once the sum of additional values exceeds the `cap`, buffs will no longer be triggered.

`effectChance` int - Percentage of effect chance.

DodgeRateIncrease

class

Public Methods

Method Name	Description
Constructor	Creates an instance of DodgeRateIncrease.

Constructor DodgeRateIncrease

constructor

Definition

```
public DodgeRateIncrease(
    EventManager.EventType eventType,
    int additionalDodgeRate,
    int cap,
    int effectChance = 100)
```

Parameters

`eventType` EventType - Event type that triggers a buff action.

`additioalDodgeRate` int - Additional value that is added to `dodge` when the buff is triggered.

`cap` int - Once the sum of additional values exceeds the `cap`, buffs will no longer be triggered.

`effectChance` int - Percentage of effect chance.

HealingOnAction

class

Public Methods

Method Name	Description
Constructor	Creates an instance of HealingOnAction.

Constructor HealingOnAction

constructor

Definition

```
public HealingOnAction(
    EventManager.EventType eventType,
    int healedHpAmount,
    int effectChance = 100)
```

Parameters

`eventType` EventType - Event type that triggers a buff action.

`healedHpAmount` int - Amount of HP that will be restored.

`effectChance` int - Percentage of effect chance.

HealingPersistent

class

Public Method

Method Name	Description
Construtor	Creates an instance of HealingPersistent.

Constructor HealingPersistent

constructor

Definition

```
public HealingPersistent(
    float healingSpeed,
    int healedHpAmount,
    int effectChance = 100)
```

Parameters

`healingSpeed` float - The duration of intervals at which HP will be restored.

`healedHpAmount` int - Amount of HP that will be restored.

`effectChance` int - Percentage of effect chance.

Invincibility

class

Public Method

Method Name	Description
Constructor	Creates an instance of Invincibility.

Constructor Invincibility

constructor

Definition

```
public Invincibility(
    EventManager.EventType actionName,
    float invincibilityTime,
    int effectChance = 100)
```

Parameters

`eventType` EventType - Event type that triggers a buff action.

`invincibilityTime` int - A time window during which the entity is immune to damage.

`effectChance` int - Percentage of effect chance.

ManaRegenerationOnAction

class

Public Methods

Method Name	Description
Constructor	Creates an instance of ManaRegenerationOnAction.

Constructor ManaRegenerationOnAction

constructor

Definition

```
public ManaRegenerationOnAction(
    EventManager.EventType eventType,
    int recoveredMana,
    int effectChance = 100)
```

Parameters

`eventType` EventType - Event type that triggers a buff action.

`recoverdMana` int - Amount of Mana that will be restored.

`effectChance` int - Percentage of effect chance.

ManaRegenerationPersistent

class

Public Methods

Method Name	Description
Constructor	Creates an instance of ManaRegenerationPesistant.

Constructor ManaRegenerationPersistent

constructor

Definition

```
public ManaRegenerationPersistent(
    float manaRecoverySpeed,
    int recoveredMana,
    int effectChance = 100)
```

Parameters

`manaRecoverySpeed` float - The duration of intervals at which Mana will be restored.

`recoveredMana` int - Amount of Mana that will be restored.

`effectChance` int - Percentage of effect chance.

MaxHPIncrease

class

Public Methods

Method Name	Description
Constructor	Creates an instance of MaxHPIncrease.

Constructor MaxHPIncrease

constructor

Definition

```
public MaxHPIncrease(
    EventManager.EventType eventType,
    AdditionType additionType,
    int additionalHp,
    int cap,
    int effectChance = 100)
```

Parameters

`eventType` EventType - Event type that triggers a buff action.

`additionType` AdditionType - Way of additional value calculation.

`additionalHp` int - Additional value that is added to `HP` when the buff is triggered.

`cap` int - Once the sum of additional values exceeds the `cap`, buffs will no longer be triggered.

`effectChance` int - Percentage of effect chance.

Shield

class

Public Methods

Method Name	Description
Constructor	Creates an instance of Shield.

Constructor Shield

constructor

Definition

```
public Shield(
    EventManager.EventType eventType,
    AdditionType additionType,
    int shieldAmount,
    int effectChance = 100)
```

Parameters

`eventType` EventType - Event type that triggers a buff action.

`additionType` AdditionType - Way of additional value calculation.

`shieldAmount` int - Additional value that is added to `Shield` when the buff is triggered.

`effectChance` int - Percentage of effect chance.

AdditionType

Modifies how the main value is increased:

> **Linear** – Adds the Additional Value directly to the base value.
>
> **PercentageBased** – Calculates the Additional Value as a percentage of the base value, then adds it to the base.