Moba Storm Photon Multiplayer Framework
Documentation V 1.0

Technical specs

Table of Contents

Introduction

Thanks for your purchase of this framework!
We hope you like this package; 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
If you find a bug please send detailed information to 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 mother tower!!!!

- Experience splits between the killers
- Gold when you kill an enemy
- Teleport to base
- Standard MOBA keys QWER
- Abilities can slow enemies, stun, dot, hot, teleport… you can even write your custom scripts to customize abilities.
- Minimap
- Item Management and inventory.
- With this collection of scripts and assets you can easy setup new players, enemies, spells, and much more.
-

Technical specs

- Multiplayer Framework using Photon Cloud.
- Menu animations with different behaviors
- Server list where you can see the actual players and their names
- Char selection window with portraits and sounds for each character
- Enemy ia, waypoints integration, using unity navmesh.
- 2 custom characters with animations and custom spells
- You can create new characters, new enemies with just a few lines of code, even you can fully customize your spells and attacks creating your custom scripts, you want a spell that stun a player? And deal dot at the same time? Or maybe a spell that teleports you and deals aoe dmg??? You can do it easily with a few lines of code there is no limits!!!
- You need mid to advance adknowledge of C# and unity3d to work with this framework because it has no editor tools or drag-drop style.

Photon Overview

The world's leading multiplayer cloud service provides all core features needed to match & connect tens of thousands of users in realtime and turnbased games. Whether you are a one man team or a AAA studio: Photon is the fastest way to build and launch multiplayer games globally.

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
Photon libraries.
https://www.assetstore.unity3d.com/en/#!/content/1786

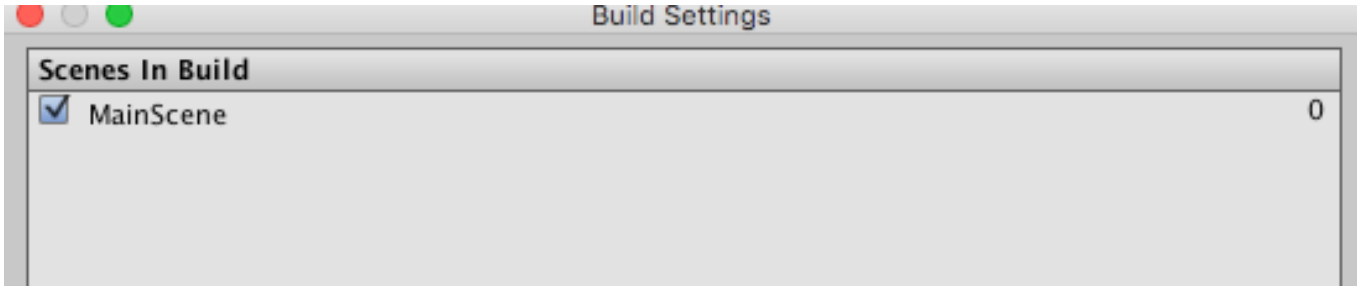
Installation

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

Photon Setup

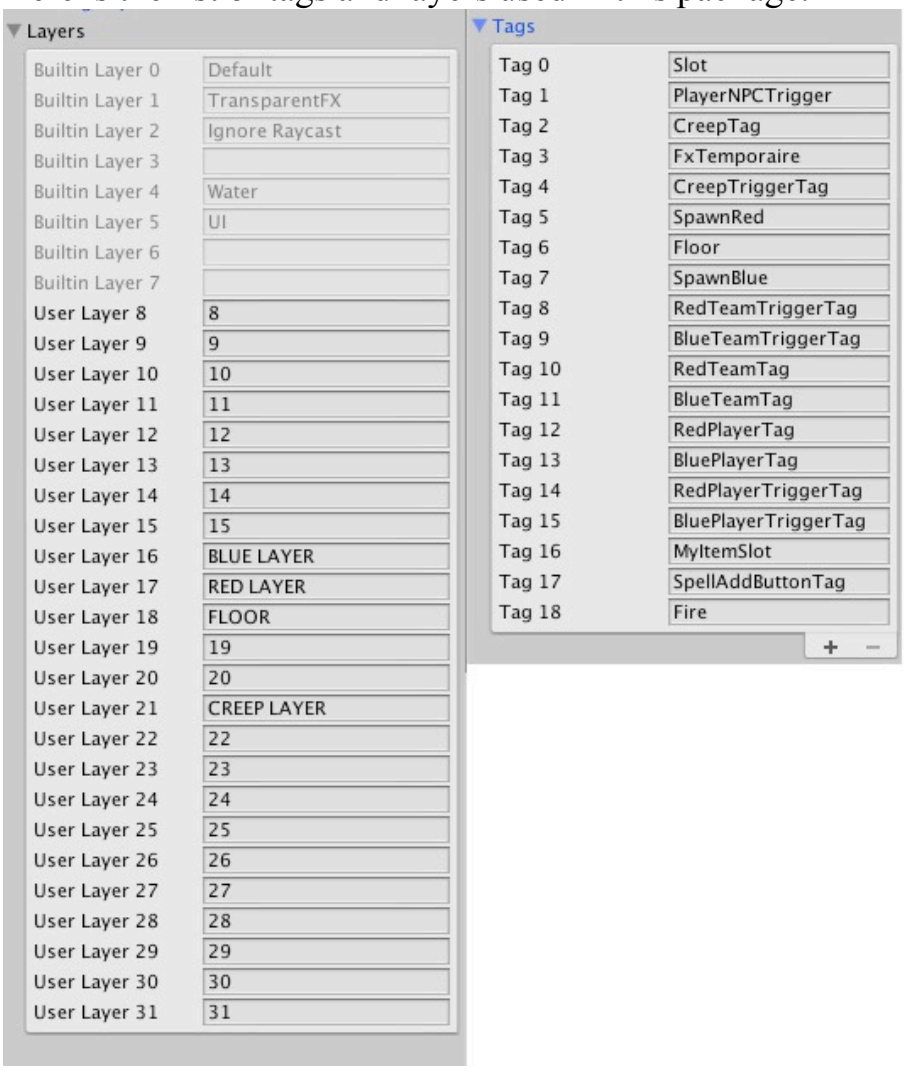
Photon Unity Networking (PUN) is really easy to setup. the PUN Wizard will pop up. Register for a new (free) Photon Cloud account by entering an email or paste an existing AppId into the field. Done.

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



Quickstart

Here is the list of tags and layers used in this package.



Layers

Builtin Layer 0	Default
Builtin Layer 1	TransparentFX
Builtin Layer 2	Ignore Raycast
Builtin Layer 3	
Builtin Layer 4	Water
Builtin Layer 5	UI
Builtin Layer 6	
Builtin Layer 7	
User Layer 8	8
User Layer 9	9
User Layer 10	10
User Layer 11	11
User Layer 12	12
User Layer 13	13
User Layer 14	14
User Layer 15	15
User Layer 16	BLUE LAYER
User Layer 17	RED LAYER
User Layer 18	FLOOR
User Layer 19	19
User Layer 20	20
User Layer 21	CREEP LAYER
User Layer 22	22
User Layer 23	23
User Layer 24	24
User Layer 25	25
User Layer 26	26
User Layer 27	27
User Layer 28	28
User Layer 29	29
User Layer 30	30
User Layer 31	31

Tags

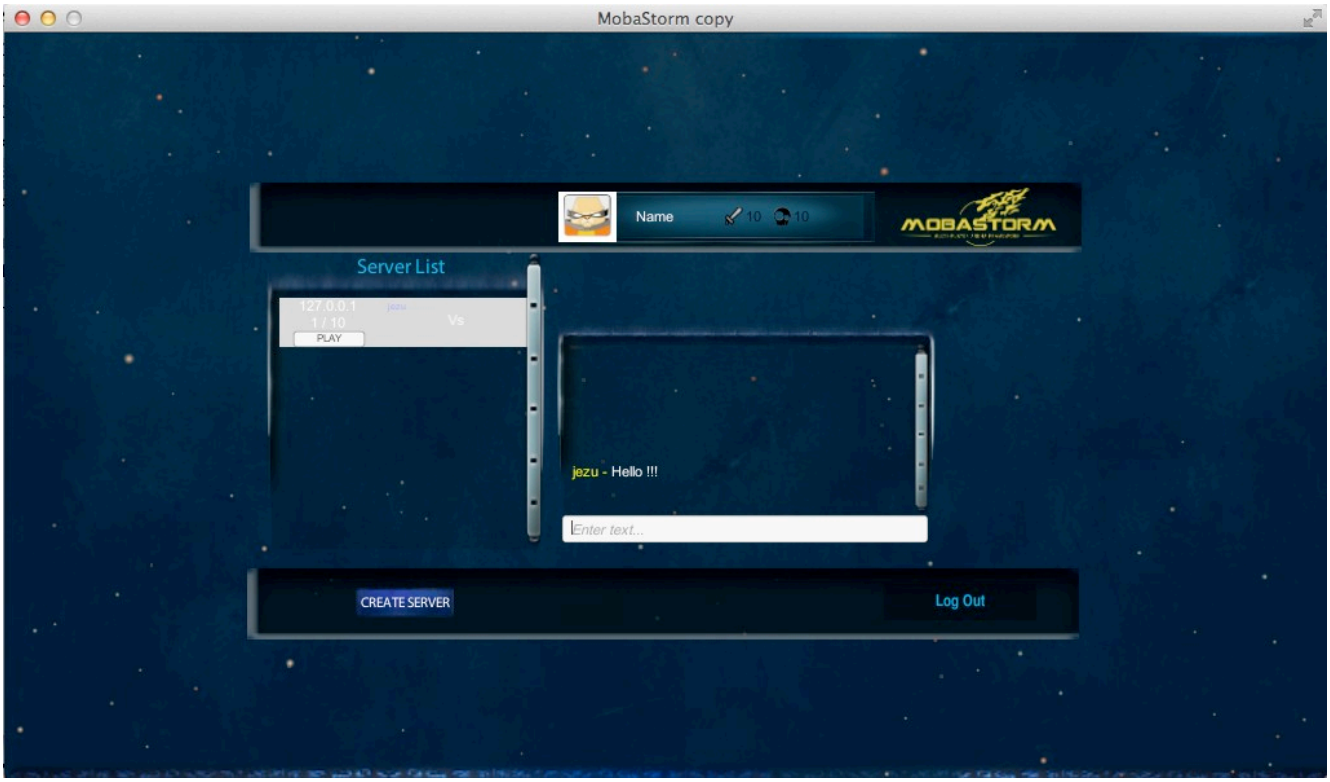
Tag 0	Slot
Tag 1	PlayerNPCTrigger
Tag 2	CreepTag
Tag 3	FxTemporaire
Tag 4	CreepTriggerTag
Tag 5	SpawnRed
Tag 6	Floor
Tag 7	SpawnBlue
Tag 8	RedTeamTriggerTag
Tag 9	BlueTeamTriggerTag
Tag 10	RedTeamTag
Tag 11	BlueTeamTag
Tag 12	RedPlayerTag
Tag 13	BluePlayerTag
Tag 14	RedPlayerTriggerTag
Tag 15	BluePlayerTriggerTag
Tag 16	MyItemSlot
Tag 17	SpellAddButtonTag
Tag 18	Fire

Building and playing the game

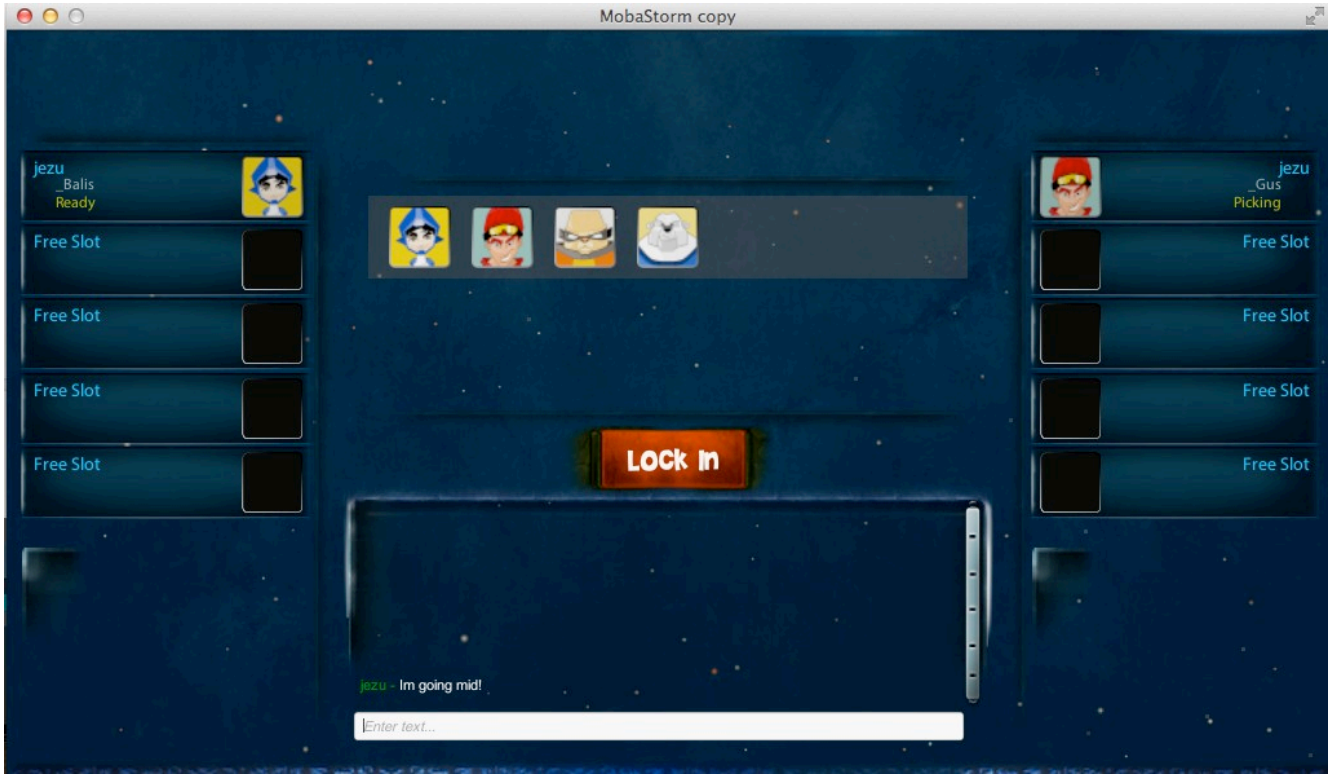
From the Unity main menu, Click "File -> Build Settings".
In the "Platform" box, select "PC, Mac & Linux Standalone"
Click the "Build" button.

Run the game and you are ready to play the game!

After you log into the game a server window will show up, the list of available servers with a grey box showing the name of the server, players connected.
If you want to join a server just click on the button PLAY. Or create server if you want to create your own Game Server.



Now you are in the character selection window, which allows you to select a character by clicking on the portrait window (MID BOX) after you select a character you can 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, Character Picked, and Picking Status.
Now when you are ready just click LOCK IN!! And you are ready to kill some waves and towers.



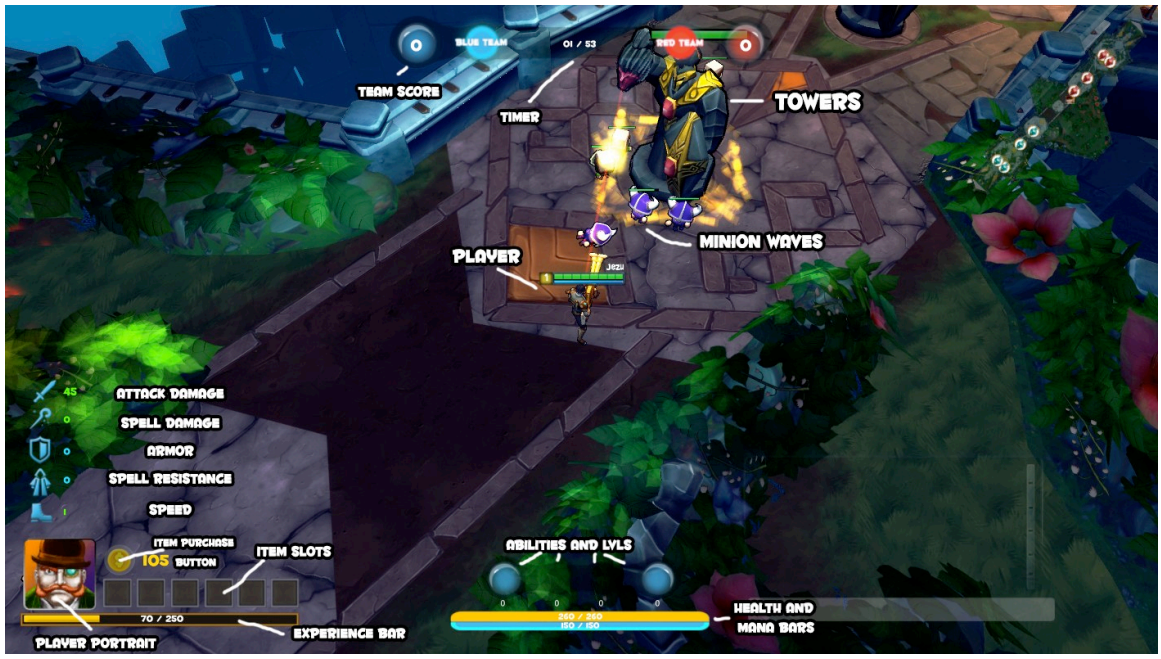
The Game

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 health bar.
The game keys works 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 damage, magic damage, current gold, armor, magic res, and experience of the player.
Bottom mid panel
It shows the health and mana of the player, it also shows the spells and the CD of each one when you use it.



Framework overview

Multiplayer

The core of this multiplayer connection system is allocated on the ClientGameObj, it contains scripts that connects to the lobby lobby and game servers.
If you want additional info or any question about the Photon api, you can go to the ExitGames site which contains a lot of manuals and reference about this multiplayer tool.

Menu / UI

The canvas gameobject contains a gameobject called MainMenu which is the center of all ui showed on the framework, this script activates the current menu to show up to the user.

Minion wave spawn

There is a few gameobjects called MinionManager that contains a script called MinionSpawnManager.cs that is responsible of creating all the enemy waves on the game. There you can specify which prefab it will instantiate, team, waypoints, health, lvl , exp, gold to give, attack dmg, ap damage, armor, magic resist. If you want to change the level of the wave, or anything you only need to change those values.

Player

The core of the player logic is managed by PlayerSpell.cs, PlayerControllerRTS.cs and PlayerStats.cs.
PlayercontrollerRTS.cs
Contains all info about the position of the player, rotation, animation, collision mask, state of the player and all the spec of each spell, this manage the spells projectors, and contains all the logic of the player attacks.

This script also communicates with the CDR gameobjects to show the cold down reduction of each spells availables.
The states of the player are managed in this order.

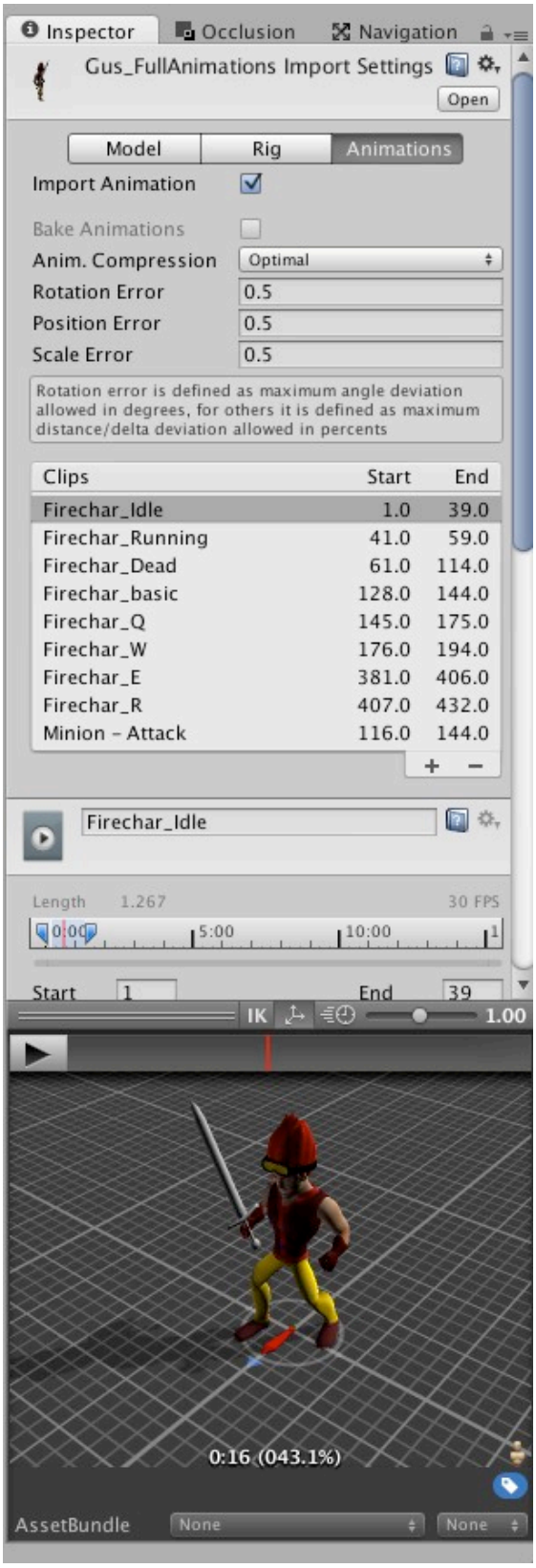
```
public enum PlayerState
    {
    idle,
    running,
    chasing,
    attackingBasic,
    attackingQ,
    attackingW,
    attackingE,
    attackingR,
    dead,
    recall
    retreat

    }
```

When the client clicks on the floor or click over an enemy the SendAttackToServer() is called, sending info about position of the collision, the state of the player, and the targetname in case of an enemy.
PlayerSpell.cs
This script is used to launch attacks of spells called by the animation event Attacking(int Type); later the script calculates what prefab it should instantiate using GetAbilityPrefab(); and where it should spawn using GetLaunchPos();. PlayerStats.cs
This script contains all the stats about the players, health, mana, team, exp, lvl, armor, magic resist, speed, attack damage, ap damage.
UpgradeSpellToClients()
Is used to add a new ability to the player, sending the type you can level up Q, W, E, or
R.
DrainHealth ()
Is responsible in dealing the damage to the player, this method is called by any gameobject that deals dmg to the players, when the player is dead this method add a kill
value to the attacker also it calculates the experience to give and divide by all the attackers stored in the array attackersList.
Scene Setup

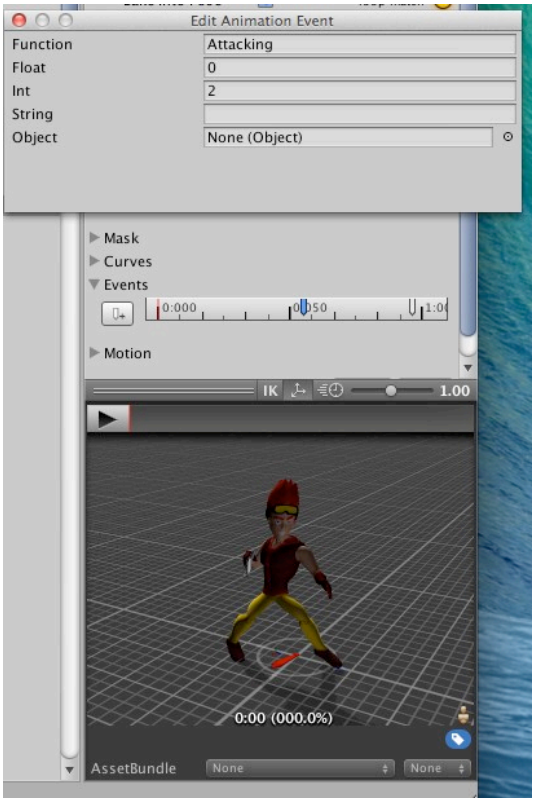
Character Creation

1 - Resources

Animated model in FBX format
You will need an FBX with a multiple animation clips or a separated animation FBX.
Set the animation name for each clip as described below changing the empty for your character name.
Empty_Idle (Loop animation)
Empty_Running (Loop animation)
Empty_Dead
Empty_Basic
Empty_Q
Empty_W
Empty_E
Empty_R



Set the animation events
All the abilities of the characters needs to call two Events

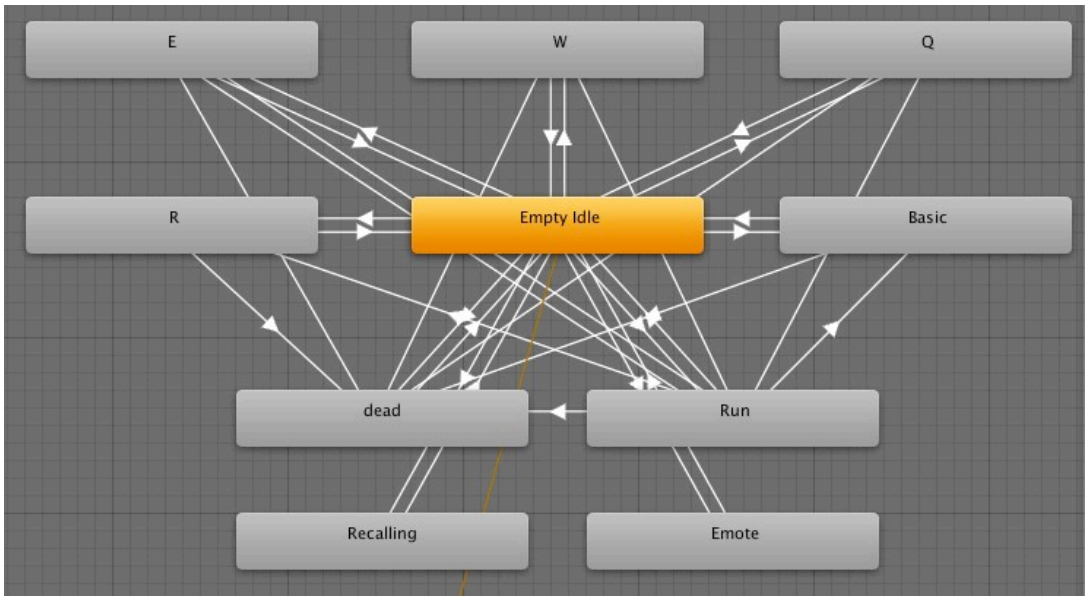


Attacking(int value)
Launch the current ability on the PlayerControllerRTS script. The int value put this event on the time that you want to instantiate the ability prefab.
Basic animation Attacking(3)
Q animation Attacking(4)
W animation Attacking(5)
E animation Attacking(6)
R animation Attacking(7)

ExitAnim()
Unlock the character animation on the PlayerControllerRTS script.

2 - Animation controller

Go to the project folder **Prefab/Controller.**
Duplicate the **Controller_Empty** to the project folder Misc/ and change the “empty” to your character name.
Fill each state animation with your current character animations.
The image shows the current empty animator.



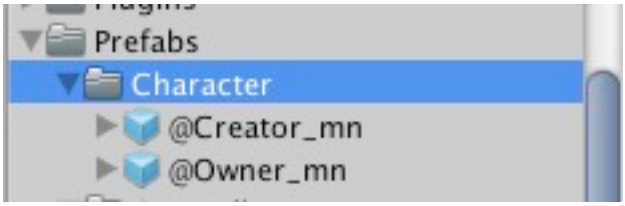
3- Character portrait image

A image of the character portrait, you can use a PNG with 137x137 pixels, or place the image inside the file included in **Textures/UiElements/Portraits.png**
After that you need to set the image as a sprite.
(You can find additional resources about doing this in the unity website)

4- Prefabs creation

You need to a prefab @Owner_mn and place your custom character name before the prefix.

Go to the folder Prefabs/Characters, here you will find two prefabs called **@Owner_mn** drop this prefab to the hierarchy window and set your custom character name before the prefix.

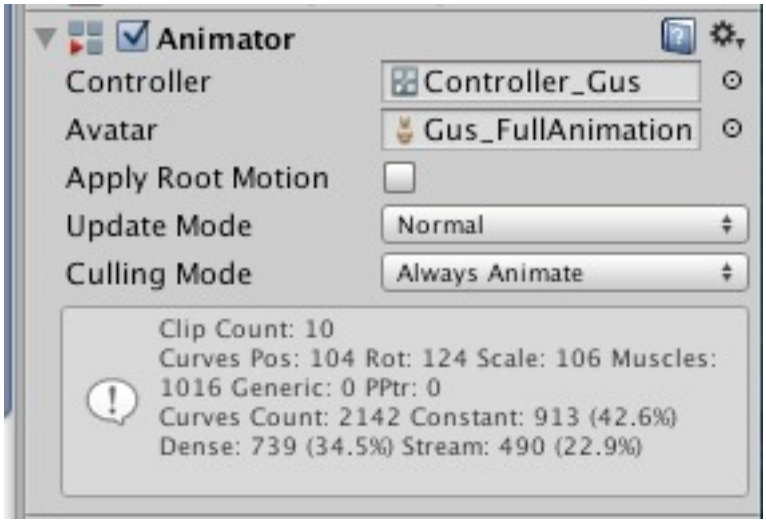


Example:
CustomChar@Owner_mn

Place your custom FBX model inside both prefabs and remove the animation component, (remember to reset the position and rotation, and make sure the rotation is facing the prefab positive local Z axis)

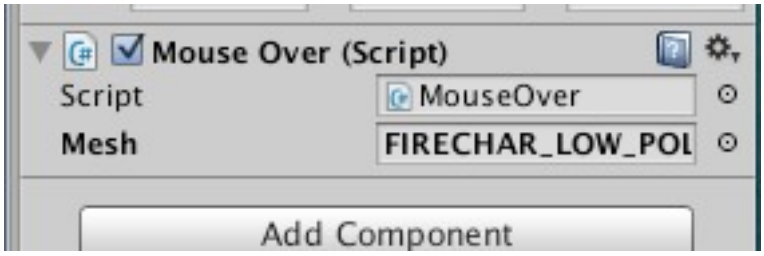
When your character is ready put the prefab in Resources/Characters (Your need to do this to get loaded by the SpawnScript.cs)

Look at the Animator component of the prefab
Change the controller for the previously created controller.
And change the avatar for the name of the avatar corresponding to the imported FBX.



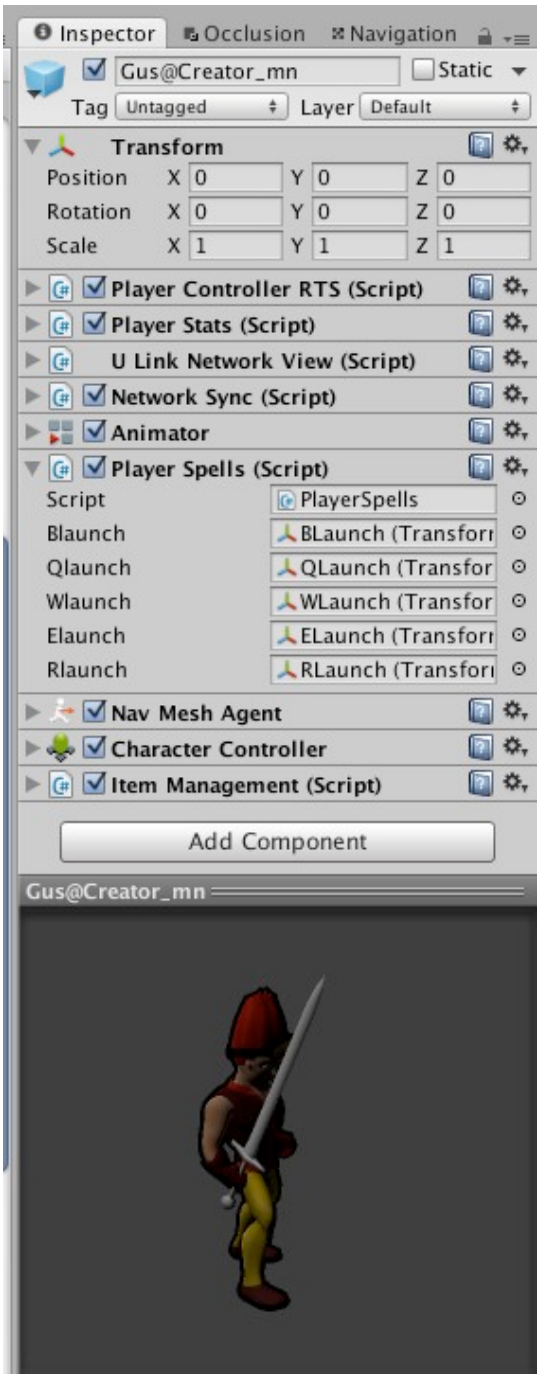
Inside this prefab you will find a gameobject called **Trigger**, make sure your character FBX fits the size of the Box Collider attached to the **Trigger** gameobject.
See the image below.

In the @owner prefab there is a component attached to the trigger called Mouse Over, Here you need to assign your character mesh by dragging the object to the field Mesh.



Change the shader component of the character to Standard Outlined, adjust the outline extrusion and set the outline color to black.

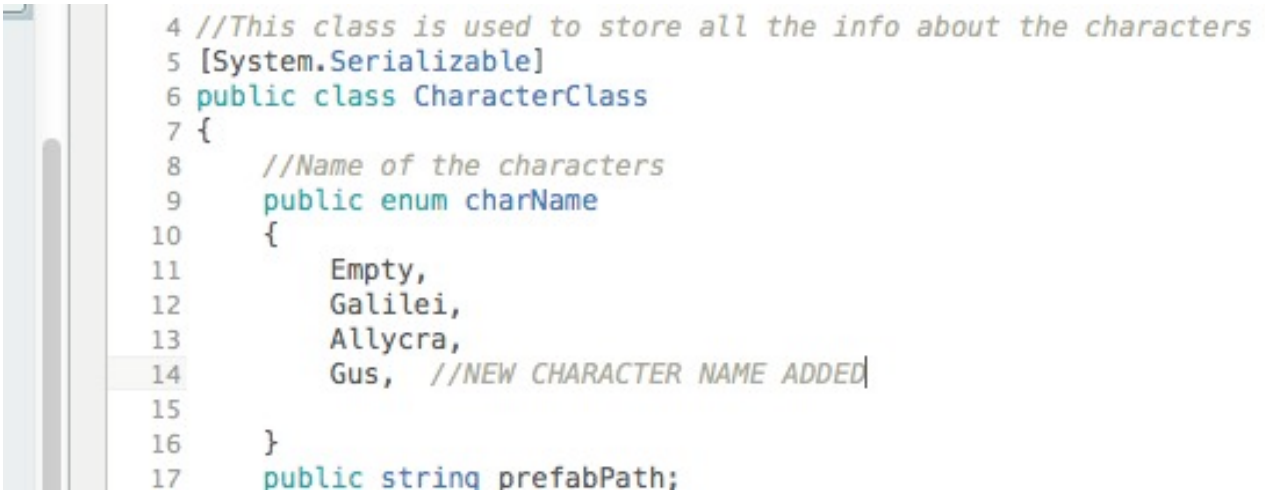
In the Creator prefab create 4 empty game objects corresponding to each launch spell position and rename accordly **BLaunch, QLaunch, WLaucnh, ELaunch, RLaunch**, this will define the start position of the spell, you can place the object anywhere, on the root, in the hand, weapon, on the sky, even you can put a custom script on this empty gameobject to calculate a custom launch position.
Now drag each object to the **PlayerSpell** component inside their corresponding field name.
The result should be looking like the image.



5- Character class

Go to the project folder **Scripts/Abilities/Characters** and open the script **DmgDataclass.cs** Here you will find a public enum called charName, just add the name of the character to the enum as you see in the picture, we just added the name Gus.

Note: remember to put the coma after the names of the characters.



```
4  //This class is used to store all the info about the characters
5  [System.Serializable]
6  public class CharacterClass
7  {
8      //Name of the characters
9      public enum charName
10     {
11         Empty,
12         Galilei,
13         Allycra,
14         Gus,  //NEW CHARACTER NAME ADDED
15
16     }
17     public string prefabPath;
```

6 - Character Data

Now lets fill all the data about the character.
On the creator prefab look for the component PlayerStats
Mark the box named (Is a player)
Mark the box named (Is creator or server)

MaxHealth Total
health **Health**
Reg. Rate
Regeneration rate
Mana
Total mana
ExpToGive
The experience points this player will give when gets killed by another player
GoldToGive
The amount of gold this player will give when gets killed by another player
Armor Res
Initial Attack Damage reduction
Spell Res

Initial Spell Damage reduction

MoveSpeed

Initial movement speed of the player (recommended 1.4)

This Respawns?

Check this and set the Respawn Time in seconds, this will determine the time to respawn after gets killed.

Stats Drop Down

Here you are going to fill all the info about each player data and custom attacks.

Prefab Path

Is the name of the character. It will be used to set the folder reference to load player resources.

Character Name

Set the character name.

Char portrait

Choose the portrait sprite you just created. (Remenber to set this on the @OWNER prefab.

Weapon Lvl

The initial weapon level of each ability, use value of 1 for the basic attack, and 0 for the Q, W, E, R.

Weapon Name

Sets the weapon name of each ability

Weapon type

Sets the attack type or general behaviour of the spell.

Locked_Shot_Range

Commonly used for the basic attack, this type of ability will only be casted directly to the enemy at a determinate distance, and uses the enemy object to set the destination.

Skill_Shot_Floor

This type will be casted on the floor, and uses a floor vector 3 position to set the ability destination.

Skill_Shot_Front

This type will be casted to the front of the player, and uses the vector3.forward as the direction of the ability.

AoeInstant

Instant cast any ability in the center of the player without any projectors or calculations, commonly used for any heal pots or buffs.

LaunchPos

Here you choose the launch position of each ability.

Floor

Sets the floor as the initial ability position.

Sky

Sets the initial ability position to +10 relative to the Y axis of the character.

Center

Sets the center of the character as the initial launch position.

B

Uses your previously empty gameobject called BLaunch

Q

Uses your previously empty gameobject called QLaunch

W

Uses your previously empty gameobject called WLaunch

E

Uses your previously empty gameobject called ELaunch

R

Uses your previously empty gameobject called RLaunch

ManaCost

Mana cost of the ability

Ad

Attack Damage of the ability

Ap

Spell Damage of the ability

Range

Range of the ability

Cdr

Cold down reduction of the spell

CdrTStamp

Cold down reduction time stamp (leave it to 0)

7-Add the character to the pick selection

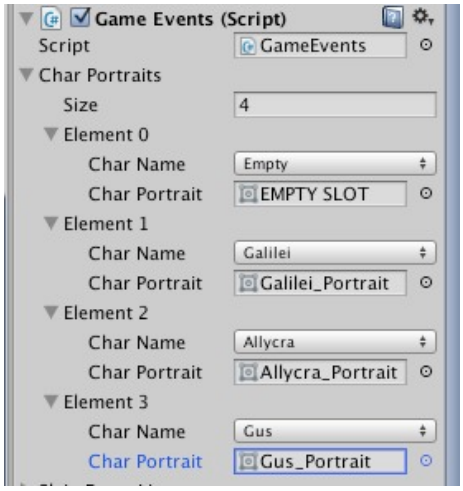
Go to the project folder **Scripts/Ui/Selection** and open the script named **PickSelection.cs**

Under the **void Start ()** method you will se a you need to call the method **addCharToList**, with the name of the character you just created as it shown in the picture.

```
void Start () {

    //YOU NEED TO ADD THE NEW CHARACTERS HERE IN ORDER TO SHOW IN THE PICK SELECTION
    AddCharToList(CharacterClass.charName.Galilei, true, portraitDir + "Galilei", prefabDir + "Galilei@", "Galilei");
    AddCharToList(CharacterClass.charName.Allycra, true, portraitDir + "Allycra", prefabDir +  "Allycra@", "Allycra");
    AddCharToList(CharacterClass.charName.Gus, true, portraitDir + "Gus", prefabDir +  "Gus@", "Gus");
```

8- Character Portraits

In the game hierarchy select the gameobject **GameManager_mn**, in the **GameEvents** components increase the CharPortraits list size and select the new CharName and the current portrait.



In the project folder **Prefabs/UiElements** you will find a prefab called **EmptyPortrait.prefab**

Copy the **EmptyPortrait.prefab** to **/Resources/UiResources/Characters** and rename the prefab to the name of the character.

In the image component, set the source image to the previously created Portrait sprite.

On the **Char_Pick** component, set the **CharName** to the name of the character. (Remember: you need to set previously the name of the character on the enum var **charName** in the script **DmgDataclass.cs**)

Abilities and projectors creation

Abilities and projectors are loaded from a default predefined resources folder by using the string **PrefabPath** defined in the **@creator** character.

All the ability prefabs must be placed on the project folder
Resources/Abilities/(character name)/

The projector prefab must be placed on the project folder
Resources/Abilities/(character name)/Projectors

All the ability scripts inherit from the **MobaStormAbility** class. Wich contains the core to receive all the data needed for any ability casted on the game and trigger the basic methods to make things easier for you.
When the ability is instantiated, an photon gameobject is instantiated with extra parameters included witch are available to handle almost any ability behavior you want.
Here are described the variables that you have available to work with.
team
The team of the player who cast the ability
originatorGameObj
The player gameobject who cast the ability
originatorCharName
The player name that cast the spell
ad
The total attack damage of the ability
ap
The total spell damage of the ability
initialPos
The initial vector3 position of the ability
destinationGameObj
The player destination gameobject if the ability was casted over an enemy.
floorPos
The vector3 destination of the spell in case the spell is a floor skill shot or front skill shot. This is the same position raycasted by the player mouse position to the floor.

mask
The layer mask that allows which enable you only to hit an enemy layer or neutral monsters.

OnStart ()
Called after the ability has received all network data.

DestroyMyselfAfterSomeTime()
Some spells needs a expire time, in case a spell never hits anything or a time buff that you want to destroy after any time. This method can be used to destroy the instantiated object after the expireTime;

MobaDealDmgDestinationObj()
Used to deal damage to the gameobject destinationGameObj.

MobaDealTriggerObj(Gameobject obj)
Used to deal damage to a given game object.

Player Spawn

The component SpwanScript.cs attached to the GameObject SpawnManager_mn is responsible for spawn all the players in the game.

public void SpawnRedTeamPlayer() and public void SpawnBlueTeamPlayer() are called by the PickSelection script to spawn characters.

The spawn position is set by redSpawnPoints and the blueSpawnPoints transform.position, which are find by the GameOject tag SpawnRed and SpawnBlue respectively.

If you want to spawn characters in another location you only need to find the GameObjects named SpawnRed and SpawnBlue in the hierarchy and move to the location that you want to spawn your character.