



**Moba Storm Multiplayer Arena Framework  
Documentation**  
©JmgDigital All rights reserved.

# Table of Contents

## [Introduction](#)

[About this framework](#)

[Technical specs](#)

## [Getting started](#)

[Requirements](#)

[Installation](#)

## [Building and playing the game](#)

## [The Game](#)

## [User Interface](#)

[Top Panel](#)

[Bottom left panel](#)

[Bottom mid panel](#)

## [Multiplayer - Sync Model](#)

[Fixed Step](#)

## [Network Instantiation / Pool System](#)

## [Spawning Entities](#)

## [Pathfinding / Cell Positioning](#)

## [Entity Editor](#)

[Model Setup](#)

[Animation Setup](#)

[Create Entity](#)

[Entity Prefab Builder](#)

[Entity Data Fields](#)

[Animations](#)

[Saving / Loading Entity Data](#)

## [Ability System](#)

[Create Ability](#)

[Remove Ability](#)

[Ability Configuration Parameters](#)

[Adding Server Prefab](#)

[Adding Client Prefab](#)

[Ability Level Data](#)

[Ability Description Parameters:](#)

## [Side Effects Configuration](#)

## [Game Data Manager](#)

## [Sprite Database Manager](#)

## [Damage Process](#)

## [Audio Manager](#)

## [Moba Entity Component](#)

## [Logic Component](#)

## [Entity Behaviour Component](#)

## [Entity Abilities](#)

## [Entity Animator](#)

## [Entity Canvas](#)

## [Task Manager](#)

# Introduction

**Thanks for your purchase of this framework!**

**We hope you like this kit, we appreciate any feedback or comments.**

We are working on a more complete manual, if you have any questions or comments email us at

[support@jmgdigital.com](mailto:support@jmgdigital.com) .

If you find a bug please send detailed information to

[msbugs@jmgdigital.com](mailto:msbugs@jmgdigital.com) .

## About this framework

This is a **moba style multiplayer framework**, with a lot of fundamental gameplay taken from the most commercial moba games.

You can choose a character to fight against the enemy team on three different lanes, where neutral monsters spawn and towers defend the "big mother tower". Every time you kill a minion or a player you gain points, which allow you to level up your character with more health and more damage. The game ends when a team destroys the big nexus!!!!

## Technical specs

- Authoritative Server Model with a dedicated server instance.
- Deterministic network sync model, the clients send inputs, and the actions are executed on all clients at the same Fixed Step resulting in the same simulation on the server and all clients.
- Complete UI Menu Manager with a modern GUI Design.
- Options to create a dedicated server, join a game server and join with a simulated latency.
- Complete custom pathfinding.
- Enemy AI cell positioning used by minions or enemies to distribute positions and lock cells to avoid overlapping positions.
- 2 Complete custom characters with animations and abilities.
- Ability Editor Window that allows to create complete abilities, and customize all parameters, add side effects and draw all properties from it.

- Character Editor Window that allows to create and configure characters, add animations and see a preview of the character in real time.
- You can create new characters, new enemies without writing a line of code, even you can fully customize your spells and attacks creating your custom scripts using the visual editors.
- Standard MOBA keys QWER
- Melee and ranged abilities
- Impact and AoE Damage
- Char selection window with portraits and sounds for each character
- Standard entity logics for "Player Characters", "Towers", "Minions" and "Nexus" with the flexibility to create your own custom Logic AI Implementations.
- Configurable Canvas Health Bar for each "MobaEntity".
- Multiple Targeting choices to cast abilities on "Target", "Position", "Self Casting", "All Targets", "Random Targets".
- Abilities can put side effects like slow enemies, stun, deal dot "DamageOverTime" or hot "HealOverTime", also you can write your custom scripts to create any kind of ability.

# Getting started

## Requirements

Be sure that you are running at least these software versions before using the starter kit, or it won't function as intended:

Unity3d [Free](#) or [Pro](#) version **5.0** or later

## Installation

When running this system for the first time it is recommended to do so in a new, clean Unity project.

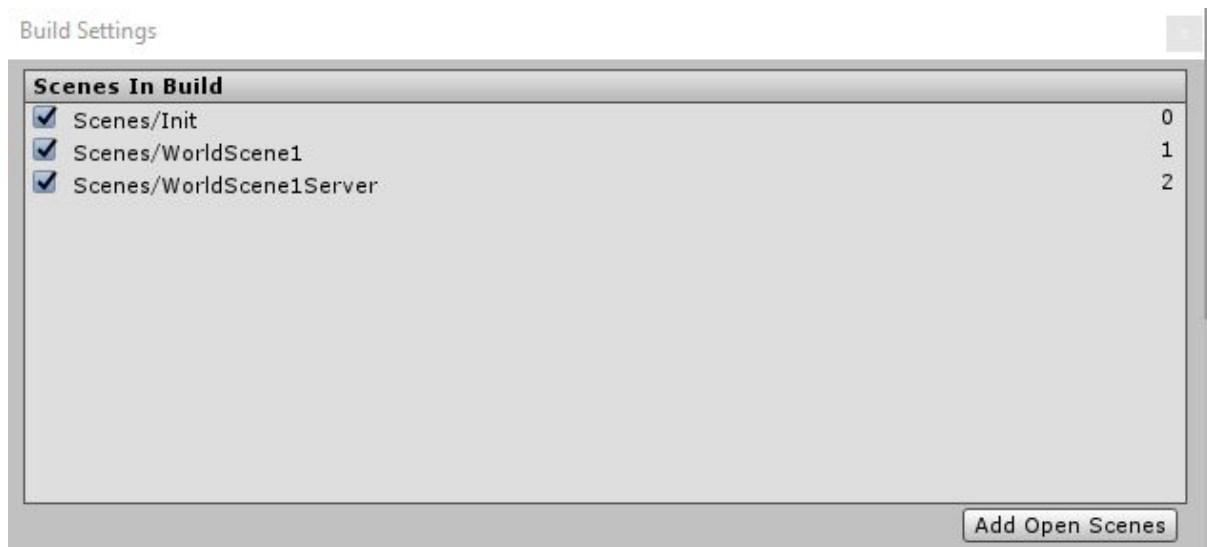
Import MobaStormAsset

Add all the included scenes to your editor **File -> Build Settings**. . For now, make sure to drag the scene "**Init**" to the top of the level list (index 0).

## Building and playing the game

This framework comes with default local IP of 127.0.0.1, so you can easily try the game without any problems or additional configuration.

Add all the included scenes to your editor **File -> Build Settings**. . For now, make sure to drag the scene "**Init**" to the top of the level list (index 0), add the scene "WorldScene1" and "WorldScene1Server". the result will look like this.



From the Unity main menu, Click "File -> Build Settings".

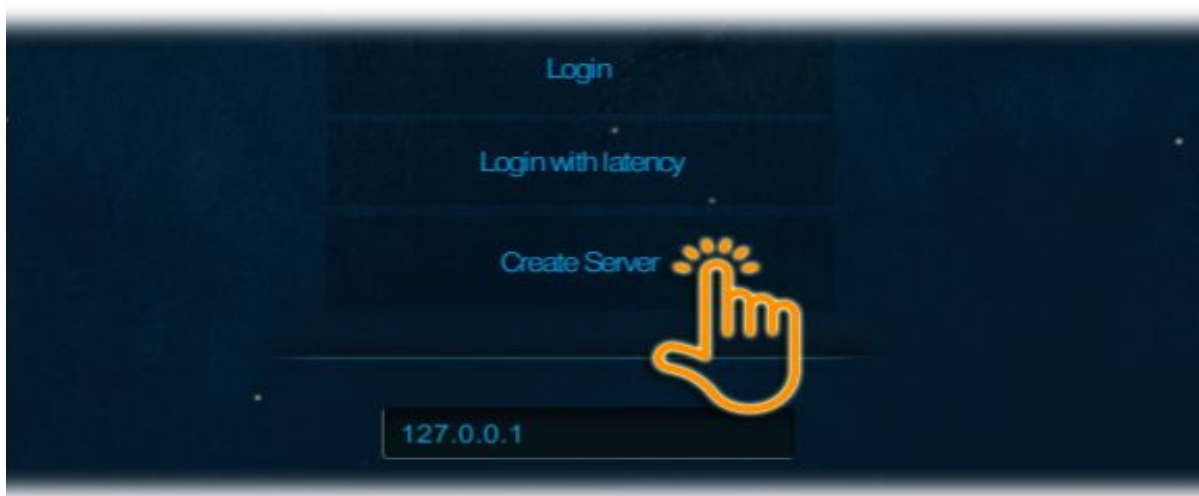
In the "Platform" box, select "PC, Mac & Linux Standalone"

Click the "Build" button.

Run two instances of the stand-alone build: When you run the instance, by default it will open the “Init Scene” and it will look like this.



- In the first instance menu, select click "Create Server"



A character selection screen must be shown indicating that the “**Server is running**” by default the server is ***Initialized on port 2000***, now you successfully create a server.



- In the second instance menu, click "Login" and you will join the game.

Tip: By default the **ip is set to 127.0.0.1** which is the local Ip.

**TIP:** this is a fast guide to get you into the game running the server and client on the same computer, if you want to set up the server and client on different machines you will need to change the IP address in the client instances to the server machine.





Now you are in the character selection window, the server assigns the joined player to any free slot available, now you can select a character by clicking on the portrait window (MID BOX). and the character icon will show on your selected slot.



You can also change your position to another slot, or change to another team by clicking any **FREE SLOT BOX**, each box contains info about the **Player Name, Picked Character and Picking Status**.



Now when you are ready just click **Ready !!** And you are ready to kill some waves and towers.





# The Game

We will cover some of the inGame features and some key bindings available on this asset: Each player, boss, tower, minion or any character will show some stats on the top of the character.

- Players will show their names, lvl, mana and health.
- Minions and bosses will show its lvl, hp and dmg.
- Entities increase stats when they level up like Armor and Health Points.
- The damage calculations are made using custom advanced combat formulas which take in consideration the target resistances, buff or debuff, even side effects can modify the damage calculation, you can create effects like "Immune to All damage", "Double Damage for 3 seconds"
- The system supports Active Abilities or Passive ones.

The game keys work as any moba standard. here we will explain some of the control keys.

- **Q W E R** use character spells.
- **B** teleport to base.
- **Right Click** (ground) move character or attack.
- **Space** Lock and Unlock camera.

# User Interface

## Top Panel

It shows the team scores and the network time.

## Bottom left panel

It shows the Attack Damage, Ability Power, Gold, Armor, Magic Resistance, and Player Experience Bar.

## Bottom mid panel

It shows the player's Health Points, and Mana, it also shows the abilities with their corresponding icons and tooltip, it also shows the Cool Down time.



# Multiplayer - Sync Model

The core of this multiplayer connection system is allocated on 2 components MobaServerManager and MobaServerClient. They handle all connection logic between server and clients, all the system is built over Unet High Level Api.

You can create a game server just by calling **SetupServer()**; or call **SetupClient()**; to connect to a game server.

The framework runs all game logic in a deterministic way by running on a Fixed Step, the clients only send Player Inputs to save bandwidth, and get the results as “Action Task” to be executed in the same step to replicate the server simulation on all clients.

## *Fixed Step*

The **NetworkTime** is responsible of keeping the server and clients in sync, by sending the **FixedStep** and the **NetworkTimeStamp**. The **NetworkTime** is constantly making adjustments to keep the minimal **FixedStep** delay based on the connection latency.

When the Server Entity executes a Task, it sends an RPC to the clients with the task parameters and the current Server Fixed Step, all Client Entities put the same Task in a queue that will run in the same Fixed Step for all clients.

# Network Instantiation / Pool System

The framework contains custom network Spawn Handler methods, which allows to create different prefabs for the Server and Clients, that way we minimize the cost to host game servers by creating server prefabs without models, visuals, or animations.

The network instantiation works side by side with our custom Network Pooling System that ensures that all GameObjects are destroyed and instantiated multiple times.

In the next image we can see how we add two different prefabs for the client and server and a quantity field if you want to PreLoad any GameObjects.

## *SpawnManager.cs*

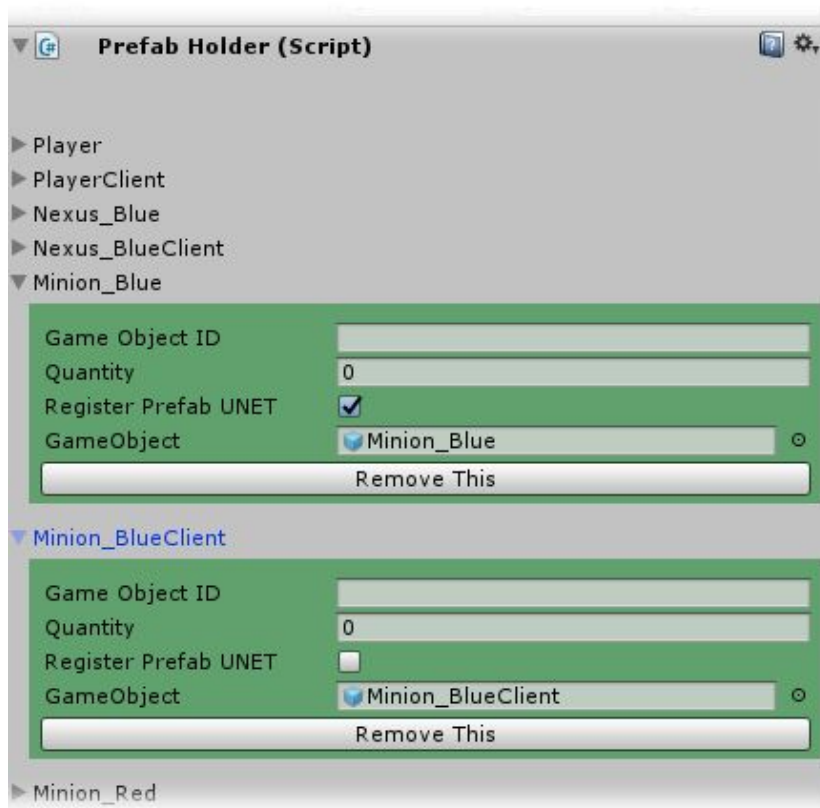
Contains all methods to spawn gameobjects and unspawn GameObjects from a pool.

*public GameObject InstantiatePool(string obj, Vector3 pos, Quaternion rot)*

Gets a GameObject from his PrefabName spawn it.

*public void DestroyPool(GameObject spawned)*

Unspawn a GameObject, and put back on the pool.



Tip: you only need to add client and server prefabs when you need to spawn a networking gameobject.

# Spawning Entities

To instantiate a moba entity you need to call on the **ServerEntitySpawner** one of the methods available. The scene **WorldScene1Server** contains the ServerEntitySpawner prefab that has the full list of static world entities to spawn and a transform to get the position and rotation.

*SpawnCharacter(MobaPlayer player, Vector3 pos, Quaternion rot, NetworkConnection connectionToClient)*

Creates a Character based on the player selection

*SpawnNexus(string nexusData, Vector3 pos, Quaternion rot)*

Creates a nexus based on the data parameter, position and rotation.

*SpawnTower(string towerDefintion, Vector3 pos, Quaternion rot)*

Creates a Tower based on the data parameter, position and rotation.

*SpawnMinion(string data, Vector3 pos, Quaternion rot)*

Creates a Minion based on the data parameter, position and rotation.

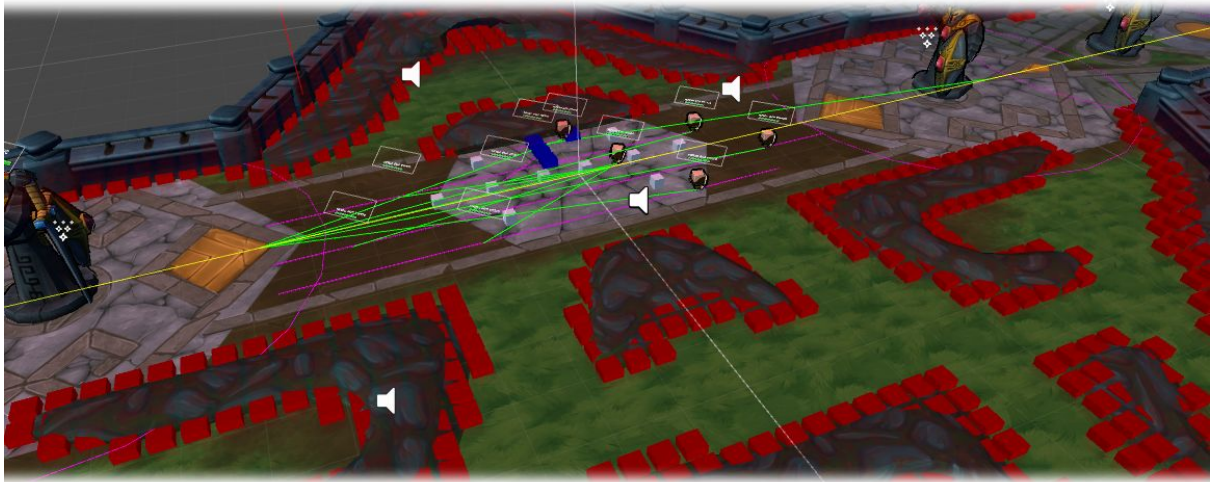
When you create a new entity the server will call NetworkServer.Spawn(Obj) to spawn the client prefab on all clients.

When the server is initialized, it will call SpawnEntityObjects() to spawn all Initial World Entities "Towers", "Nexus".

# Pathfinding / Cell Positioning

The framework uses their custom Pathfinding system using A\* and grid cells, it also contains a smooth path system using **Brensnham algorithm**.

Cells are used for pathfinding and enemy AI to find the best spot in attack range.



## *Pathfinding.cs*

Contains all methods available for pathfinding.

### *FindPath(Vector2 startPos, Vector2 targetPos)*

Create a new path from a vector2 world position to a vector2 world position.

### *SmoothPath(List<Node> nodeList)*

Smooth a path using the Brensnham algorithm.

### *GetNearestWalkableNode(Vector2 fromPos, Vector2 targetPos, float range)*

Get a nearest walkable node to attack based on the ability range, this method will search neighbours to find the better Node to position and attack. Used by minions.

## *Grid.cs*

Class used to store all the Grid Nodes and contains all methods to access them.

### *CreateGrid()*

Creates a grid using a predefined world size and spacing.

### *Node NodeFromWorldPoint(Vector2 worldPosition)*

Returns a node from a vector2 world position.

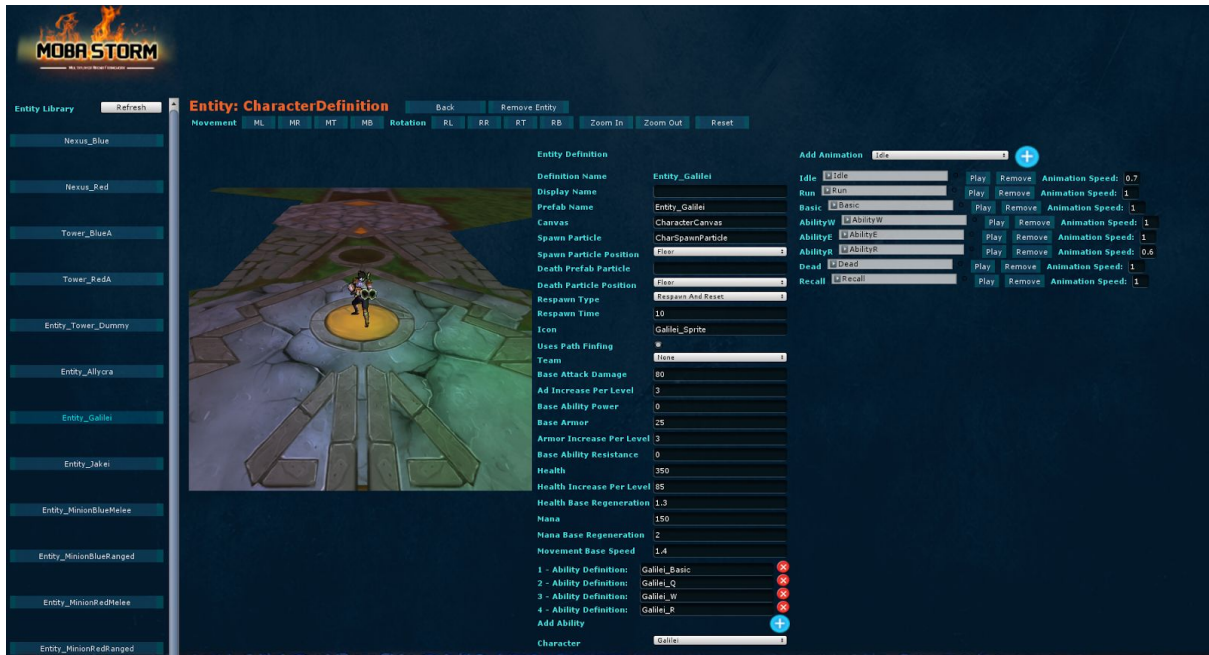
### *List<Node> GetNeighbours(Node node)*

Return a list of node neighbors.



# Entity Editor

The Entity Editor is used to create and put Entities into the game in a easy way. In the Entity Editor you can add and preview an entity model, add animations, set animation speed, customize entity parameters, and datas.



## Model Setup

To create a game entity is necessary a 3d model (FBX).

Import the FBX model into the folder Assets/Models/"YourNewEntityName".

Go to the project tab and Select the FBX model.

In the inspector, click on the RIG tab, and choose legacy on the Animation Type.



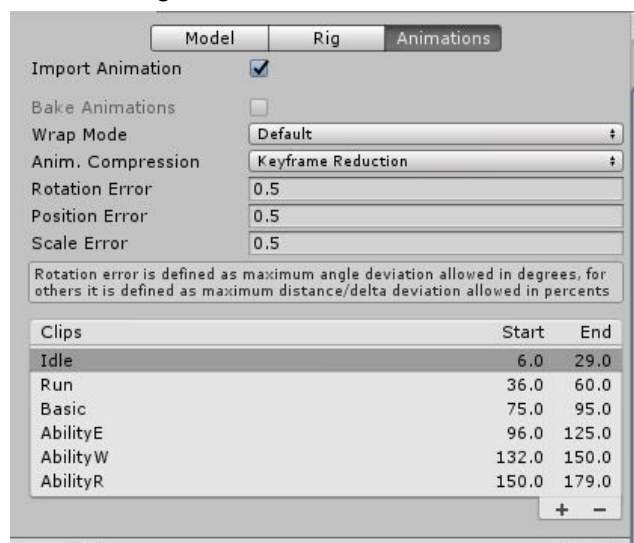
## Animation Setup

For the system to recognize animations for entities, they will need a special nomenclature defined by a enum. You can find in **EntityAnimator.cs** all entities handle their basic animations like idle, run, basic, dead, stun... you can also use the 2 predefined custom animations, or add more if you want.

```
public enum EEntityState
{
    Idle = 1, {loop}
    Run = 2, {loop}
    Basic = 3,
    AbilityQ = 4,
    AbilityW = 5,
    AbilityE = 6,
    AbilityR = 7,
    Dead = 8,
    Recall = 9,
    Emote = 10,
    CastingQ = 11, {loop}
    CastingW = 12, {loop}
    CastingE = 13, {loop}
    CastingR = 14, {loop}
    Stun = 15,
    Custom1 = 16,
    Custom2 = 17,
}
```

This is an example of Galilei animation configuration. If you previously set the correct names for animations in the FBX model, the **Entity Editor** will recognize them automatically when you add the new Entity.

You can also add them later using the editor.



Tip: for animations to work properly, mark the Loop mode for the correct animations described above in the EEntityState.

## Create Entity

All entities are located in Prefabs/Entities/"CharacterName". When you press the "Create Entity" button inside the editor a unique name will be required, also a logic that will define the type of the character (Hero, Minion, Tower, etc) and a 3D model object(.fbx object).

Once you have created a "Entity", will be necessary to refresh the character list pressing the refresh button at the top-left side of the screen. This will display the new Entity listed inside the "Library" section.

When you press on any item inside the "Library" section (top left side of the screen) a new section called "Entity Data" section will be displayed, in this section you can specify all the data that will define the selected character.

## Entity Prefab Builder

When you create an entity, the editor will generate the prefabs, and add the required components to work inside the game.

- **Add Entity Components:**

All entities must have an EntityLogic, you can select from all available logics ("CharacterLogic", "TowerLogic", "MinionLogic", "NexusLogic")

Add the current entity component "CharacterEntity" or "AIEntity"

Add NetworkIdentity component.

Add EntityAnimator component

Add EntityAbilities Component

Add EntityBehaviour

Add EntityCanvas

- **Add Entity Transforms**

Add all EntityTransforms with the corresponding position and it will put each transform in the approximated standard position. You can reposition all transform to fill your needs later by modifying the generated prefabs.

- **Add Box Collider**

The system will generate a basic collider that you can tweak later in the generated prefab to fit the size of the mesh.

The system will also add the corresponding data to the DataManager.

Tip: when you modify any transform position or colliders, remember to copy those changes in both prefabs "Client" and "Server".

## ***Entity Data Fields***

- Data Name:

String that presents the unique identifier of the character and can't be edited.

- Display Name:

String that will be displayed in game.

- Canvas:

Canvas Health prefab used by this entity. We include 3 standard canvas that you can use in most cases.

{CharacterCanvas, HealthCanvas, HealthCanvasSmall }

- Spawn Particle: Particle instantiate when the entity spawns in the game.

{CharSpawnParticle}

- Spawn Particle Position:

The entity transform position inside the model in which the "Spawn" particle will appear.

- Death Prefab Particle:

Particle instantiate when the entity die.

- Death Particle Position:

The entity transform position inside the model in which the "Death" particle will appear.

- Respawn Type:

The action that will execute this entity at the moment of respawn.

- Respawn Time:

Defines the time between the entity destruction and respawn event.

- Icon:

The portrait icon for this entity inside the game.

- Use Path Finding:

Defines if the entity will be using path finding inside the game.

- Team:

Specify the team in which the character belongs,

Note: in the case of characters there is no need to specify the team.

- Add Ability:

Add an ability data for this character. (Must be created in the ability editor)

- Character:

An enum that represents the character in the moba system. (in case you want to extend the system functionality)

- Base Attack Damage:  
Base entity attack damage.

-Ad Increase Per Level  
Attack damage added every time an entity level up.

-Base Ability Power:  
Base Entity Ability Power

-Base Armor:  
Base Entity Armor.

-Armor Increase Per level:  
Armor added every time an entity levels up.

-Base Ability Resistance:  
Base Entity Ability Resistance to spells.

-Health:  
Maximum entity health points.

-Health Increase Per level:  
Base Health Points increase every time an entity level up.

-Health Base Regeneration.  
Base Entity Health regeneration per second.

-Mana:  
Maximum entity mana points.

Mana Base Regeneration:  
Base entity mana regeneration.

Movement Base Speed:  
Base entity movement speed. (Recommended 1.4f.)

## Animations

You can add and preview animations inside the editor.

To add a new animation just select an action type, and press the "Add" button.

When you do this, a new "Object Field" will appear.

You can drag or search in the project a new animation clip and press the "Add" button. Once the animation is added you can preview it or remove it with "Play" and "Remove" buttons respectively.

You can also change the animation speed by changing the Animation Speed field.

## Saving / Loading Entity Data

The framework includes an option to save and load entity data to a file, using this you can save your current entity data or have multiple configurations for an entity.

There is two available buttons at the bottom of the Entity Fields that look like this.



Ad Increase Per Level	3
Base Ability Power	50
Base Armor	20
Armor Increase Per Level	3
Base Ability Resistance	0
Health	500
Health Increase Per Level	85
Health Base Regeneration	5
Mana	250
Mana Base Regeneration	5
Movement Base Speed	0
Add Ability 	
<div> <div>Save Entity Data</div> <div>Load Entity Data</div> </div>	

All game data are located in the Assets/GameData folder, each entity type uses a custom file extension.

("char") CharacterEntityData

("minion") MinionData

("nexus") NexusData

("tower") TowerData

("global") GlobalData

# Ability System

The framework contains a flexible ability system that allows to create almost any kind of ability without writing a line of code.



The screenshot shows the 'Ability Configuration' window in MOBASTORM. It is divided into three main sections: 'Add Ability', 'Ability Configuration', and 'Side Effects Configuration'.

**Add Ability:** A list of abilities with their IDs and icons. The 'Add Ability' button is highlighted with a blue plus sign.

**Ability Configuration:** A form for configuring an ability. The 'Ability Identifier' is set to 'Galilei\_R'. The 'Ability Description' is 'Mega Bomb'. The 'Ability Type' is 'S'. The 'Requires Target?' checkbox is checked. The 'Look Target?' checkbox is checked. The 'Target Type' is 'Position Target'. The 'Target Alliance' is 'Hostile'. The 'Max Targets' is '1'. The 'SkillShot Type' is 'Floor Skill'. The 'Indicator Prefab' is 'SimpleIndicator'. The 'Target Indicator' is 'SimpleFloorImage'. The 'Range Indicator' is 'SimpleRangeImage'. The 'Plays Animation' checkbox is checked. The 'Ability Animation' is 'Ability R'. The 'Anim Percent Launch' is '60'. The 'Cast Required?' checkbox is checked. The 'Ability processor type' is 'Launch Ability'. The 'Projectile Identifier' is 'Galilei\_R'. The 'Projectile Quantity' is '1'. The 'Projectile Speed' is '10'. The 'Ability launch Position' is 'Left Hand'. The 'Launch Particle' is 'Galilei\_ImpactBasicClient'. The 'Impact Particle' is 'FX\_Galilei\_R'. The 'Launch Sound' is 'FX\_Galilei\_R'. The 'Launch Sound Volume' is '100'. The 'Impact Sound' is 'FX\_Galilei\_R'. The 'Impact Sound Volume' is '100'.

**Side Effects Configuration:** A section for configuring side effects. The 'AbsorbDamageEffect' is selected. The 'Add Side Effect' button is highlighted with a green plus sign.

**Damage Logic:** A section for configuring damage logic. The 'Damage Ass Logic' is 'Damage Ass Logic'. The 'Can Damage towers?' checkbox is checked. The 'Damage Ass Range' is '2'. The 'Can level up?' checkbox is checked. The 'Base Damage' table is as follows:

	Lvl: 1	Lvl: 2	Lvl: 3
Base Damage	95	125	145
Base Ability	40	65	110
Range	5	5	5
ColdDown	15	15	15
ManaCost	50	50	50

The 'Deals (0) + (1) Attack Damage + (2) Ability Dam' section is empty. The 'Add Description Param' section is empty. The 'Param Number: 0' is 'Entity\_Base Damage'. The 'Param Number: 1' is 'Ability\_Attack Damage'. The 'Param Number: 2' is 'Ability\_Ability Power'.

## Create Ability

Too add a new ability just fill in the TextBox with the new Ability Identifier and click on the blue button to create the ability.



The screenshot shows the 'Add Ability' interface with annotations. A yellow arrow points to the 'Add Ability' button, which is labeled 'New Identifier'. A hand icon is pointing to the 'Add Ability' button. The 'Add Ability' button is labeled 'Add Ability:'. The 'Add Ability' button is labeled 'NewAbility Identifier'. The 'Add Ability' button is labeled 'ID:Galillei\_Q'. The 'Add Ability' button is labeled 'Select'. The 'Add Ability' button is labeled 'Ability Identifier'. The 'Add Ability' button is labeled 'Ability Description'.

## Remove Ability

The left panel contains all the ability data buttons, with their corresponding icons. You can click any tab to load and customize the ability or click on the red button to remove the selected ability.



## Ability Configuration Parameters

### Ability Identifier:

Unique identifier for the ability.

### Ability Description:

Description Name to be showed InGame.

### Ability Type:

Current ability type, current types. { Basic, Q, W, E, R, Passive, Special1, Special2, None}

### Requires Target:

Select this if the current ability requires targetting. The target could be a Position or Entity.

### Look Target

Select this if you want the entity to face the target while casting the ability.

### TargetType:

Select the type of the target that you want the ability to make effect.

{ EntityTarget, PositionTarget, SelfTarget, RandomTarget, AllTargets }

### Target Allegiance:

Select the allegiance to be used by the ability to filter targets or detect collisions.

{ Allied, Hostile }

### Max Targets:

Max targets the ability will try to cast the ability on.

### SkillShot Type:

Only showed If the TargetType selected is a PositionTarget.

FrontSkill: The ability will be casted from the entity to the target direction.

FloorSkill: The ability will be casted from the floor position.



### *Collision Detection*

Only showed If the SkillShot Type is FrontSkill.

Distance the ability will be using to detect collisions.

Recommended Setting: { 0.3f }

### *Indicator Prefab*

Indicator used to handle the gizmo logic to show the ability range, position and direction.

Tip: This prefab requires a component attached with an implementation of the Indicator class.

Default Value: { SimpleIndicator }

### *Target Indicator:*

Target Image Indicator Prefab loaded by the Indicator.

Tip: This prefab requires a component attached with an implementation of the ImageIndicator class.

Default Values: { SimpleFloorImage , SimpleFrontImage }

### *Range Indicator:*

Range Image Indicator prefab loaded by the Indicator to show the ability Range to the player.

Tip: This prefab requires a component attached with an implementation of the ImageIndicator class.

Default Values: { SimpleRangeImage }

### *Plays Animation:*

Mark this if the ability runs animations.

### *Ability Animation:*

This is the name of the animation clip used to launch the ability.

### *Anim Percent Launch:*

Animation percent that the ability will be launched. This way we can interrupt abilities or animations.

### *Cast Required:*

Mark this if the ability use a cast animation.

### *Cast Animation:*

This is the name of the animation clip used for the cast, this animation require to be a loop.

### *Casting Time:*

This is the total casting time in seconds the animation will be playing.

### *Cast Particle Identifier:*

CastParticle Prefab instantiated when the cast starts.

### Launching Type:

State in which the ability will be launched. { CastingState, AnimationState }

Ability Processor Type:

Default Values: { DamageImpact, LaunchAbility, CastSideEffectsOnly }

### Projectile Identifier:

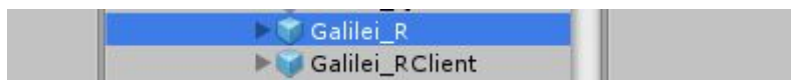
Ability Prefab Identifier to be instantiated.

Tip: This prefab requires a component attached with an implementation of the AbilityComponent class.

- Select the name of the prefab you want to use.



- Now Create two empty prefabs and name those following this nomenclature. For the server prefab just use the same ProjectileIdentifier name: Galilei\_R, for the client prefab we should use the same name adding "Client" at the end. The result should look like this.



- Now let's add the component to the prefabs to get the logic works, by default the framework came with some default scripts that gives you a great start to create your own projectiles.

#### **BasicFrontAbility.cs**

Move the projectile towards using the attacker direction.

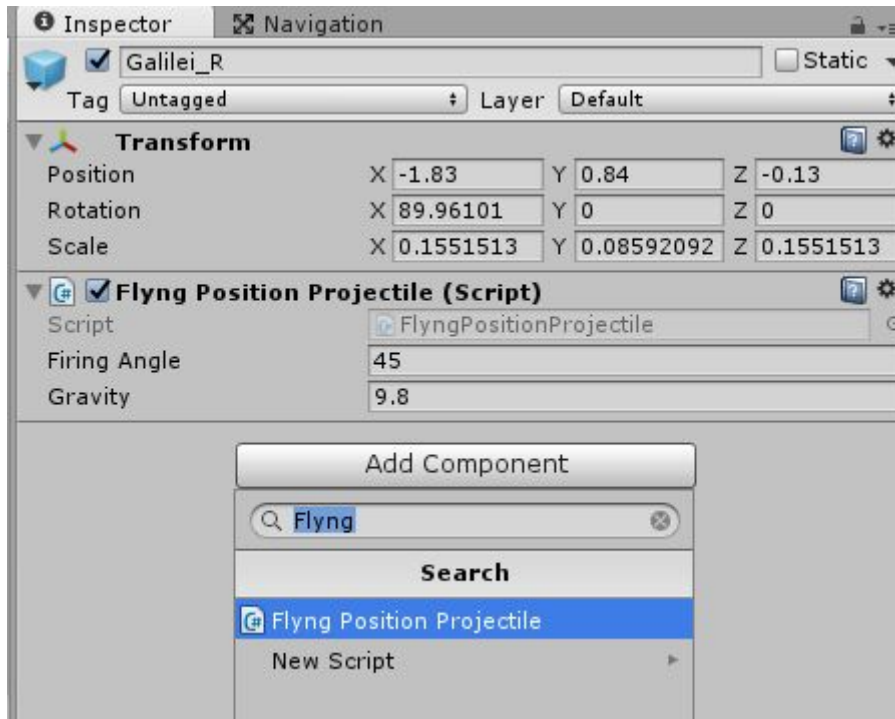
#### **FlyngPositionProjectile.cs**

Launch a projectile to the air using a firing angle and gravity, this is used to cast abilities to the floor.

#### **TargetProjectile.cs**

Use this component when the ability target is an entity, this will launch a ability that moves to a target and impact.

- To add the components just select the prefabs in the project, go to the inspector and click **AddComponent Button**, then find one of the previously mentioned scripts and added to both prefabs.



- The last step is add both prefabs to the Prefab Holder, go to Prefabs/Holders/ and find the Holder\_Abilities prefab.

## Adding Server Prefab

Go to the inspector and click **Add New Object**, you should see a green box with the new empty item data.

By default the quantity field should be 0 “We don't want to Preload any of the ability prefabs when the game starts”

Mark register with UNET. “This prefab will be marked as a Server Prefab and used by the SpawnManager to instantiate and recognize this prefab”

In the GameObject assign our ServerPrefab, in this case “Galilei\_R”.

## Adding Client Prefab

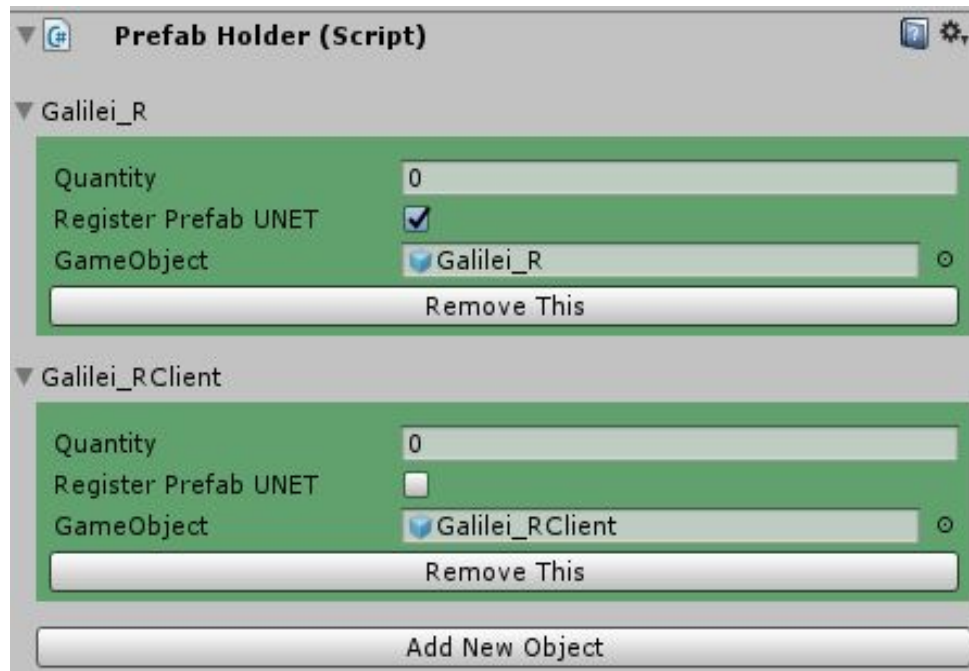
Go to the inspector and click **Add New Object**, you should see a green box with the new empty item data.

By default the quantity field should be 0 “We don't want to Preload any of the ability prefabs when the game starts”

Don't Mark register with UNET. "This is a client prefab, we don't want to register client prefabs with unet"

In the GameObject assign our ClientPrefab, in this case "Galilei\_RClient".

The result should look like this:



### Projectile Quantity:

Quantity of projectiles to be launched by the ability System.

### Projectile Speed:

The velocity of the projectile.

*Default Value : { 10 }*

### Ability Launch Position:

The TransformPosition of the model used by the ability to instantiate projectiles.

{ Head, RightHand, LeftHand, Center, Floor, Sky, Model }

### Launch Particle Prefab:

Prefab to instantiate when the ability is launched.

### Impact Particle Prefab:

Prefab to instantiate when the ability impacts an enemy.

### Launch Sound:

Sound identifier when the ability is launched.

### Launch Sound Volume:

Volume of the Launch Sound

Val: { 0- 100 }

### Impact Sound:

Sound identifier when the ability impacts an enemy.

### Impact Sound Volume:

Volume of the Impact Sound

Val: { 0- 100 }

### Damage Logic:

This is the damage logic the ability will run when the ability impacts.

Val: { DamageTargetLogic , DamageAoeLogic }

### Can Damage towers:

Select this box if the ability can impact or target towers.

### Damage AoE Range:

Range in units used to deal damage to all enemies in range.

### Can Level Up:

If the ability can be leveled (Mostly used by the Player Characters).

## Ability Level Data

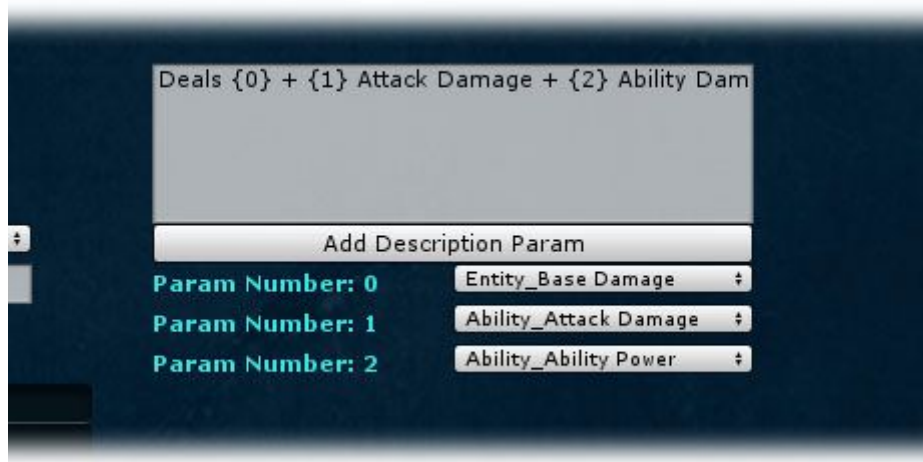
The ability contains some parameters that needs to be changed when the Entity level up.  
The maximum level can be configured in the Global Parameters.

	Lvl: 1	Lvl: 2	Lvl: 3
Base Damage	95	125	145
Base Ability	40	65	110
Range	5	5	5
ColdDown	15	15	15
ManaCost	50	50	50

- Base Damage:  
Current Entity BaseAttackDamage at level N.
- Base Ability:  
Current Entity BaseAbilityPower at level N.
- Range:  
Current ability range at level N.
- ColdDown:  
Current Ability ColdDown in seconds at level N.
- ManaCost:  
Current Ability Mana Cost to cast.

## Ability Description Parameters

The description parameters are used to show a tooltip to the player about each ability, this is also used by the **DamageProcess** to calculate the Total Ability Damage.



If you want the ability to deal any type of damage, just click Add Description Parameter and a new parameter will be added, then you can select what type of damage do you want to add.

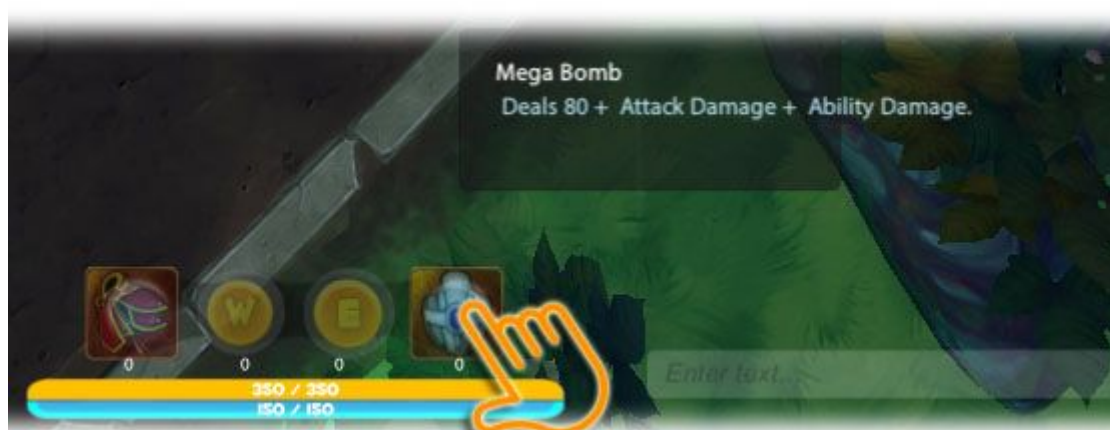
### Initial Parameters:

{ Entity\_BaseDamage, Entity\_BaseAbility, Ability\_AttackDamage, Ability\_AbilityPower }

Notice that each parameter added has a number, those numbers are used to link the DamageType value to the Description just by adding those ParamNumber in the description box between brackets {0} like in the previous picture.

Tip: if you leave the ability without parameters, the ability will deal no damage.

If you leave the mouse over the ability icon, a tooltip will popup showing the Ability Description Parameters.





# Side Effects Configuration

On the Top Right of the Ability Editor Window you will find a module to add or remove side effects.



You will see a popup with the list of all available Side Effects, the current implementation uses reflection to find all classes that implement the class SideEffect.

To **ADD** a new side effect just select the side effect from the list and click on Add Side Effect, when you add a new side effect all the custom parameters will appear on the editor.



To **REMOVE** a side effect just click on the red button RemoveSideEffect.



The framework comes with some common side effects that you can use to create your own side effects.

### **StunEffect.cs**

Stuns the target for an amount of time.

Duration: Total duration of the effect in seconds.

### **DotEffect.cs**

Deal damage over time to a target for an amount of time.

DotType: Type of damage overtime { Acid, Bleed, Burning, Poison }

Damage: Total Damage to deal.

Duration: Total duration of the effect in seconds.

Intervals: Intervals the spell will deal damage.

### **MovementMod.cs**

Adds a modifier to the Movement Speed of the entity.

Duration: Total duration of the effect in seconds.

SpeedMod: The speed value to add. { Recommended values: From -0.3f to 0.3f }

### **BoostDamageEffect.cs**

Increase the total damage calculation by a percent.

Duration: Total duration of the effect in seconds.

BoostPercent: Damage Percent Increase {from: -100 to 100 }

BoostType: Type of damage to increase { BaseAttack , BaseAbility }

### **StandingInvulnerability.cs**

Sets a character invulnerable for an amount of time, if the character moves or change the status, the effect will be interrupted.

Duration: Total duration of the effect in seconds.

# Game Data Manager

In Moba Storm the configuration parameters and all entity data are centralized on the GameDataManager, this way we can serialize and export all configuration to a Json file or Xml.

## Global Config:

Contains all parameters for the Global Configuration of the framework.

MinionAggroRange:

Range used by AI to detect enemies.

MaxChasingDistance:

Max distance allowed for AI to chase enemies.

TargetPathUpdateRate

Delay in seconds to perform a target path update.

TargetUpdateRate

Delay in seconds to perform a target search.

WaveSpawnStartDelay

Delay in seconds to start spawning minions.

WaveSpawnDelay

Delay in seconds between waves.

AccumulativeExpTable

This is the table containing the total experience required for characters to level up.

## Entity Data:

Contains a list with all datas for characters

DataName:

Unique Identifier used to access the current data parameters.

DisplayName:

The entity Display Name

Prefab:

The prefab used by this entity. The prefab name must be only the name of the Server Prefab, the SpawnManager automatically instantiate the ClientPrefab on all clients.

CanvasPrefab:

Prefab used for the canvas, all canvas prefabs must have a CharacterCanvas component attached.

We include 3 preconfigured prefab configurations:

CharacterCanvas

Shows The Entity Name, Entity level, Health Bar and Mana Bar.

HealthCanvas

Only Shows the health ("Used mostly for towers and structures")

HealthCanvasSmall

A small version of the HealthCanvas ("Used for small entities like minions")

SpawnParticle:

Prefab particle to be instantiated when the character spawns.

SpawnParticlePosition:

TransformPosition to instantiate the SpawnParticle.

DeathPrefabParticle

Prefab particle to be instantiated when the character die.

DeathParticlePosition:

TransformPosition to instantiate the DeathPrefabParticle.

RespawnType:

Respawn entity choice.

Values

NoRespawn: when the entity is destroyed the model will stay on the position.

NoRespawnAndDestroy: when the entity is dead the entity will not respawn and the prefab is unspawn.

RespawnAndReset: when the entity is dead, it will reset his position to the initial and reset all data.

RespawnTime:

Time in seconds the entity will respawn.

Icon:

MobaEntity Icon Sprite identifier.

UsesPathfinding

Mark this if the Entity is going to move using pathfinding.

Team:

Choose between Neutral, Blue or Red.

BaseAd:

Entity base attack damage.

BaseAdPerLevel:

Base attack damage to add when the entity level up.

BaseAp:

Entity Base Ability Power.

Armor:

Base Entity Armor.

ArmorPerLevel:

Base entity armor to add when the entity level up.

ApRes:

Ability Power Resistance.

HealthPerLevel:

Health points to add when the entity level up.

HealthMax:

Max Entity health points.

HealthRegenerate:

Health regeneration each second.

ManaMax:

MAX entity mana.

ManaRegeneration

Mana regeneration each second.

Speed:

Entity Move Speed.

The system allows you to create custom datas just creating a class that implement the MobaEntityData.

## Sprite Database Manager

Contains a reference to all textures and sprites used by the framework.  
All sprites or textures must be referenced on the SpriteDatabaseManager prefab.  
The methods available to get the sprites or textures at runtime are.

*GetSprite(string sprite)*

Call this method to get any sprite from the database.

*GetTexture(string texture)*

Call this method to get any texture from the database.

If the asset is not found on the database, it will return a not found sprite by default.

## Damage Process

The framework contains a class named DamageProcess which is responsible of processing all damage formulas, and resistances.

Understanding the math behind MobaStorm will help you analyze the game better.

→ [Armor / M.Resist Calculation](#)

→ [Attack Speed](#)

→ [AttackDamage / AbilityPower](#)

→ [Damage Calculation](#)

→ **Armor Calculation**

**EntityArmor**: BaseArmor + itemArmor

**TotalArmor** = EntityArmor + ( **EntityArmor**\* ( BuffArmorPercentMod/ 100))

**ItemArmor**: Base value coming from items.

**BuffArmorPercentMod**: Percent modifier coming from Buff/Debuff.

## → Magic Resist Calculation

**EntityMagicResit**: BaseMagicResit + itemMagicResist

**TotalMagicResist** = EntityMagicResit + ( EntityMagicResit \* ( BuffMagicResitPercentMod / 100))

**itemMagicResist**: Base value coming from items.

**BuffMagicResitPercentMod**: Percent modifier coming from Buff/Debuff

- **Example:**

Galilei starts at lvl 1 with 34 armor, that's the **Entity Armor**, and he buys an item that provides 10 flat armor, that's the **Item Armor**, also when he enters battle, he receives an ally buff that boost his armor an 20%:

$$\begin{aligned}\textbf{EntityArmor} &= 34 + 10 \\ &= 44\end{aligned}$$

$$\begin{aligned}\textbf{TotalArmor} &= 44 + ( 44 * ( 20/100)) \\ &= 52.8\end{aligned}$$

This means Galilei will have a **TotalArmor** of 52.8, until the buff runs out , if it runs out :

$$\begin{aligned}\textbf{EntityArmor} &= 34 + 10 \\ &= 44\end{aligned}$$

$$\begin{aligned}\textbf{TotalArmor} &= 44 + ( 44 * ( 20/100)) \\ &= 44\end{aligned}$$

## → AttackSpeed

**AttackSpeed** = BaseAttackSpeed \* ( 1 + 4% \* NumbersLvlUp )

- **Example**

Galilei have a BaseAttackSpeed of 0.35 in lvl 1 ( means he will auto attack 0.35 times each second ) and he gains a 5% bonus AttackSpeed each times he reaches lvl 7

$$\begin{aligned}&= 0.35 * ( 1 + 0.04 * 6LevelsUps ) \\ &= 0.35 * ( 1 + 0.24 )\end{aligned}$$

$$= 0.35 * ( 1.24 )$$

$$= 0.43$$

Galilei at lvl 7 will have a AttackSpeed of 0.43

- **AttackSpeed Buff or Reduction**

$$\text{AttackSpeed} * (1 \text{ } \pm \text{ AttackSpeed Buff/Debuff})$$

- **Example**

Galilei receives an ally buff of 20% Attack Speed increase

$$= 0.43 * ( 1 + 0.2 )$$

$$= 0.43 * ( 1.2 )$$

$$= 0.51$$

Galilei will have an AttackSpeed of 0.51 after he receives the ally buff

- **Example**

Galilei receives an enemy debuff of 20% Attack Speed Reduction

$$= 0.43 * ( 1 - 0.2 )$$

$$= 0.43 * ( 0.8 )$$

$$= 0.34$$

Galilei will have an AttackSpeed of 0.34 after he receives the enemy debuff

## → **AttackDamage / AbilityPower**

- **AttackDamage Calculation**

$$\text{EntityAttackDamage} = \text{BaseEntityAttackDamage} + \text{ItemAttackDamage};$$

$$\text{TotalDamage} = \text{EntityAttackDamage} + (\text{EntityAttackDamage} * (\text{BuffAttackDamageMod} / 100))$$

**Entity**: Base Attack Damage coming from the entity.

**ItemAttackDamage**: Base value coming from items.

**BuffAttackDamagePercentMod**: Percent modifier coming from Buff/Debuff.

- **AbilityPower Calculation**

$$\text{EntityAbilityPower} = \text{BaseEntityAbilityPower} + \text{ItemAbilityPower}$$



**TotalAbilityPower** = EntityAbilityPower + (EntityAbilityPower \* (BuffAbilityPower / 100))

**Entity**: Base ability power coming from the entity.

**ItemAbilityPower**: Base value coming from items.

**BuffAbilityPower PercentMod**: Percent modifier coming from Buff/Debuff.

- **Example**:

Galilei starts at lvl 1 with 30 Attack Damage, that's the **Entity AttackDamage**, and then buys an item that provides 15 Attack Damage, that's the **ItemAttackDamage**, also in battle, an ally gives him a 15% Bonus Attack Damage:

**EntityAttackDamage** = 30 + 15  
= 45

TotalDamage = 45 + ( 45 \* ( 15 / 100 ))  
= 51.75

This means while Galilei have a 15% AttackDamage Buff, his **TotalDamage** will be 51.75, when the buff runs out, his total **TotalDamage** will be reduced 15%:

**EntityAttackDamage** = 30 + 15  
= 45

TotalDamage = 45 + ( 45 \* ( 0 / 100 ))  
= 45

## → **Damage Calculation**

**TotalADReductionMultiplier** = TotalArmor / ( TotalArmor + 100 )

**TotalIM.RReductionMultiplier** = TotalMagicResist / ( TotalMagicResist + 100)

- **Example:**

If you have 75 armor,

$75 / ( 75 + 100 ) = 0.4285714286$

**TotalReductionMultiplier** = 0.4285714286

This means you will mitigate 42% of the physical damage, if enemy have 100 AD and deals 1 auto attack:

**DamageCalculation** = AttackDamage - ( AttackDamage \* TotalADReductionMultiplier )

$$= 100 - ( 100 * 0.4285714286 )$$

$$= 100 - ( 42,85714286 )$$

$$= 57,14285714$$

This means the character getting an auto attack that deals 100 AD, will mitigate 42% of that damage receiving only 57 Damage

## Audio Manager

The framework comes with their own custom Audio Manager that allows to all game entities or any monobehaviour to reproduce 3d sounds without add audiosources to each entity.

The audio manager instantiate custom audio sources that handle the logic to reproduce the sounds in the correct way, those custom audio sources live in a Queue when they are inactive, there is no limit to play sounds, if all custom audio sources are busy, the manager just create a new one dynamically.

AudioManager.cs

Play2dSound(string soundKey, int volume, bool looping = false)

Play3dSound(string soundKey, int volume, GameObject sourceObj, bool looping = false)

Play3dSound(string soundKey, int volume, Vector3 position, bool looping = false)

## Moba Entity Component

The moba entity class is the base class for all game entities, characters, towers, nexus, minions.

The framework came with 2 standard implementations.

CharacterEntity.cs

Used by standard characters, contains the standard methods to handle the basic character setup, and initialize all parameters.

AIEntity.cs

Used to create all Non-character entities, towers, nexus, minions npcs or any other entity.

You can also create your own custom implementations of CharacterEntity and AIEntity to generate a complete custom character, for example can create a character with **Energy** instead of **Mana**.

## Logic Component

It's the core of all entities in the game, it handles all the logic to make the process player input, or run actions using AI intelligence.

The package includes 4 custom logics that you can use to create your game entities.

### **CharacterLogic.cs**

Contains all standard methods to process the player inputs, like select target, casting abilities, stop movement, show gizmo indicators.

- You can create your custom CharacterLogic that inherits from this class, to create your own customized character, or create custom controls like jumping? or flying? you have endless possibilities with this architecture.

### **MinionWaveLogic.cs**

Contains the minions logic, basically the minions logic consist in a character that follows a path and detect enemies in range, when an enemy enters on AggroRange the minion will try to position in range of attack and cast his basic ability.

### **TowerLogic.cs**

Tower logic is a static entity that is always trying to find enemies in range, and attack them using a priority ("Characters Should be the last character to attack")

### **NexusLogic.cs**

The nexus logic is basically a static entity, then this entity is destroyed the game ends.

## Entity Behaviour Component

Behaviours in MobaStorms were created to give the system the capability to create EntityBehaviours and reuse those by multiple Moba Entities.

The system came with a set of basic behaviours to meet the requirements to all basic entities on the game, those are:

### **AbilitySimpleTarget**

Contains a complete implementation to cast an ability to a MobaEntity target, this handles all the animation parameter, like casting, launching ability and cast side effects.

### **AbilitySimplePosition**

Contains a complete implementation to cast an ability to a Position target, this handles all the animation parameter, like casting, launching ability and cast side effects.

### **AITargetBehaviour**

This is used by minions, this behaviour runs when AIEntity finds a target, and this implementation move the Entity on path to get to the correct node in attack range.

### **TowerTargetBehaviour**

Used by towers to cast a the basic attack from towers, when the target is dead or out of range the behaviour is ended.

## **Entity Abilities**

Entity component that handles all entity abilities, contains a reference to all entity abilities and handle all the process related to them like ColdDown, AbilityLevel Up, Ability Impacts, apply and process all SideEffects.

### **ApplyAbilityEffects(Ability ability, MobaEntity attacker)**

Used to apply all side effects from an ability to the current entity.

### **TryApplySideEffect(Ability ability, SideEffect effect, MobaEntity attacker)**

Used to apply any side effect to the current entity.

### **RemoveSideEffect(SideEffect effect)**

Removes a side effect from the current entity.

### **SpawnSideEffectPrefab(SideEffect effect, EntityTransform entityTransform)**

Spawns a side effect prefab on the server and sends an RPC to spawn the same effect on all clients.

### **UnSpawnSideEffectPrefab(SideEffectPrefab prefab)**

Unspawn a side effect prefab on the server and sends an RPC to unspawn the same effect on all clients.

# Entity Animator

Entity component that handles all entity animation states, and play all animations on the clients, to minimize resources on the server side, by default animations are not played on the server.

All animations in the moba are controlled by the server using the animation length.

States and animations are defined like this.

**public enum EEntityState**

```
{
    Idle = 1,
    Run = 2,
    Basic = 3,
    AbilityQ = 4,
    AbilityW = 5,
    AbilityE = 6,
    AbilityR = 7,
    Dead = 8,
    Recall = 9,
    Emote = 10,
    CastingQ = 11,
    CastingW = 12,
    CastingE = 13,
    CastingR = 14,
    Stun = 15,
    Custom1 = 16,
    Custom2 = 17,
}
```

You can add more custom states if needed.

**public float ChangeState(EEntityState newState)**

Method available to change a entity state on the server side.

**public void ClientAnimationChange(EEntityState newState)**

Method available to change a entity state on the client side and play the corresponding animation.

# Entity Canvas

Entity component that contains a reference to the current CharacterCanvas. Wich handles the logic to show the Health Bar and Mana Bar from entities, this also shows the name and entity level.

The character canvas can be customized on the **CHARACTER EDITOR**

**ShowFloatingText(FloatingText.FloatingTextType type , string text)**

Shows a floating text for all common cases, types allowed.

```
public enum FloatingTextType
{
    GenericFloatingText,
    DamageFloatingText,
    GoldFloatingText,
    SideEffectFloatingText,
}
```

# Task Manager

All game entities came with a task queue, used by client entities to enqueue tasks and execute them in a determinated simulation step given by the server, ("Fixed Step").

Currently we use this queue to execute major entity actions on clients and keep all the actions synchronized between server and clients, some of those tasks are:

## **CalculatePathTask**

Calculate a path and move the character.

## **SetRotationTask**

Rotates the client entity to a Quaternion.

## **StopPathTask**

Stop client entity pathfinding move and clear the path.

## **SetAnimationTask**

Runs a animation on the client entity.

## **TargetAbilityTask**

Execute a target ability on the entity client.

### PositionAbilityTask

Execute a position ability on the entity client.

### AbilityHitTask

Used when an ability impacts.

### SetNodeResidentTask

Used by minions, to lock the current cell.

## Inventory System

Added in Moba Storm 1.1

This system includes a complete customizable inventory system, that works in parallel with the Ability System to create Active Items.

Equippable items, Consumable Items can be created, those can be passive or active.

You can find the editor inside of Unity Menú **MobaStorm / InventoryEditor**



### Item Identifier

Unique Identifier for this item.

### Item Description

Description to be shown InGame.



## Item Cost

Purchase cost for this item.

## Item Stacks

Define if the item is stackable just by set the amount greater than 0.

{value = 0} Not Stackable

{value = 3} 3 Stacks Allowed

## Item Type

{Consumable, Equipable }

## Item Use


{ Active, Passive }.

## Active Ability


If the current item is active, you need to link in this box any Ability Already created on the Abilities Editor.











## Passive Effects


Here you can add any Side Effect available. In the next screen we will show you an example of an item with two Passive Effects to add MxHealth and ArmorMod.



### Item Configuration

Add Ability:  

- ID:Potion\_Low 
- ID:Potion\_High 
- ID:Axe\_Small 
- ID:Axe\_Big 
- ID:Shield\_Small 
- ID:Shield\_Big  
- ID:Boots 
- ID:Cloak\_Small 
- ID:Cloak\_Big 



Select

Item Identifier: Shield\_Big

Item Description: Equip: Heals (100) Max H

Item Cost: 200

Item Stacks: 1

Item Type: Equipable

Item Use: Passive

None (Ability Base) 0

### Passive Effects

AddEntityModEffect

Add Passive Effect

Effect Type: AddEntityModEffect

SideEffect Identifier:

Effect Prefab Name:

Side Effect Sound:

Effect Sound Volume: 0

Effect Sound Volume:

Launch Effect Pos: Head

Use Type: Item Passive

Mod Type: Max: Health

Modifier: 100

Remove this effect

Effect Type: AddEntityModEffect

SideEffect Identifier:

Effect Prefab Name:

Side Effect Sound:

Effect Sound Volume: 0

Effect Sound Volume:

Launch Effect Pos: Head

Use Type: Item Passive

Mod Type: Armor Mod

Modifier: 80

Remove this effect

New Side Effect Added:

### AddEntityModEffect

Adds any modifier to the current MobaEntity Target Stats.

Here is the list of the MODS available.

Health,  
MaxHealth,  
Mana,  
MaxMana,  
SpeedMod,  
CoolDownPercentMod,  
HealthRegenerationMod,  
HealtRegenerationPercentMod,  
ArmorMod,  
ArmorPercentMod,  
MagicResMod,  
MagicResPercentMod,  
AttackDamageMod,  
AttackDamagePercentMod,  
MagicDamageMod,  
MagicDamagePercentMod,  
GoldBonusMod,

### InGame Item Store Screen



