SCROLLING SHOOTER

EBAL STUDIOS

Contents

Introduction

Shmup Boss™© is a Unity engine shoot ‘em up game genre template. It takes inspiration from classic and modern shmups and scrolling shooter games. Shmup Boss implements a wide array of features in a user friendly manner and includes many different demo scenes that represent a variety of games you can start off with.

Shmup Boss will help you build your own scrolling shooter without the need for coding by simply modifying the demo scenes, or building your own levels after following the tutorials.

If you are an experienced programmer, Shmup Boss will offer you a very strong foundation for you to start with if you want to further customize your shmup game and add features to it. All the major challenges of pooling/spawning/events/health/weapons/collisions/effects/damage/movement/etc... have been implemented in a clear, documented, organized and easy to understand code for you to expand upon.

That being said, please note that this pack is not the be all and end all of Shmups, and does not contain every feature possible by a scrolling shooter, features which some deem essential for a shmup genre might not be included. There is an infinite number of potential features and no pack can ever aspire to make them, but we tried our best in advertising what is included and what to expect before making a purchase.

From the Author

I won’t trouble you much with the story of why and how you are able to see this pack on the asset store, but suffice to say, Shmup Boss has actually cost me an untold effort to learn and make considering that I come from a 3D art background. It was not my original intention to make a template of this kind since financially it is not the most prudent choice. But I have completed the second version because it is something that I have started and wanted to prove to myself that I can finish and not quit halfway through.

This pack is not perfect, but great care has been taken with every component and naming every method and class. If you have experience in building projects of this magnitude then you probably know how challenging it is to get everything working in order and the days on end that will go in debugging and getting it all to work. I tried my best when making the current version of this pack and I sincerely hope that you find it useful and that it would help you in making your own games one way or another.

Features

Core

- Mobile and Desktop support.
- Vertical and horizontal scrolling shooter.
- Finite and infinite level creator.
- Comprehensive pooling system.
- Advanced background layering system.
- Behaviours and effects based on events
- Global difficulty system.
- Main menu with settings.
- In level UI.
- Audio & sound effects system.

Agents (Player, enemies and bosses)

- Relatively easy to use enemy wave spawning system.
- Drops, score and pick up system.
- Player upgrades (visual, weapons, speed, health and shield).
- Different types of enemy movers (simple mover, curve mover, waypoint mover and AI mover).
- Enemy mines.
- Multi phased boss system.

Weapons

- Bullet munition weapons.
- Missile munition weapons.
- Particle weapons.
- Ability to shoot radially.
- Timed rate control.
- Fire and hit fx.

Additional

- Camera tracking and shake effects.
- Color flash effects and health and shield bars.
- Ready to use 2D and 3D art libraries.
- Animated sprite effects and explosions libraries.
- 2D sprites change based on movement direction.
- Royalty free soundtracks and SFX to use in your commercial game.

Updates

Addressing the Issue of Features and Updates Requests:

We quite understand that there are more features you wish to have been implemented. You have certain expectations on what a shmup game should include, you cannot believe that Shmup Boss have left out some of those features and it would make everything so much easier if we update this pack and add these few essentials that you request, the exact way you need them and preferably as soon as possible!

Just to put things in perspective from the point of view of an asset store publisher which will perhaps help you when communicating with any creator for support. Please keep in mind that:

- **Adding features to a template is a lot more work than to a game:**
  In a template you have to account for all possible variants of any feature, making your own game on the other hand requires simply doing what you need and nothing more, you do not need to think of the dozen ways your users will interpret that feature and you can make hack tricks to finish all the features used by your game. Not to mention that in an asset store template you also require things like support, documentation and tutorials for any added feature.

- **Publishers can never make all the features requested by everyone**
  A lot of publishers are creating those assets and templates on their own, are not part of a team and have a limited capacity to the amount of work they can make, there is an infinite number of possible features! One can spend a lifetime and not do everything the way everyone wants.

- **Asset store templates are commercial products**
  Unity asset store sales and income are far less than what most users perceive, which means Unity asset store publishers like myself have less resources to create and update their products. While a successful game can potentially sell hundreds of thousands of copies or even millions. On the other hand, if a Unity asset store pack sells 20 copies a month it is considered a hit!

  A Unity asset store publisher simply cannot put all resources and working time on a product that barely makes 100$ a month. The larger the number of sales, the more the publisher will be incentivised to update any pack and put more effort into it. Sometimes an asset is successful and the publisher can invest more in it, while in other cases it might not meet the predicted projections and developers are forced to take on other projects.

  Making updates and improvements is dependent on the number of sales made, amount of positive feedback/reviews and what projects are currently pursued by the developer and are not to be taken for granted as an entitlement. ***If you like this pack and would like to see it updated more frequently, the best thing you can actually do is to write a positive review***, this means that we could potentially make more income and be incentivised to further develop Shmup Boss.

Finding an asset on the store which is close to what you need means that you probably feel fortunate that there is something that can speed up the creation of your game. At the very least it can save a few days of work even if you just pick up a few ideas from it which will be worth the amount paid in terms of hourly wage. ***You should only be buying a pack that already has all the features which you feel are absolutely essential for you.***

Making or not making updates and which ones is completely up to the publisher. What we offer is this pack in its current form, all existing features are advertised and you are completely free to make an educated decision.

The Unity asset store is a small community, many packs you see posted on the asset store have been published not because it was worthwhile financially to create them specifically for the asset store, but has been published because their creators have already created them for their own games or development and they thought it would be a good idea to share them with the community since they have already finished them.

I say this because in some instances, I and all Unity asset store developers receive feedback that seem to somehow imagine that we are making huge amounts of income from our Unity packs and have large teams to handle support or constant updates but are just not willing to help them out which is never the case.

The vast majority of asset store creators are doing the best they can with the means they have. I just hope you think of me and of any other Unity asset store creator as your colleague. Who knows; the gaming community in general is relatively small and you might even one day work with one of the asset store developers!

Releases

V 2.0

In version 2.0, we have rewritten this pack from scratch and we have even rebranded the name from Shmup Baby™Ⓒ to Shmup Boss™Ⓒ. The first version had the serious issue of not using any pooling system, which meant patching it up was very difficult, in addition to that, some scripts really needed reorganizing and restructuring. We feel that version 2.0 has a better foundation to build upon and expand if the opportunity ever arises.

Overview of changes in version 2.0
Going into the details of every change will take many pages and it is easier if you simply treat this pack as a completely revamped one and view the new tutorials and examine the new demos. Below are just some of the major changes or new features to look for.

- Code has been rewritten to be more organized, easier to debug and expand.
- Added pooling systems for: backgrounds, background objects, agents, effects, pickups, munition and sound effects.
- Added a boss system.
- Game object based munition weapons in addition to an improved particle weapon system.
- Additional options for the scrolling background, how enemies are spawned and a new wave system.
- Added limited ability to spawn ground enemies for the finite level controller.
- Global difficulty multiplier.
- Camera zoom with tracking and camera shake.
- A more generic player select menu included in the main menu.

On the other hand, some small features have been dropped when upgrading to version 2 such as:

- Ability to shoot missiles, missiles are categorized as bullets now and cannot be fired at.
- I removed the ads manager, I have actually an updated script for it which I might share later on GitHub but I didn't want to support any problems that might come up with Unity Ads which I have no control over.
- Objects control have been morphed into the FX spawner, and other scripts. Because things are now pooled, it had to be changed.

Folders

- **Art**
  This folder contains all art source files, this includes: animations and their controllers, materials, meshes, sprites and textures for 3D meshes.

- **Audio**
  5 soundtracks and 27 sound effects that you can use in your commercial game.

- **Documentation**

- **Prefabs**
  This folder is very important to refer to while making your game, please try not to modify anything of the existing prefabs, some prefabs are used throughout multiple levels suchs as the munition and the UI, if you want to modify something, just duplicate it and place it in your own folder. If anything does not work as expected in agents, bullets, effects, etc.. you can always compare what you have built with the existing prefabs and see if there are any differences.

  The prefabs located inside the UI folder are particularly important as they are used throughout the levels. You do not need to create your own UI, simply use this one, or duplicate it and add it to your own level.

  Please note that any prefabs used inside the level must have unique names! Any prefabs with identical names used inside the same level will be treated as the same prefab and only one of them will be pooled. Always use unique names with any prefabs in the same level.

- **Resources**
  This folder is used for the audio mixer and the difficulty multiplier.
  the Audio mixer needs to remain in a folder called “Resources” and is fetched automatically when the scene plays. And the same applies for the multiplier data. Please do not rename or move this folder.

  If you have your own resources folder in your project, copy the Audio and Multiplier folders into your resources folder to avoid having 2 resources folders in the same scene.

- **Scenes**
  The demo scenes are located here.

- **Scripts**
  Holds all C# scripts used in this package. You can change these folders location. The “Editor” folders though need to stay in the same hierarchy it was placed in. For example: you could move the “Player” folder inside of “Weapons” to wherever you want to; but the “Editor” inside of “Player” needs to stay inside the “Player” folder. Each folder contains a simple readme to explain what scripts are inside of it.

Getting Started

The best way to start with Shmup Boss would be to study and use the existing demos, always keep an original copy of the pack or of anything you intend to modify. Then just start playing the demo scenes, study the different components, how the enemies or the player are laid out, the different waves used in the spawner, etc.. This should give you a general idea of the workflow in this pack and as you continue to duplicate and modify the existing prefabs and scenes, you will get more of an idea of how this pack works.

Additionally you can always highlight any variable in the inspector, many variables which could use an explanation have one in the form of a tool tip, if you hover over it, a text box will appear explaining more.

Video tutorials provided on our Youtube channel are also an excellent way to understand how things work and how to build your own game, you can check our forum thread to find out more and how to access those videos: https://forum.unity.com/threads/1030870/. Every script also has sufficient commentaries and summaries. Even if you do not know how to code, you can consider clicking on “edit script” just to read the summaries to learn more on how it works.

You can start a level by going to the edit menu, clicking on Shmup Boss, Level, Create finite level.
Then you will need to drop the level UI prefab from the prefabs UI folder. Similarly create a player from the same edit menu under agent, change its settings, add visuals, make a prefab of it and reference that prefab inside the level script.
To add backgrounds, drop a sprite into the scene, add a trigger box collider to it, make a prefab of it. Then use it inside the scrolling background layer.
For spawning enemies, from the project tab, create a scriptable object wave data of the appropriate type and assign to it the enemy you have created from the edit menu, create weapons, link them to agents. And you are done.

Now in actual application, there are many small details to pay attention to, that’s why I recommend either replicating the demos and changing them or viewing the video tutorials which can always explain things which are very challenging in the documentation.

Please note that Shmup Boss, like any other complex system, is a delicate machine. You need to pay careful attention to all the little details. Leaving certain values at zero, leaving fields empty, using agents without spawning them or not noticing if you are using a prefab or a scene object could break the levels. To avoid this, please carefully replicate the demos, carefully watch the video tutorials and read the notes in the following page. ***If you completely skip this manual, please at least make sure to read the following pages under: Important Notes!***

Important Notes

- Shmup Boss is a complete package which contains an interconnected web of components. I tried to make the code as modular as possible, but when building a complete game, there must be some connection points. Additionally, you can't have convenient features such as the weapons knowing their default shooting direction in a vertical or a horizontal level without depending on a core level component that defines it.

  When building your own level. Please note that you need all the components that are included in the demo levels. These core components are:

  - Level
  - Level UI
  - Level Audio Manager
  - Level Camera
  - Multiplier
  - Input Handler
  - Scrolling Background
  - Treadmill
  - Play Field
  - Despawning Field
  - Enemy Pool
  - FX Pool
  - Pickup Pool
  - Munition Pool Enemy
  - Munition Pool Player
  - Finite/Infinite Spawner
  - Player prefab referenced in the level component
  - Enemies prefabs with unique names referenced in waves data listed in the finite/infinite spawner.

  If you click on the edit menu → Shmup Boss → Level → Create Finite or Infinite Level Components You will be able to shortcut creating most of the needed components by the level, you will still need to drag and drop the Level UI/Level Audio manager from the prefabs/UI folder, change the various components settings, create player, enemies, waves etc..

  Comparing any level you build from scratch with the components of any demo scenes should give you an idea on how to build a level.

- **Always backup your project before downloading updates!** Even if an update seems small enough, you never know if it can break something or change your levels. Make a backup of your project and keep a copy of it when downloading and testing updates.

- A wave is created by right clicking inside the project tab, choosing create, Shmup Boss, waves and picking your wave type.

  Curve waves are only used with enemies using a curve mover, waypoint waves with a waypoint mover. A side spawnable wave can use a wider variety of movers which include: A simple mover, AI mover, magnet mover side spawnable, missile mover side spawnable. Using a curve mover enemy or a waypoint mover enemy with a side spawnable wave will cause errors.

- Please double check the values you are using or that no fields are empty.

  In many cases when you initialize an effect, munition, background object, etc.. some fields like the scale, speed, size, etc.. can start at zero. If you leave them at zero, you will get strange results and wonder why they do not show in the scene, while in fact they are only scaled to zero or something similar, I tried my best to add checks and protection but it cannot be helped that for some things you need to be careful.

- Source option: “By Game Manager” which exists inside the level Player Source, Multiplier Difficulty Source, Input Handler’s Input Method and Auto Fire is related to building a game with the main menu.

  When building a single level or testing your level, you need to always avoid setting this to “By Game Manager”, only change to “By Game Manager” before building a game with a main menu. The Game Manager option means that those settings will be set by the main menu and not by the component you have inside the level. This is explained further in the video tutorial regarding building a level.

- To start a level, you must use a player prefab in the level comp.

- Do not forget any active agents, munition or effects inside the scene before playing otherwise you will get errors.

  Because everything is pooled and spawned/despawned. Leaving active objects in the scene which are not referenced in a spawner means they skip pooling and cause issues. Please make sure you do not have any active agents or munitions in the scene. (Things are spawned from prefabs and you do not need to have a player or an enemy inside the scene unless you want to edit or preview it)

  To alleviate this issue, I have created the simple “Disposable.cs” script located in the project tools folder, and the game object called “Prefabs Editing and Testing”, if you nest any objects you are currently modifying inside the scene under the “Prefabs Editing and Testing” they should be deleted automatically when the scene starts not causing any issue.

- If you would like to see previews for things such as the FX spawner, weapon or munition effects, these prefabs must be placed inside the scene, otherwise an empty preview object will be created without you being able to see the preview.

- ***Names matter! Make sure every prefab you use in your level has a unique name***, 2 enemies with the exact name will be mistaken for the same enemy and only one will be used and pooled, spawned/despawned. Same goes for the munition or background. As a habit please use unique names for every prefab you use in a level.

- The size of the background field, which can potentially control many other fields and settings in the game is determined by the very first background in the first layer in the scrolling background. Also please make sure all backgrounds in all background layers have the exact same size. The first background size is used for all other backgrounds and layers

- The scrolling background script uses prefabs for its background layers and not sprites. Please do not try to assign sprites to the scrolling background. To assign your own sprite as a background; drag that sprite into the scene, add to it a box collider 2D, set it to trigger and make a prefab of it. Then use that prefab in the scrolling background.

- If you decide to make your own objects from scratch instead of using Shmup Boss template creation buttons from the edit menu or duplicating existing prefabs,. Please give careful attention to all the little details such as rigidbody2D settings and collider 2D settings. Missing a simple checkbox could have serious consequences.

  All rigidbody2Ds are kinematic, additionally the rigidbody2D for the player uses continuous collision detection and full kinematic contacts (enemies use discrete collision detection and does not use full kinematic contacts.)

  Colliders for the player, enemies and boss sub phases are non trigger, while colliders for munition such as bullets and missiles and colliders for pickups, backgrounds and background objects are all trigger.

- If you are building a game which uses 3D models and lighting then you will need to bake the lighting before building the scene. You do this by going to Window → Lighting → Settings → At the very bottom → Uncheck Auto Generate and click on Generate Lighting. Otherwise it's very possible that for example after you hit retry in a level you get a dark unlit scene. For more please refer to Unity's reference regarding lighting.

- Layers are assigned automatically by all the different pooling scripts. If you intend to merge this pack with your own pack and want to know or change the layers numbers used, you can go to the scripts folder, project, ProjectLayers.cs script and see the numbers used and for which layer and if you wish to change them.

- When you create a player in a vertical level, make sure its visuals are pointing upwards and enemies downwards, in a horizontal level, the player visuals should be pointing rightward, while enemies leftward. Weapons firing rotation will be set automatically so you do not need to worry about it.

  Munition should always be pointing upward regardless of level type or if used by the player or the enemy.

- Have you started the game but cannot see the player, waves, backgrounds, etc.. that you are using? Try checking your depth index for the player at the level comp, your waves depth index, backgrounds and background objects depth indices.

- Weapons starting direction is set automatically by the level type, for this reason, if you click on a prefab right after opening Unity and before opening any level you might get a non intended firing direction. Opening your level solves this.

Demo Scenes

- **Space Hero**
  This contains three levels, vertical, horizontal and infinity. Those for the most part demonstrate a good part of the features of Shmp Boss.

  The infinite level is just to demonstrate the infinite survival mode and make it clear how levels progress, while the vertical and horizontal try to employ different types of waves, enemies and effect.

  The vertical space hero level ends with a boss with multiple phases which you can explore.

- **Space Hero Main Menu**
  This uses the levels from the space hero demo, only 2 more simplified versions have been added alongside a main menu, to demonstrate how you can have levels with a main menu.

  If you notice, inside each level in this demo; the Level, Input Handler and multiplier have their options set as By Game Manager instead of By Level, By Multiplier, etc.. This is to enable the game manager in the main menu scene to control the settings instead of by the level.

  If those are not set to By Game Manager, your levels will not take any settings from the main menu.

  To use this demo you must first start the main menu scene, if you start the individual levels, without the main menu, you might get errors.

  Additionally, in vertical levels you might not see the proper black color covering the sides of forced aspect level. This is only in the editor and it does not show in build.

- **Fighter Jet**
  This one is a good example of using 2D sprites that change with direction, and an arcade experience. You might want to see how some enemies spawn an FX which is different from the usual FX (namely the Curve2 enemy which spawns a destroyed version of a jet and the tanks).

  At the end of this level you will also encounter a multi-phased boss which is different from the boss of the vertical space hero level.

- **Flying Turtle**
  This one is the simplest of our demos, no directional sprite swap modifiers or anything complex, just to show off something basic which doesn't use lots of components. But an interesting concept to take from this one would be the use of our limited ground enemies feature, you will find nested underneath the pools (specifically the enemy pool), some ground enemies, you can observe how these are spawned and used.

  At the end of the level, an enemy with an AI mover is used.

- **Neon Blaster**
  Similar to the flying turtle, this is a simple level to show off a different style of artwork. It uses transparent background layers to achieve interesting effects.

- **Underwater**
  Similar to the fighter jet demo, but doesn't use any directional sprite swap components. This one is a good example of layering 2D background layers and sprites.

- **Weapons Demo**
  This scene was used to record some of the clips in the trailer, to change the weapons the player uses, simply activate and deactivate the weapons inside the player prefab.
  The player in this demo uses the gradual sprite swap component.

Package Components

All of these components can be found through the Add Component menu, choosing Shmup Boss and selecting the appropriate component. Additionally you can just drag and drop the script you want from the scripts folder or by finding its name from the Add Component menu.
Click on a main title (in bold letters) to jump to the page containing it.

Index

Level Core

Most of Shmup Boss components' are interconnected one way or another, the following scripts are by far not the only "core" elements in this pack, but have been categorized thus as they are perhaps more important than others.

Level

The main class in any Level, most scripts are dependent on it, it determines the level type (vertical or horizontal), spawns and keeps track of the player, score, coins. Events can also be added to it when a level starts or ends.

The fact that this script stores whether a level is vertical or horizontal is very important, since movers, spawners, different fields, weapons, etc.. all need to know if a level is vertical or horizontal.

Requires:

- **A reference for the player prefab or a game manager which stores the player reference depending on the player source selected option.**

Level

- **Level Type**: Select whether the level you are building is vertical or horizontal.
- **Space Between 2D Indices**: Agents and backgrounds are spawned using a Z depth index which determines their Z position, this space determines the distance between those depth indices.

Player

- **Player Lives:** Player lives at the start of the level.
- **Player Max Lives:** The maximum number of lives a player can obtain through pickups.
- **Time To Respawn Player:** The time between the player elimination and its respawning.
- **Player Source:** This determines the prefab source for the player. In the case of building a full game where you have different players to select from, this must be from the game manager, but when testing your level or when building a game where there is no player selection from a main menu this should be from the input you use in the level.

If Source is By Level

- **Player Comp:** Will only be used if the player source is by level, this component will be used as the prefab source for the player to be spawned.

Player Spawn Position

- **Spawn X Position Ratio:** The player spawn position on the X axis in relation to the play field.
- **Spawn Y Position Ratio:** The player spawn position on the Y axis in relation to the play field.
- **Player Depth Index:** The Z depth index the player will be located in. If you forget this at zero, you will most likely not see the player when playing the game.
- **Is Drawing Player Spawn Position:** Shows a circle representing where the player will be spawned inside the play field.

Multiplier

This class is tightly involved in all aspects of Shmup Boss, it offers control for agents vitals, movers speed, weapons rate, and munition damage and speed. This will essentially give you an easy to change control setting instead of rebuilding each level with a different difficulty manually, it will also allow global difficulty changes, that being said you will also need to be careful about the data array you input and to make sure the input is correct and if using a main menu, is in accordance with the difficulty values there.

Because the multiplier is intertwined with many components, it is pivotal that it can always be accessed with its referenced multiplier data. Please note that currently the multiplier data is stored in the resources folder and is used in all of the demos, changing the multiplier data in the resources folder means that it will change for all demos.

This multiplier also controls the difficulty change in the infinite spawner through the infinite increment multiplier data.

You must be careful about changing the length of the multiplier data array, if you are using a multiplier with source “by multiplier” in a single level, then it is not a problem to change the length of the array, and selecting your own difficulty level. If you are building a game with a main menu though, if you want to change the array length, you must also change the difficulty names enum, and the settings menu script.

To create your own multiplier data, in the project tab, right click, go create, Shmup Boss and select Multiplier Data.

The multiplier component finds which multiplier data array is used and stores it in a static class called “CurrentMultiplier”, this improves the performance because all game objects can easily access the static members of “CurrentMultiplier” instead of having to constantly access a referenced multiplier component.

To test or see what multiplier data values are currently used in your level, you can add a script called “CurrentMultiplierTester.cs” located in the project, tools folder to any game object inside your scene. This will enable you to see what values are being used. And in the case of the infinite level it will help you understand how the infinite incremental values are interacting with the difficulty multiplier with every level progression.

Requires:

- **References for all the multiplier data.**
- **If you select the difficulty level source "by game manager", you will need to start the game from the main menu to select the difficulty from there.**

- **Difficulty Level Source**: Where the difficulty level is set, whether from the main menu "By Game Manager" or from the number you input below "By Multiplier". If you select "By Multiplier", then the number you input in the difficulty level will determine which item in the multiplier data array is selected, difficulty level of 0 will select the first multiplier data in the array to be used, level of 1 will use the second item of the array and so on.

If Source is By Multiplier

- **Difficulty Level**: The index at which the multiplier data array will return the indexed array difficulty. This will only be used if you have selected the difficulty level source to be: by multiplier. (As explained above, number zero will use the first member of the array, 1 the second, etc..
  There is a validate protection to prevent you from selecting below zero or above the length of the array.

Multiplier Data Array

- **Multiplier Data:** This array is essential to the functioning of this pack, it gives you the option to change the difficulty of the game. If you do not wish to have difficulties options, keep its length at one and the difficulty level at 0 and make sure the difficulty source is set by Multiplier. and fill it with the default multiplier data.

  The difficulty level determines which multiplier data is returned from the array. This difficulty level can be set by the multiplier here or by the game manager settings menu. The settings difficulty names array are currently set to 3, so the array here needs to be set to 3 as well, if you change this array length to more or less you will need to change the available difficulty names array in the SettingsMenu.cs script.

Infinite Level Multiplier Settings

- **Infinite Increment Multiplier Data**: This data is used to change the difficulty in every level progression when using the infinite spawner.

Input Handler

Selects which controls are active and how they are processed, whether auto fire is enabled or not and its related pointer options and if you are using mobile controls.

If building for mobile, this class is dependent on the LevelUI class to find the virtual joystick and buttons.

This script is usually added to the player to add movement to the camera but you can add the “Zoom FX By Mover” to multiple objects as well and not just one. For example you could add Zoom FX By Mover to a few types of enemy objects to add a dramatic effect to the game once these enemies are on screen.

Requires:

- **Virtual joystick and button referenced in the Level UI if building for mobile.**
- **Game Manager and to start the game from the main menu if “Input Method” or “Auto Fire” are By Game Manager.**

- **Input Method:** Which input method is active and controls the player, if you select the ”by game manager” option; the selection of controls vs pointer will happen in the settings menu of the main menu.
- **Auto Fire:** If auto fire is enabled, the player will not need to press fire1 to fire the weapons and they will always be shooting, if by game manager then the selection will be determined by the main menu settings.
- **Is Movement Normalized:** Makes the movement uniform regardless of anything that can cause differences or accelerating/deaccelerating input such as not moving the mobile virtual joystick all the way, or the time taken for a button down to reach its full value.

Pointer Settings

- **Pointer Offset:** When mobile touch controls are active (pointer input method), this offset will enable that the touch pointer will be a little off the player so that the user's finger is away from the player ship in the game.
- **Pointer Threshold:** This would avoid any possible jitter made from small movements of the mouse or touch finger in mobile and make it so only big movements are accounted for.

Level UI

Level UI

Contains all the UI elements needed inside a level which includes: score, lives, vitals, level complete canvas, game over canvas, etc..

Please do use the existing prefab in the prefabs UI folder. When creating a finite/infinite level template from the edit menu; This level UI (And the level Audio manager) will not be added, and will not to be added manually. Starting the level UI from scratch and attempting to populate it on your own, is a time consuming process. Using the prefab is a lot quicker.

The Level UI is needed to show not only the HUD, but also the pause menu, game over, level complete, etc..

Please note that I have a coins option. The coins are saved at every game over and when completing the level and are stored in the save file, but there is no use for them in the main menu, they have been added so that anyone who wishes to expand this pack could find they have already been completely prepared and is able to add options for purchase or special content. Making those is well beyond the scope of this pack.

Requires:

- **Level Instance. Which is essential for finding all the different values for the player, score, etc..**
- **Game manager. (You do not need to add it unless using a main menu, otherwise it will be created automatically for you)**
- **All the different texts, buttons and sliders to be referenced.**

- **Is Forced Into Camera Aspect Ratio:** If the level is vertical, built for desktop and its camera is using a vertical aspect ratio; Checking this will make the UI forced with that aspect ratio as well. Otherwise the UI will always fill the canvas..

HUD

- **Hud Canvas:** The UI canvas which holds all the heads up display such as vitals sliders, score, lives, etc..
- **Hud Pause Button:** The button which will pause the game and bring up the pause menu.
- **Player Lives:** This text will be changed into an X character with the number of lives currently available.
- **Player Score:** This text will show the current accumulated score during this level.
- **Player Coins:** This text will show the current accumulated coins during this level.
- **High Score:** This text will show the highest score accumulated previously. It will only store the high score after you have finished the level or the game is over.
- **Health Slider:**
    - **Vitals SliderType:** Always make sure the health bar is selected here.
    - **Slider To Control:** The slider which will display the current player health.
- **Shield Slider**
    - **Vital Slider Type:** Always make sure the shield bar is selected here.

- **Slider To Control:** The slider which will display the current player shield.

Pause Game

- **Pause Game Canvas:** The UI canvas which holds all the pause menu controls
- **Pause Game Resume:** The button which when clicked will hide the pause menu and continue the game which was previously paused.
- **Pause Game Main Button:** This button will exit the current level and return to the main menu. (Home button)
- **Pause Game Retry Button:** This button will restart the current level.
- **Pause Music Slider:** The slider which will control the music volume.
- **Pause Sound FX Slider**: The slider which will control the SFX volume.

Game Over

- **Game Over Canvas:** The UI Game over canvas which holds the score, retry and home buttons.
- **Game Over High Score:** The highest score achieved so far in this level.
- **Game Over Current Score:** The score currently accumulated.
- **Game Over Main Button:** This button will take you back to the home main menu.
- **Game Over Retry Button:** This button will restart the level.

Level Complete

- **Level Complete Canvas:** The canvas which will appear when the level is completed, showing the score and the home and next level buttons.
- **Level Complete High Score**: The highest score achieved so far in this level.
- **Level Complete Current Score:** The score currently accumulated.
- **Level Complete Next Level Button:** This will take you to the next level.
- **Level Complete Main Button:** This button will take you to the home main menu.
- **Level Complete Retry Button:** This button will restart the level.

Mobile Virtual Controls

- **Virtual Controls Canvas:** This canvas holds the controls used when building for mobile and are only visible when the game is built for Android or IOS.
- **Virtual Joystick:** The virtual stick which will control the player direction if the build is for mobile.
- **Virtual Fire1 Button:** The button which will cause the player to fire if auto fire is disabled, if auto fire is enabled this button will be hidden.

Joystick

A class which finds the Vector2 direction of a virtual stick image when dragged in relation to a background image.

This is used with the joystick controller for building with mobile and which is referenced inside the Level UI, inside the Level UI prefab, under the virtual controls canvas, you can see how the joystick is implemented.

Please note that when testing for mobile inside the editor and using the virtual controls, those controls will not be active, they will only be active inside your build when the application is running inside your mobile device and not inside the editor.

- **Stick Limit:** The limit of how much a stick can move in relation to its background image.
- **Stick:** The rect transform that is connected to the UI canvas image which represents the joystick which the user will move.
- **Minimum Threshold:** The minimum radius at which the movement of the joystick will affect a change in the movement of direction.
  For example, a value of 0 means the slightest movement of joystick will cause the player to move while a value of 0.5 means that you will need to move the joystick more than half of the background radius for the player to start moving.

Hold Button

This will enable a virtual image button to find if it is still pressed when it's held down. This is only currently used with the main fire button and you can find its implementation inside the Level UI, Virtual Controls Canvas, Fire1 Button.

- **Pressed Image**: the sprite image that will replace the current sprite when the button is pressed.

Level Audio Manager

Gives the ability to control the music and sfx via a master audio mixer and mixer groups, and it also enables selecting different audio sound effects for the different level UI events.

Any sound effect that will be played inside the level will have to go through this audio manager, the background music should also be applied here.

Please note that when creating a level template through the edit menu, Shmup Boss, Level, Create finite/infinite level template. This will not be added! It is included with the level UI prefab which you will need to drop manually from the prefabs UI folder.

- **Background Music:** The background music that will be looped and controlled by the music volume in the settings.
- **UI Audio Clips**: An array of all possible audio clips that can be played by any level UI. Please be careful not to give two audio clips the same UI event.

Camera

Level Camera

Controls the camera size and offsets it to align it with the background or input, it also controls the aspect and zoom values and holds the viewfield rect.

This camera script is interconnected with other scripts, it calculates the space between indices for the level, finds the viewfield, which might be dependent on the background and this view field could also be sued by the different fields in the game.

Requires:

- **Camera**
- **Level**
- **Scrolling background if the view field source option is by background dimensions.**
- **Camera Player Tracker if you wish to use the camera shake from the agents.**

Viewfield Source

- **Viewfield Source:** If the viewfield which could be used by the play field or the despawning field is calculated by the background or by input.

If Source By Input

- **Input Field Size:** Only used if you have selected by input in the viewfield source, this input will control the camera viewfield.

Camera Zoom

- **Zoom Factor:** Use values over 1 to zoom in, if you want to have a camera shake effect, or camera tracking, you must have some zoom value over 1.

If Desktop Build & Vertical Level

- **Vertical Aspect Ratio:** This aspect ratio will be forced on the level if you are building a game for desktop and the level is vertical, otherwise this aspect ratio will not be used. In mobile or a horizontal level the camera fills the level automatically.

Gizmos

- **Is Drawing Input Field:** Will draw the input field, please note that the input field will only be used if the view field source is by input, if for example you select viewfield source by background dimensions and do not input any input field there will be nothing to see.

Camera Player Tracker

The component is basically responsible for tracking and following the player, if you wish to see any results the camera must be zoomed in by the zoom factor in the level camera script.

This script is also responsible for shaking the camera. Shaking the camera requires that the camera is zoomed in, that the shake settings are set properly and that your agents such as the player or enemy have a camera shake FX.

When the camera is zoomed in, the camera player tracker script will move the camera to follow the player and it will also activate the camera shake if an agent calls for it.

Requires:

- **Camera**
- **Level Camera**
- **Level**
- **Player referenced in the level**
- **Camera shake FX applied to agents if you wish to use a camera shake effect.**

Camera Tracking Player Settings

- **Tracking Transition Time:** This time determines how long it will take for the camera to follow the player into its new position. A shorter time means it will follow it instantly, a longer time means it will lag behind and move slowly towards the player. This will also define how long it takes for the camera to get back to focusing on the player after shaking, it is better to avoid larger values and to stick to values less than 1.

Shake Settings

- **Shake Duration:** How long does the camera shake last.
- **Shake Distance:** When the camera shakes, it will move from side to side. The shake distance is how much the camera will move when it shakes.
  Please note that this distance is capped by the actual camera zoom. If the camera is zoomed in only slightly, then there will be no distance for it to shake. You need to be careful when assigning the shak distance, not to assign a value which is too big which would give you unpredictable results, use values in accordance with your camera movement zone.
- **Shake Speed:** How fast the camera moves when shaking.

Game Fields

Shmup Boss uses multiple fields for a number of reasons, there are fields that despawn any enemies or munitions that go out of the game, another to know where to spawn enemies, another to know where to spawn ground enemies.

The classes have been made quite generic to enable adding more classes easily if anyone wanted to expand upon this pack and add his/her own fields.

All of those fields require the following:

- **Level Camera if view field source is the view field**
- **Scrolling Background if background dimensions is selected, additionally it may require the scrolling background if the camera view field is dependent on the background.**

And they all contain the same fields, the only difference is in the function of each field.
These are their fields:

- **Field Source:** How is the field calculated, and what is the original source that will be used as a field.
- **Offset:** Adds a general offset for the gamefield, this might be needed if you want to give extra margins, so that for example things are not spawned/despawned exactly at the edge.
-

  This can make the field bigger or smaller, with a unified value across all edges, unlike the specific offsets below which only make the field smaller but are easier to control for a specific edge.

  If you wish to make only one edge larger, a trick would be to make a general offset value which will increase all the edges, then make all other 3 edges smaller via the offset values below.

- **Offset Top:** Determines the margin from the top for the field.
- **Offset Bottom:** Determines the margin from the bottom for the field.
- **Offset Right:** Determines the margin from the right for the field.
- **Offset Left:** Determines the margin from the left for the field.

If Source By Input

- **Input Field:** This is only used if you have selected Input as the field source.

Gizmos Settings

- **Is Drawing Field:** Will show how the field looks like, please note that for a field to be shown, it needs to have what it requires. For example, if one field is getting its dimensions by background, yet your scrolling background script has no background in it to calculate the dimensions, the field will not be shown, additionally if you choose by input, yet do not input any values, there will be nothing to show.

Play Field

This field determines where agents (player and enemies) and effects are spawned and the movement bounds of the player and the AI mover. Without the player will not be able to move and spawner will not know where to spawn enemies.

Despawning Field

This is needed so that any munition or enemies going out of view can be despawned, it has been separated from the other fields because you might want to offset it so that things do not despawn exactly at the edge. It also makes it easier to debug things if it is a separate script

Ground Enemies Spawning Field

Just like the despawning field, this has also been separated, to give further control, additionally your game might not need it as you might not have any ground enemies. This is why it is deactivated by default when creating a template.

Please note the ground enemies are limited, they can only be used with the finite spawner, they do not end a level and have limited movement options.

Particle Destruction Field

You may or may not need this one, even if you are using particle weapons. If you do not use any particle weapons then you most certainly do not need this field. If you do, it depends. If you use the distance field in the particle weapons, then those particles will be despawned through time and you will not need a particle destruction field.

It is recommended that you do not use this field, this has only been added as an extra option. You will need to use it only, if you keep the distance option for the particle weapons at 0, it will then destroy any particle that goes out of this field.

But please note that this field will activate any particle hit fx you have, so if you plan to use it, you must skip using particle hit fx, or just make the field large and do not add sound for the particle hit fx.

Pools

One of the main characteristics of Shmup Boss, is that everything is pooled. It means during the game, things are not actively instantiated and destroyed unless the existing pooled objects are not enough, in that case the pool will expand to accommodate for the required objects.

All spawned backgrounds, background objects, enemies, pickups, munition, effects and the player are pooled at the very start of the level, and only activated/spawned when needed. Once they are eliminated or despawned they go back to their respective pool until they are requested again.

There are different pooling scripts to account for the different types to be pooled, but all of them with the exception of backgrounds and background objects, inherit from “PoolBase.cs” and are quite easy to edit or restructure. No culling options have been added but it should be quite easy if someone knows how to code to add them if needed to the pool base script.

In the case of the scrolling background, only the minimum amount of backgrounds are pooled, this is because the script will go ahead and check how many are needed, how many will be visible during the game and pool backgrounds accordingly. In other pools you will need to specify how many are needed either through the pools such as in the Enemy and FX pools or through the object being pooled such as munition or pickups since it will be impossible to predict how many are needed during the game and only an estimate can be made.

If the pooled amount is less than what will be required, the pool will expand as needed. If it is more, you will just be using slightly more memory than needed. To see how many objects are being used in relation to your pools, you can always play the level in the editor then check in your hierarchy tab, the pooled lists of objects and see the objects being spawned/despawned (activated/deactivated).

A very important thing to consider for pools is that naming must be unique. In the same level, all enemy prefabs for example must have unique names, if two enemies have the exact same name, then only one of them will be pooled and the enemy pool script will not recognize that they are different. This goes for pickups, munition and effects.

This occurs because I have taken the approach of indexing different objects through their names. If you prefer to take a different approach and know how to code, the pool base script and all scripts derived from it should be quite easy to edit and modify.

Below I will very quickly explain on the specific pools, Scrolling background and background objects spawner will be explained in their respective sections.

Enemy Pool

This component is not only needed for pooling the enemies, other pools are dependent on it. The list of enemies it finds is also used by the other pooling scripts to find what pickups, munitions or effects to pool.It will find what enemies to pool by going through the spawner (finite or infinite), and also through its hierarchy.

As explained in the previous section, all enemy prefabs used must have unique names otherwise they will be treated as the same enemy.

A limited ground enemies feature is implemented through this pool, if you would like to have ground enemies in a finite level, simply nest those ground enemies underneath the enemy pool object. They will automatically be pooled and spawned when they enter the ground enemies spawning field.

Note: If you duplicate ground enemies, Unity will add a (number) to the name. For example: You have “EnemyBig” nested underneath the enemy pool, when you duplicate it multiple times, the newly duplicated objects will have names: “EnemyBig (1)”, “EnemyBig (2)”, “EnemyBig (3)”, etc.. All those objects will be accounted as the same object and the enemy pool script will disregard the added (number) to the name.

To understand how the ground enemies work, you can see the flying turtle demo, where I have used two types of moles as ground enemies and you will find them nested underneath the pools object which contains the enemy pool.

Requires:

- **Finite or Infinite spawner to find which enemies (adversaries) to pool.**
- **If you plan to use the limited feature of ground enemies, your ground enemies must be nested directly underneath it. (please completely avoid nesting any other type of object underneath it)**

- **Pool Limit:** The enemy pool script will go through the waves to find how many times an object is used and register that number, the pool limit will cap this number. For example it might be found that a certain enemy will be spawned a 100 times throughout the level. But you know that at every single instance, no more than 10 will be used. So you can use the pool limit which will limit how many enemies are pooled to 10.

FX Pool

There are a number of effects used throughout this pack, this includes effects used by the FX spawner (such as explosions), effects used by firing the weapons and effects used when munition or particles hit a target.

The FX Pool will go through the list of enemies and the player, find any effects they use, find what weapons they use, and what munition is used with those weapons and all the associated effects with them and pool them. Just like all other pooling in this pack, things are categorized by name. ***Different effects must have different names or otherwise they will be treated as if they were the same effect.***

The FX pool will looks all effect found in the following scripts:

- FX Spawner Player
- FX Spawner Enemy
- Weapon Fire FX
- Munition Hit FX
- Particle Hit FX

Requires:

- **Level**
- **Player prefab reference in the level (If Main menu is used, the game manager player will be used)**
- **Enemy Pool**

- **Pool Limit:** How many clones are pooled of each effect.

Pickup Pool

This will go inside all the drops of all enemies and find what pickups they use and pool accordingly. ***All pickup prefabs names must be unique otherwise they will be treated as the same pickup.*** How many pickups are pooled is determined in the pickup pool limit component.

Requires:

- **Enemy Pool**

Munition Pool Player

This will look up the player used by the level, find its listed munition weapons and pool any munition used by them. The number of munition pooled depends on the pool limit set in the munition prefabs in their respective bullet or missile components.

Munition with identical names will be treated as the same munition, please use unique names for each munition prefab.

Requires

- **Level**
- **Player prefab reference in the level (If Main menu is used, the game manager player will be used)**

Muniotion Pool Enemy

This will look up all the enemies listed by the enemy pool, find their listed munition weapons and pool any munition used by them. The number of munition pooled depends on the pool limit set in the munition prefabs in their respective bullet or missile components.

Munition with identical names will be treated as the same munition, please use unique names for each munition prefab.

Requires

- **Enemy Pool**

Background

An important characteristic in a scrolling shooter, is the actual scrolling background! The following components control how the backgrounds scroll. The background scripts and dimensions will help determine the different fields sizes.

Scrolling Background

A pivotal component, the dimensions of the very first background prefab in the first layer of the scrolling background could potentially determine the view field of the camera, the despawning field and all other fields.

Please note that backgrounds must be prefabs and not sprites! This means that if you have a background sprite. You must first drag it to the scene, add to it a trigger box collider, make a prefab out of it then use it inside the scrolling background script. This has been done this way to make the script more generic for potential 3D objects use and not just sprites.

All background layers must use the same dimensions. For example, if you use a background with size of 1 x 2 units as the first background of the first layer. Then all other background objects and layers must use the exact same dimensions, otherwise you will have gaps. We recommend that you start off with a sprite for example using dimensions 1024x2048 or 2048x1024 and sticking to it throughout your layers.

The speed of the first background layer will determine the speed of the treadmill which might be used by other components.

The scrolling background script is self integrated and pools its own background by itself, but requires the treadmill component to move.

Additionally, there has been applied some validate protection to make sure the numbers you input produce sufficient background to be viewed.

Requires:

- **Treadmill (Must be added to the same scrolling background object, and is needed move the backgrounds)**
- **Level (To know if level is vertical or horizontal and act accordingly)**

- **Active Backgrounds Limit:** How many backgrounds are visible simultaneously when the game is played, the value you should input is dependent on your viewfield and background dimensions. If still confused what this value does, just play the scene then change the number and play again.
- **Backgrounds Layers:** The background layers that make up the level, please make sure that your main layer is the first layer. This first layer speed will determine the speed of ground units and its size will also determine the background field which depending on your settings could determine your playfield, viewfield and other fields. ***Please note that you cannot use two backgrounds which have the same name or otherwise they will be considered as the same one.***
    - **Layer Name:** This layer name is only used for organizational purposes in the editor to make it easy to know which layer is which.
    - **Backgrounds:** The background used in this layer with their repetition values. Background names used here must be unique in order to be pooled and spawned/despawned.
        - **Prefab:** The background prefab that will be used, ***this must have a trigger 2D collider.***
        - **Count:** How many times will this background be repeated.
    - **Speed:** How fast will this layer move
    - **After Sequence Ends**: What happens when the backgrounds prefab and their repetition values have run its course.
        - **Loop Sequence:** Will loop the entire sequence infinitely.
        - **Loop Find Image:** Will loop the final image infinitely
        - **Stop At Final Image:** Will change the layer speed to zero when final background prefab has been reached.
        - **Finish Sequence:** Will stop spawning backgrounds when the sequence ends
    - **Depth Index:** The Z depth layer index where a background layer will be spawned, an index of 0 means the bottom most layer, higher numbers means the layers will go on top, the distance between these layers is determined by the space between indices.
- **Preview:** Will generate a quick preview of how the backgrounds will look like, this will only spawn the sequences only once, without any repetition values.
- **Delete Preview:** Will delete the above generated preview.

Background Objects Spawner

In addition to your background layers, you might want to add some objects that are just to add complexity and variability to your background, they are not enemies or anything that can be fired at. This component pools and spawns objects that will be used as background decorations with many parameters to randomize how they are spawned and moved.

Note: The background objects spawner actually inherits from PoolBase which is shared by all the other pools, but have been categorized under the background section for easier mental connections.

Requires:

- **Playfield (To know where to spawn objects)**
- **Level (to find the space between indices which is currently calculated by the camera)**

- **Random Objects Layer:** The layers which will contain all the different background objects which will be spawned.
    - **Layer Name:** This layer name is only used for organizational purposes in the editor to make it easy to know which layer is which.

      Prefabs To Spawn
    - **Prefabs:** Game objects that will be spawned as background objects in this layer. Please make sure they have a trigger collider 2D.
    - **Is Spawning In Random Order:** If checked, this will spawn the prefabs randomly, otherwise they will be spawned by their order in the array.
    - **Is Spawning Infinitely:** If true, this layer will continue to spawn objects non stop during the level and the spawn count will not be considered.
    - **Spawn Count:** This number is only used if IsSpawningInfinitely is unchecked, spawn count would limit the number of spawned objects to this count.
    - **Pool Limit:** How many are initially pooled.

      Spawn Time Settings
    - **Wait Time Before First Spawn**
    - **Wait Time Between Spawns:** The time in-between spawning each background object in this layer. This is only used if IsRandomWaitTime is unchecked.
    - **Is Random Wait Time:** If checked, the layer uses the min/max random time instead of the wait time between spawns.
    - **Min Random Time:** Only used if IsRandomWaitTime is checked, represents the minimum possible time between spawning background objects in this layer.
    - **Max Random Time:** Only used if IsRandomWaitTime is checked, represents the maximum possible time between spawning background objects in this layer.

      Scale Settings
    - **Scale:** Only used if “Is Random Scale” is unchecked, this will modify the scale of the spawned objects. ***Please note that if you forget this value at zero you will not see any background objects and properly wonder what is happening.***

- Is Random Scale: If true, the layer uses the min/max random scale values instead of the scale field.
- **Min Random Scale:** Used if IsRandomScale is checked and sets the minimum random scale for the spawned objects.
- **Max Random Scale**: Used if IsRandomScale is checked and sets the maximum random scale for the spawned objects.

Spawn Position Settings

- **Depth Index:** This depth index will determine the position of the layer on the Z axis. (forgetting this value at zero might mean that you will not be able to see any spawned objects.)
- **Spawn Side**: Which side are objects spawned from, their direction of movement will be selected automatically based on their spawn side.
- **Spawn Side Offset:** How far from the edge of the playfield are the background objects being spawned.
- **Spawn Position:** The ratio of where on the spawn side will the object be spawned. For example a value of 0.5 means that the object will be spawned from the center of the spawn side while a value of 0 means it will be spawned at the first corner and 1 on the opposing corner.
- **Is Random Spawn Position:** If checked, the spawn position will be discarded and the min and max random spawn position values will be used to pick the position of spawning on the spawn side.
- **Min Random Spawn Position:** Only used if the isRandomSpawnPosition is checked.
- **Max Random Spawn Position:** Only used if the isRandomSpawnPosition is checked.

Speed Settings

- **Speed:** The speed of the spawned background object which will have its direction set automatically depending on the spawn side. Note: this value is only used if IsRandomSpeed is unchecked.
- **Is Random Speed:** If checked, the speed value will be discarded and the min and random speeds used instead.
- **Min Random Speed:** Only used if the IsRandomSpeed is checked.
- **Max Random Speed:** Only used if the IsRandomSpeed is checked.

Rotation Settings

- **X Rotation Speed:** The rotation speed of the spawned random background object on the X axis, this is used only if IsRandomRotationSpeed is unchecked.
- **Y Rotation Speed:** The rotation speed of the spawned random background object on the Y axis, this is used only if IsRandomRotationSpeed is unchecked.
- **Z Rotation Speed:** The rotation speed of the spawned random background object on the Z axis, this is used only if IsRandomRotationSpeed is unchecked.
- **Is Random Rotation Speed:** If set to true, the min and max random rotation speeds will be used instead of the rotation speeds on the axis.
- **Min Random Rotation Speed:** Only used if the IsRandomRotationSpeed is checked.
- **Max Random Rotation Speed:** Only used if the IsRandomRotationSpeed is checked.

Treadmill

Moves all the background layers in the scrolling background component. In addition to that it is also responsible for moving the ground units and the movement of spawned background objects are dependent on the speed multiplier in this script.

The multiplier gives you the option to later expand on the script and perhaps write your own controllers so that you can make the treadmill speed increase or decrease, for example to possibly decrease it when a boss comes to add some sort of a dramatic effect to your Shmup.

Requires:

- **Level (To know if level is vertical or horizontal)**
- **Scrolling Background (so it can access it and move the backgrounds, The treadmill component must be assigned to the same game object of the scrolling background)**

- **Speed Multiplier:** This multiplier will change the speed of anything that is dependent on the treadmill. This includes all of the scrolling background layers, the spawned background objects, and ground units.

Spawners and Waves

Enemies in Shmup Boss are spawned through a dedicated spawner script which uses scriptable object data to know what to spawn. Level completion is dependent on the spawned enemies and that all waves have finished spawning.

There are two types of spawners, the finite spawner which is used with your standard level which is completed and then another scene can be opened which represents another level (a standard level), and the infinite spawner which is used in an infinite survival mode game.

Please note that there is a limited ground enemies spawning option which occurs through the enemy pool component and not through those spawners, it has its own limitation and is not to be mixed with the finite or infinite spawners. Additionally levels completion is not related to ground enemies (Technically a level will not end unless all enemies are destroyed but only after the final wave ahs been spawned)

The waves which are to be used with the spawners are scriptable objects which can be created from the project tab, by right clicking, then choosing create, Shmup Boss, Waves and then the appropriate type of wave.

Enemies must be assigned to wave type based on the movers they have. For example enemies with a curve mover must only be assigned to curve waves. More will be explained in the section for each wave type.

Finite Spawner

Use this with any standard level you are making (A level which ends and is not infinite). The enemy pool script will look up all the waves listed in this component to know which enemies to spawn. Please be careful not to keep any waves empty or not to leave any waves without their enemies referenced.

Waves are spawned by time difference, regardless whether or not a wave has finished spawning, the level ends after the final wave has finished spawning and all active enemies have been eliminated.

Requires:

- **Level**
- **Enemy Pool**

- **Waves Start Time:** The delay time for spawning the first enemy wave.
- **Waves:** All the wave data that are going to be spawned and the time between each one.
    - **Wave Data:** The wave data that will be spawned. This can be either a side spawnable wave, a curve wave, or a waypoint wave and are created by right clicking inside the project tab, choosing create, Shmup Boss and waves.
    - **Time For Next Wave:** The wait time for spawning the next wave.

Infinite Spawner

The infinite spawner is used when you want to build an infinite level, a survival mode game where the level never ends. This spawner has sublevels within it, showing you progress each time a sublevel ends. It also has the important feature of increasing the difficulty as you progress with each level which is strongly connected to the multiplier component infinite increment multiplier data.

It uses a streams concept so that you can control when to spawn waves infinitely, the stream concept might be somewhat a challenge to explain in writing although it is quite simple in operation, for this reason I have included a print level streams button to help you better understand which waves will be included in which sublevels. Just use your streams with start, end and every nth level value. Hit print level streams and read in the console to understand how it works.

Requires:

- **Level**
- **Enemy Pool**
- **Multiplier**

- **Waves Start Time:** The delay time for spawning the first enemy wave.
- **Streams:** The streams that hold all the waves that will be spawned and defines in which sub levels they will be spawned in.
    - **Waves:** The waves which will be spawned during this stream.
        - **Wave Data:** The wave data that will be spawned. This can be either a side spawnable wave, a curve wave, or a waypoint wave and are created by right clicking inside the project tab, choosing create, Shmup Boss and waves
        - **Time For Next Wave:** The wait time for spawning the next wave.
    - **Increased Amount:** The number of enemies that will be increased after each time this stream has spawned, this does not need to be 1 or bigger and fractions can be used.
    - **Start Level:** An infinite level is made up of an infinite number of sublevels. The start level, determines at what sub level a stream wave will start.
    - **End Level:** An infinite level is made up of an infinite number of sub levels. The end level, determines at what sub level a stream wave will start.
    - **Every Nth Level:** In addition to the start and end level, the every nth level will determine the frequency of spawning this stream. A frequency of 1 means the stream will spawn in every level between the start and end levels, a frequency of 2 means it will spawn once, skip the next level, then spawn again, and so on.

Level Number Declaration

- **Level Declare:** The text which will be replaced with the word Level followed by the current sub level number.
- **Level Declare Delay:** The time between finishing the sub level in an infinite spawner game and showing the level declaration text.
- **Level Declare Stay:** For how long will the level declaration text stay active.

Console Print Level Streams

- **Level Limit:** how many sub levels to print which streams will be included in them.
- **Print Level Streams:** Will show what stream will be included in what sub levels based on the start,end and every nth level values of your streams.

Waves Data

A scriptable object data that is created from the project tab by right clicking, create, Shmup Boss and selecting waves. This data will store the enemies to spawn and their spawn parameters and are used inside the finite and infinite spawners.

All wave data share the same following fields:

Spawn Settings

- **Number of Enemies to Spawn:** How many enemies are spawned by this wave.
- **Is Spawning Enemies in Random Order:** If true, this will spawn the enemy prefabs randomly. If false it will spawn the enemy prefabs in an ordered sequence.
- **Time Between Enemies:** The time between spawning each enemy in the wave, this is only active if isRandomTime is set to false.
- **Is Random Time Between Enemies:** If checked, time between enemies will be discarded and the min and max time below will be used to define a random time between spawning each enemy.
- **Min Time Between Enemies:** Only active if isRandomTime is checked, will determine the shortest time possible between spawning enemies.
- **Max Time Between Enemies:** Only active if isRandomTime is checked, will determine the largest time possible between spawning enemies.

Spawn Position Settings

- **Depth Index:** The Z depth layer index where an enemy will be spawned.
  An index of 0 means the bottom most layer, higher numbers means the layers will go on top, the distance between these layers is determined by the space between indices.

Prefabs to Spawn

- **Enemy Prefabs:** The prefab/ prefabs which will be spawned by the wave data. In a side spawnable wave data it must be an enemy with a mover type that is side spawnable, in a curve wave data it must be one with a curve mover and in a waypoint wave must be one with a waypoint mover.

Side Spawnable Wave Data

Wave data which applies to all enemies with movers that spawn from the side, this includes all types of movers with the exception of the waypoint mover and the curve mover. This will use the play field boundaries to know where to drop the enemies when first spawned, and it will also use the spawn side to know the rotation of the spawned enemies.

This wave data can be applied with all side spawnable mover, you cannot use this wave data for enemies which have a curve mover nor ones which have a waypoint mover.

Enemies with the following movers can be used a side spawnable wave:

- **Simple Mover**
- **AI Mover**
- **Missile Mover Side Spawnable**
- **Magnet Mover Side Spawnable (typically used for mines)**

In addition to the common wave data fields, a side spawnable wave has the following:

- **Spawn Side:** The side from which adversaries will be spawned from.
- **Spawn Side Offset:** This will offset the spawn position of the adversaries. The bigger the value the further away from the side the adversary will be spawned.
- **Spawn Position:** This will only be used if isRandomSpawnPosition is unchecked, it will determine the location on the spawn side, i.e. on the center or around the corners.
- **Is Random Spawn Position:** If this is checked it will discard the value of the spawn position and use instead the min and max values to determine the spawn position ratio on the edge.
- **Min Random Spawn Position:** This will only be used if isRandomSpawnPosition is checked.
- **Max Random Spawn Position:** This will only be used if isRandomSpawnPosition is checked.

Curve Wave Data & Waypoint Wave Data

Both the curve wave data and the waypoint wave data use the exact same fields, the only reason they have been separated was to place protections for using only a curve mover enemy prefab with the curve wave and only a waypoint mover enemy prefab with the waypoint wave.

The starting position for enemies spawned from these waves will be the first point of the curve or the first waypoint.

Agents

Speaking in coding terms, currently the only agents in Shmup Boss are the ones inheriting from the Agent class, this would only include: Player, Enemy, Enemy Detonator and Boss Sub Phase. But for documentation and ease of use terms I would like to add to this documentation category the boss and pickups to ease the mental connections.

Agents all have controls for their vitals (health & shield), invincibility, weapons, collision settings. The player and enemy components differ in additional fields which will be explained in their respective sections.

Inside these agents' classes scripts they have controls for the specific events that can occur to them, such as being hit, spawning elimination etc.. Additionally they have controls for taking hits from munition.

Their vitals are strongly connected to the multiplier values and are affected by them.
This pack uses 2 types of weapons, weapon munition and weapon particle, each has its own advantages, disadvantages, thus why we added two types of weapons in this system. For a weapon to fire it must be listed in the agent and the agent game object must have the appropriate shooter (player or enemy shooter).

To save yourself and to make sure the agents you create have all the right settings, you create the agent through the edit menu, Shmup Boss, agents and pick the template you would like to create. You would still of course have to add the visuals and any event to activate the sounds of explosions, make weapons etc..

Requires:

- **Kinematic Rigidbody 2D**
- **2D Collider (non trigger)**
- **Multiplier**
- **Munition Pools (To know what to despawn when hit by a munition)**

These common fields are shared amongst agents:

Vitals

- **Health:** Full health of the agent at the start of the level.
- **Shield:** Full shield of the agent at the start of the level.

Invincibility Settings

- **Invincibility Time After Spawn:** For how long does the agent remain invincible after it has been spawned.
- **Is Pass Through When Invincible:** When set to true it will make bullets and missiles pass through the agent when the agent is invincible. When false, bullets and missiles will hit the agent but without causing damage.

Weapons

- **Munition Weapons:** any munition weapons used by the agent need to be listed in this array. Additionally for a weapon to fire it also needs the respective shooter (player or enemy shooter)
- **Particle Weapons:** any particle weapons used by the agent need to be listed in this array. Additionally for a weapon to fire it also needs the respective shooter (player or enemy shooter)

Advanced Settings

- **Is Triggering Munition FX When Hit:** When set to false, any munition (bullets or missiles) hitting this agent will not spawn a hit fx. This will give you a slight performance improvement since every munition hitting this agent undergoes a get component operation to see if it is spawning a hit fx or not.

Collision Settings

- **Collision Damage:** The damage caused to the other agent which collides with this agent.
- **Can Take Collisions:** Whether or not this agent can be damaged by collisions.
- **Collision Cool Down Time:** The time after taking a collision in which the agent will be immune from further collisions.

Player

The agent script addition to the base agents, is the upgrade options. Inside the script there is also a list of events which relate to all the different possibilities a player can encounter. The agent event system is one of the strong points of Shmup Boss.

In addition to the requirements of the agent, the player Rigidbody2D needs to use full kinematic contacts and collision detection should be continuous.

Vitals Upgrade Limits

- **Max Health:** The max health a player can achieve after taking health upgrades.
- **Max Shield:** The max shield a player can achieve after taking health upgrades.
-

Weapons Upgrade Settings

- **Weapons Upgrade Stage:** The start weapons upgrade stage for the player in the level.
- **Max Weapons Upgrade Stage:** The max weapons upgrade stage for the player after picking up upgrades.
-

Visual Upgrade Settings

- **Visual Upgrade Stage:** The start visual upgrade stage for the player in the level.
- **Max Visual Upgrade Stage:** The max visual upgrade stage for the player after picking up upgrades.

Enemy

In addition to the base agents' fields, the enemy also has control for drops (pickups), elimination rewards and elimination by time.

The rigidbody2D for the enemy only needs to be kinematic, all other settings can be set at default.

Requires (In addition to agent's requirements):

- **Level**

Drop Settings

- **Drop Radius:** The radius in which dropped items will possibly be spawned in. This will be drawn as a yellow circle in the scene.
- **Drops:** Array of pickup items with their possibility of being dropped.
    - **Pickup Prefab:** This prefab should have the pickup script added to it, additionally you may want to add a magnet maver and a player tracker to it, or you can either use an already prepared pickup prefab or create one from the template.
    - **Drop Chance:** The percentile chance of a drop, a value of 100 means making the drop when the enemy is eliminated is certain while a value of 50 means there is a 50% chance.

Elimination Rewards

- **Score:** Score gained when eliminating this enemy.
- **Coins:** Coins gained when eliminating this enemy. Please note while this score is fully implemented and is saved in the save file after game over or a level is complete, I have not implemented any options or templates for it, I have added this coins so that someone who knows how to code can easily expand on it, I have prepared all the hard foundation for that sort of expansions but doing it is well beyond the scope of this pack.

Disable Settings

- **Disable After Time:** Despawns the enemy after this time has passed.
- **Is Eliminated When Disabled:** This gives elimination rewards and drops items when the enemy is disabled after time.

Enemy Detonator

This adds the option for the enemy to detonate in a proximity mine like manner. If the player comes within this agent' detonation start radius; it will explode after the set time, damaging the player if it's within its explosion radius.

All of these components options are exactly like the enemy but with the added controls related to the detonation.

A tracker player (more on tracker's later) is added by default to the game object with this component, which will make the detonator start detonating when the player is within the detonation radius, but if you can possibly any type of tracker to it.

Requires (In addition to enemy's requirements)

- **Tracker**

Detonation Settings

- **Explosion Damage:** The damage dealt to the player if it's within the explosion radius at the time of the explosion.
- **Explosion Radius:** The radius in which the player will be damaged in if it's located within at the time of the explosion.
- **Detonation Start Radius:** If the player enters this radius, then the detonation process will start and after the timeToDetonate the explosion will occur.
- **Time to Detonate:** The time between the detonation start and the explosion.

Boss Sub Phase

This is only used as part of a boss and must be nested under a boss hierarchy and it must reference the boss. This class adds the option of showing a destroyed version and most importantly will be deactivated instead of despawned from the enemy pool, since the boss is the one that will be spawned/despawned from the pool and not its sub phases. It also has the important function of notifying the boss when it's eliminated.

To use this, you must first create a game object and add the boss script to it, you can then create another object which is nested underneath the boss object. It of course requires all other components needed by any enemy.

To understand the process or how a boss works, you can see in the space hero main vertical level, the boss at the end, which is located inside the demos enemies prefabs, or the boss at the end of the fighter jet demo.

You can also create a quick setup for a boss from the edit menu, Shmup Boss, Agents and then select create boss template. The template created is just bare bones, and needs to be referenced. You will need to add the sub phase into phases/sub phases array, and to link the boss component to it. This is better explained in the video tutorials.

Requires (In addition to the enemy's requirements):

- **Boss component referenced in it.**
- **And for the sub phase to be referenced in the boss script.**

Boss Sub Phase Settings

- **Boss:** The boss this sub phase is related to, this boss will activate/deactivate this sub phase. Without this reference the boss and this script won't work.
- **Destroyed Version:** This game object will be set active when this sub phase is eliminated to be a representation of a destroyed version of this sub phase.

Agents Related

Boss

The boss component is made up of multiple phases and each of these phases are made up of its own sub phases. This component can simulate a multi phased boss, with phases being activated according to the current phase index with options of having them visible/active, damageable and shooting from the start or not.

You can see how a boss is set up from the boss prefabs used in the space hero vertical demo and the fighter jet demo, additionally a bare bone boss template can be created from the edit menu which still needs referencing and making setting up.

The boss object is the one which will be pooled in its entirety and not its sub phases which will not be pooled, or spawned/despawned.

I have added a validate checker for the options of the boss sub phases, to help understand how it works. For example, in the first phase you will not be able to uncheck any of its options Because it's the first phase! The first phase is always active, for the later phases, for the enemy to shoot or be damaged from the start it must be active from the start as well. So if you want to deactivate the is active from start if the other 2 are active you must first deactivate them as well.

- **Destroy Only When All Phases Destroyed:** You have the option to either destroy the enemy when the final phase is destroyed, or when all phases are destroyed. For example, if you have the final phase active from start, and damageable from start, then it is possible that it will be destroyed before the other ones are.. In that case if this isn't checked. The boss will be destroyed even though some phases are still intact.
- **Boss Phases:** The phases that make up the boss. These phases are made up of sub phases which need to have the boss sub phase script added to it and not just a standard enemy script.
    - **Boss Sub Phases:** The boss sub phases that make up this phase. You can have multiple sub phases that are all part of the same phase, all of those sub phases will need to be eliminated if the phase to be considered eliminated and the boss to progress to the next phase.
    - **Is Game Object Active From Start:** When checked, this will make this phase active from the start even if it's not the first phase. But while this phase will be visible it will not fire or be damaged unless the parameters below are checked. (If you have checked all 3 options, to uncheck this you will to uncheck the two below first)
    - **CanBeDamagedFromStart:** This will make it possible to damage this phase from the start.
    - **Is Shooting From Start:** This will make this phase start shooting from the start.

Pickup

Pickup items generate an effect in the player when it collides with them. Pickups are used inside an enemy drop, they are pooled, spawned/despawned by a specialized pool (Pickup Pool) and you can choose from a number of pickup effects.

Although you can assign a multiple number of pickup effects to the same pickup, please pay careful attention to the order of effects in the array, for example if you place a heal first in the array then an upgrade shield, then the newly added shield capacity will not be filled. You must first upgrade the shield then heal to make it complete. I have chosen this option because it is the most logical one and gives you added control on how to control your pickup effects.

Requires:

- **Level**
- **Kinematic rigidbody2D**
- **Trigger collider 2D**
- **Tracker Player**
- **Magnet Mover**

- **Pool Limit:** How many clones of this item will be pooled. Try to use a number close to the maximum you believe will ever be used on screen.
- **Pickup Effects:** When adding multiple pickup effects, please pay careful attention to the order of your effects. For example always place a health upgrade BEFORE a health fill or heal and not the other way round.
    - **Pickup Type:** Heal would heal both the health and the shields. All other options are self explanatory.
    - **Amount:** In numbers, an amount of 5 will increase the health for example 5 points.

Agent FX

One of the strong points of this pack, is that its agents make use of events, such as “on spawn”, “on hit”, “on elimination” etc.. All of the following components, add their actions to these events and are activated when a specific agent event occurs.

Following the same workflow of these classes, you can feel free to add your own custom made classes and actions to this established and convenient system.

For most of those events you will find a player version and an enemy version. This has been made because the player has events which are not shared with the enemies and vice versa. This can keep things separate and not allow for any mix up. Because the only difference between those is the events selection, I have explained them in the same sections.

All of those events can be added, by going into the “Add Component” menu from the inspector, choosing Shmup Boss, then FX.

SFX Player & Enemy

Components for playing sound effects when the player or enemy events occur. In any game sound effects are essential. The SFX player and enemy scripts provide an easy way to add sound effects to any possible agent event.

Just add a SFX Player component to the game object holding the player component, or the SFX Enemy component to the game object holding an enemy (or enemy detonator or a boss sub phase) component. Choose how many sound effects you want to have, pick your audio clips, agent event and volume and you are done.

Note: It’s quite common to forget to change the volume of the audio clip and leave it at zero, which means you will not hear any sound. Please always make sure your clip volume is not at zero.

Requires:

- **An agent component**

- **Sound Effects:** The list of audio clips which are triggered by an agent event and have volume controls (volumetric AC).
    - **Volumetric AC:** The audio clip with volume controls which will be played when a selected agent event occurs. (A volumetric AC is a fancy term for an audio clip with a volume control)
    - **Agent (Player or Enemy) Event Trigger:** The agent event that will trigger the clip to play. This will be either a player or an enemy event depending on the component you are adding.

FX Spawner Player & Enemy

In any scrolling shooter, you will need to have things like an explosion when the player or an enemy is eliminated, maybe some effects to appear when they are hit, or a special effect to spawn when the player picks up something.

The “FX spawner player” and “FX spawner enemy” spawn effects when a player or an enemy event occurs, those effects are spawned by the FX pool and are only eliminated through an FX Eliminator which will be explained later in this section.

You can add the FX spawner for the player from the Inspector, add component menu, pick Shmup Boss, FX, player, then FX Spawner Player and similarly for the FX Spawner Enemy. After that you will need to place the FX game object you want to spawn when an event occurs, pick your event and options and you should be ready to go.

Important note: Any FX game object prefabs referenced in the FX GO must have an FX Eliminator, or there will be no way for it to be despawned from the FX pool.

At first glance, it appears that the FX spawner (Player/Enemy) has a lot of options, but those mostly relate to the start position option you pick, and only a portion is used depending on the options you pick. You will most likely use the “By Agent” option and settings and disregard the “By Field” and “By Input” settings and options unless you have a specific feature you want to implement.

Caution: You can only see the preview if the prefab is dropped inside the scene and not through the prefab viewing window.

Both the “FX Spawner Player” and “FX Spawner Enemy” share the exact same fields with the only exception of having different events according to their agent types.

Requires:

- **Agent (Player for the “FX Spawner Player, Enemy for the FX Spawner Enemy).**

- **Fxs:**
  - **Fx Go:** The game object that will be spawned from the FX pool when an agent event occurs. This game object must have an FX Eliminator on it so that it can be despawned.
  - **Hide From Preview:** Hides this effect in the scene when the preview button is pressed. This has been added so that when you view multiple effects they do not overlap.
  - **Modify Scale:** Changes the scale of the spawned FX.
  - **Start Position Option:** This will define how the position of the spawned object is found. The option you choose here defines which settings below are used, Available options are:
    - **By Agent:** Will spawn the effect at the location of the agent which has triggered, if you pick this option; the active settings will be the agent settings which will allow you to offset the position.
    - **By Play Field:** Will spawn the effect in the playfield regardless of the agent position according to the “by play field” settings

- **By Input:** Will spawn the effect at the input position you set in the “by input settings” option.

By Agent Settings

- **Agent Position Offset:** The position offset of the spawned object. This will be used only if the start position option is by agent.

By Play Field Settings

- **Spawn X Position:** Determines on the X axis where the spawned object will be located in relation to the play field. This will be used only if the start position option is by play field.
- **Spawn Y Position:** Determines on the Y axis where the spawned object will be located in relation to the play field. This will be used only if the start position option is by play field.
- **Depth Index:** Determines the Z depth (position) of the spawned object. This will be used only if the start position option is by play field.

By Input Settings

- **Input Position:** The position of the spawned object. This will be used only if the start position option is by input.

Spawn Trigger

- **(Agent) Event:** The player or enemy event that will cause the spawning of the FX

- **Preview:** Shows a preview inside the scene of how the effects will appear, this preview has its limitations, it will not show any preview in the prefab window, you will need to have dropped the prefab inside the scene and hit the preview button from inside the dropped game object there. Secondly, if you have picked the by play field option, to view the effect, you must have a play field inside the scene and to have at least played the scene once to have stored an instance of it.
- **Delete Preview:** Deletes the preview you have just created.

Reminder: Please make sure your spawned FX has an FX Eliminator/Animated FX Eliminator added to it.

Flash FX Player & Enemy

You might want to change the color of an agent, when a certain event occurs such as taking a hit. The “Flash FX Player” and “Flash FX Enemy” change the sprite or material of a renderer to the color you set when the selected event occurs.

Flash FX allows for multiple renderers which you must reference manually. So if you have an enemy that is made up of multiple sprites or multiple renderers you should reference all of them, additionally if you have a player with multiple visual upgrade phases, all of them must be referenced.

All though the flash FX scripts allow for multiple flashing colors on different events, it only accounts for the last event that occurred, for example if you have a flash fx event that occurs when the player takes a hit changing the color to red, and another event to change the color to blue when it picks up a shield. If the player picks up a shield after it has taken a hit, only the blue color will be shown and the hit red color will be disregarded.

Requires:

- **Agent (Player or Enemy)**
- **Renderer**

- **Target Renderers:** Renderers which materials will change when an event triggers them to. If no renderers are referenced, the flash fx will attempt to get the renderer component of the game object it is attached to. This works with sprite renderers and mesh material renderers.
- **Flash Effects:** flash effects that are triggered by a player or enemy event for a set time
    - **Color:** The color the flash effect will change the material/sprite to. Pay careful attention to the alpha value.
    - **Flash Time:** For how long this flash will last.
    - **Agent (Player Or Enemy) Event:** The agent event that will trigger the flash, this is set by the player or enemy and is translated from player or enemy events into agent events.

Camera Shake Player & Enemy

If you want to shake the camera you must first make sure that the level camera with a zoom value greater than one has the Camera Player Tracker applied to it.

Without the level camera zoomed in, and the camera player tracker added to it, with the proper settings applied, no agent event can shake the camera.

After making sure your camera is already set up; to shake it, add a “Camera Shake Player” to the player or a “Camera Shake Enemy” to an enemy, pick the event that you wish for the camera to shake at and you are done.

For example if you want the camera to shake every time the player is hit, first make sure the camera has the “Camera Player Tracker” component added to it. Then add to your player prefab a “Camera Shake Player” component and pick the “On Hit” event.

Another example, if you want the camera to shake when any enemy is eliminated, you must (after making sure the camera has a zoom value and a camera player tracker script) add the “Camera Shake enemy to every enemy prefab that will be spawned and pick the “On Elimination” event.

This setup gives you better control, this way you could possibly choose for the camera to shake only when a certain type of enemies is destroyed, or maybe when they are spawned, or when a player is eliminated, etc…

Both the “Camera Shake Player” and “Camera Shake Enemy” are exactly the same. The only difference is that they have different events.

Requires:

- **Agent (Player or Enemy)**
- **Level Camera (With a zoom value greater than one)**
- **Camera Player Tracker**

- **Agent (Player or Enemy) Event:** The agent event that will cause the leve camera to shake when it occurs.

Vitals Slider

Controls Unity UI sliders representing the vitals (health or shield) of an agent (player or enemy). The same script can be applied to both the player, the enemy or any of its derived classes. It needs any agent which has vitals and it attaches its update sliders method to the agent's taking health and shield damage events.

Just drag and drop the AgentVitals prefab from the Prefabs UI folder, and see how it is set up (A simple Unity sliders canvas), add the vitals slider component to the agent and reference the UI. The player in the space hero demo has a vitals slider component added to it which you can see how it functions.

Please note that this component is not related to the general UI slider and can be used with any agent. The main player slider is located inside the Level UI (See the segment on the Level UI)

Requires:

- **Agent (Player, Enemy, Enemy Detonator, Boss Sub Phase)**
- **Unity sliders canvas.**

- **Vitals Sliders:** The health or shield sliders which will change according to the change in the agent vitals. (You can add multiple sliders of any kind)
    - **Vitals Sliders Type:** Whether this slider is controlled by the agent health or shield.
    - **Slider To Control:** The slider that will show the health or shield changes that occur to the agent.

Visual Upgrade

When a visual upgrade drop item is picked up by the player, this component will update the player visual. You must first create multiple visuals within the player prefab, add a visuals upgrade component, then reference those visuals to the Upgrade Visuals array.

The visuals upgrade script is a simple script which activates the game object in the upgrade visuals array according to its array index and current visuals upgrade level.

Note: You should also set the number of "max visual upgrade stage" in the player script in accordance to your visual upgrade stages.

To see an example of how it is setup please see the Space Hero demos players,

Requires:

- **Player**
- **To pickup a visuals upgrade (A visuals upgrade pickup effect is separate from the weapons upgrade pickup effect)**

- **Upgrade Visuals:** An array of all the possible visual upgrades for the player. Each array element represents an upgrade stage, element 0 will display the first stage, element 1 will display the second stage and so on.

FX Eliminators

Any effect spawned by FX spawner (player or enemy), weapon fire FX, munition hit FX or particle hit FX requires an FX eliminator to be added to it, so that it can be despawned, otherwise it will simply stay active or in the scene after it has been spawned.

All effects are pooled by the FX Pool and spawned/despawned from it. As just mentioned now, the effects which are the responsibility of the FX pool and need an FX eliminator to be added to them are those spawned by:

- FX Spawner Player
- FX Spawner Enemy
- Weapon Fire FX
- Munition Hit FX
- Particle Hit FX

You will find many examples of those effects stored in the Prefabs, VFX, Explosions and Weapon FX folders, additionally some are located inside the Prefabs, Demos, Fighter Jet and Flying Turtle FX folders.

The FX eliminator's job is to simply put the effect back into the FX Pool after a certain time. The animated FX eliminator, is just the same as its base FX eliminator, with the only difference that it can automatically find the length time for the animation clip without having to input it manually.

The added elimination option of Destroy instead of despawn has been added just in case you have instantiated an effect instead of spawning it from the FX pool.

FX Eliminator

Eliminates an FX by either despawning it to the FXPool or destroying it depending on how it was created.

Requires:

- **To have been spawned by the (FX spawner player or enemy, weapon fire FX, munition hit FX or particle hit FX)**

- **Elimination Option:** After the assigned time is expired, do you want to despawn an FX to the FXPool or do you wish to destroy it? This option is important, if an FX was spawned, then you must select the despawn option, if it was instantiated then you must select the destroy option, destroying a spawned FX will make the FXPool empty and attempting to despawn an instantiated FX won't work. Generally speaking since FX instantiation is not widely implemented, please keep this at the despawn option.
- **Wait Time:** The time until eliminating an FX. (The time the effect is visible after being spawned)
- **Is Parented To Treadmill:** If this is checked, then FX will move with the background, giving it the feeling of a grounded effect.

Animated FX Eliminator

The “animated FX eliminator” is exactly the same as the FX eliminator with the only difference that it has an added option for finding out the length of an animated clip automatically.

Use the animated FX eliminator with animated clips instead of the FX Eliminator.

Like the FX eliminator it also requires to be spawned by the FX pool, **additionally it requires:**

- **Animator**
- **Renderer**

It has the same options as the above FX Eliminator but contains these extra options:

Animated Clip Settings

- **Is Using Wait Time Instead of Clip Length:** If this is true, the animated clip time will be discarded and the wait time will be used instead.
- **Start Delay Time:** Delays playing the animated clip by this time value.

Weapons

Shmup Boss uses two types of weapons, munition weapons which are pooled and spawned/despawned by the munitio pool player/enemy and particle weapons. Particle weapons are highly performant but lack the added control you get with game object bullets/missiles in munition weapons.

Having two systems of weapons takes additional time to get used to, but you can just use whatever system you prefer, we noticed that some users prefer the performance gains of particle systems while others need to have the added control and versatility of munition weapons.

Any weapon you create will need to be referenced in its agent according to its type, moreover, you will need to have a shooter component with the agent, without these two prerequisites your weapons won't fire.

If you combine those systems with simple components such as the "weapon rate controller", "focus rotator" or simple rotator you will be able to achieve many types of potential weapon patterns.

Additionally, If you know how to code, you can also easily expand upon the bullet and missile systems and upgrade them by adding controls for changing munition speed or rate of movement and upgrade the system into a very complex bullet hell system of weapons.

There are quite a number of options for the rotation of the weapons, some are included in the shooters while others are in the weapons themselves, this should give you all the control you need to control the rotation of the weapons.

Shooters

For the weapon to fire, you must add shooters to your agents, if you create the player or enemies from the edit menu, Shmup Boss, create agents templates; shooters will be added automatically. These shooter scripts have been separated from the agent classes to make modifying and customizing the agent classes a lot easier and to make the scripts more modular.

The “Shooter Player” and “Shooter Enemy” will access the agent scripts, and fire the weapon listed taking into consideration in the case of the “Shooter Enemy” The movement direction.

If your agent does not contain any weapons you do not need to add any shooter components.

Requires:

- **Agent (Player or Enemy)**

“Player Shooter” has no fields.

Note: The separation of the munition weapon settings from the particle weapons settings in the “Enemy Shooter” could give you the added advantage of starting the weapons at different times without having to add a weapon rate controller.

Enemy Shooter Fields:

Munition Weapons Settings

- **First Munition Shot Delay:** Delay time before firing munition weapons.
- **Is Munition Following Movement:** When checked and the munition weapon settings use the weapon rotation, munition will be fired with the rotation of the enemy.

Particle Weapons Settings

- **First Particle Shot Delay:** Delay time before firing particle weapons.
- **Is Particle Following Movement:** When checked, particles will be fired with the rotation of the enemy.

Particle Weapons

Particle systems are highly performant and do not need any pooling, on the other hand they have their own limitations since each bullet is a particle and not a separate game object, you are free to use whatever weapons system you feel fulfils the need of your development.

The settings of the particle weapon are very similar to the ones in the munition weapons but the interface is slightly different and there are few more options.

You can create particle weapons from the edit menu, Shmup Boss, weapon and create particle weapons. Or by creating an empty game object and adding to it a particle system and particle weapon player or enemy.

Any weapons you create will need to be referenced inside the agent, additionally the agent will need to have a shooter component.

Sometimes after inputting the settings for the particle weapon you will not be able to see the results of your input, or the weapon will not appear to fire in the preview, simply select any other object then the weapon again and it should be reinitialized.

To get the correct particle firing direction in the prefab preview, you must be either inside or have already opened the type of level (vertical or horizontal) this weapon is used in, for example if you just open Unity, then open a particle weapon prefab and preview it, the weapon firing direction will not know if the level is vertical or horizontal and will assume vertical direction. But if you open your scene it will be able toa access the level vertical or horizontal static variable.

The particle weapons have been the only item in this package that have used a comprehensive editor interface, it makes editing the weapons easier, but I have not used this technique elsewhere in the script because this makes the package much harder to debug or to add your own features and expansion, but I have left this inspector approach for the particle to give an idea on editor scripts for anyone who want to learn from the pack.

Bullets used with this particle weapon system are simple materials, you can view those materials in the art folder, materials, bullets folder. Mobile alpha blended materials can show some barely visible edges but are the most performant, you are free to create any material type you want, additionally if you want the pick bullet button to function, you must have a folder name bullets containing the materials for the particle weapons.

A special feature with this type of weapons is the overtime, in munition weapons no overtime is needed. When a weapon is destroyed or disabled, its fired munition are not affected, while in particle weapons if the weapon firing this weapon is destroyed, its particle will instantly disappear with it, this is a problem because it means that when an enemy using a particle weapon for example is despawned, its bullets will disappear. This has been solved by the overtime field you will see below which de-attaches the particle system after creating a displacement.

Particle bullets are either destroyed by distance on their own, or if you do not wish to input any distance field you can use the particle destruction field. But please note that if you use a particle hit FX component, if a particle bullet is despawned by the particle destruction field it will spawn an effect and you should be better off using the distance option in the particle weapon.

Requires:

- **Particle System**
- **Multiplier**
- **Agent to be referenced in (player or enemy)**
- **An agent shooter (player shooter or enemy shooter)**

Stage (The player weapon will have an add and remove stage buttons which represent the different weapon upgrade stages the player can pick up.)

- **Pick Bullet:** Will show the materials saved in the bullets folder, to add materials to this list, you can create new materials and place them in the same bullets folder, if you want to rename the folder "bullets" you must modify the "WeaponParticleInspectorTools.cs" script. Alternatively you can just pick any material from the box selection on the right side.
- **SFX:** The sound effect clip that will be played when a bullet is fired.
- **SFX Volume:** The volume of the SFX clip
- **SFX Cool Down Time:** This time would force a wait period between making firing sound effects, this is usually needed when you have a weapon with a very high rate of fire which would make the sound effects overlap and become incomprehensible.
- **Rate:** Rate of munition fired per second, higher values means a faster firing rate.
- **Damage:** The damage each particle will deal to the target it hits.
- **Size:** The size of the fired particle sprite. This only determines the visual size and is not related to any collisions.
- **Collider Size:** The collider size of the particle (The one responsible for actually colliding with the target). You can typically use values which are half the size of the particle, a white circle will show the size of the collider when previewing the bullets).
- **Speed:** How fast the particles will travel
- **Rotation Speed:** How fast will the particle bullet rotate around itself.
- **Bullet Number:** Used to fire munition radially, when used please add a value to the arc angle spread or otherwise the munition will be fired on top of each other.
- **Arc Angle Spread:** Determines how spread out the radial bullets are if the bullets number is bigger than 1.
- **Arc Radius:** Offsets the fired radial particles away from the weapon center by the radius value.
- **Distance:** The distance particles will travel before being eliminated, a distance of 0 means they will continue traveling infinitely and you would need to use a particle destroyed to eliminate them instead.
- **Overtime:** After the agent with the weapon which uses this particle stage is despawned, the weapon needs to stay active for a time. If the weapon is deactivated immediately; all of its particles will disappear instantly, this time determines for how long will the particle weapon stay active after its agent has been despawned.
- **Is Following Rotation:** Would make the particle follow the rotation of the weapon which is firing them otherwise they will be unaffected.

- **Curve:** The curve that will hold how the particles move over time, can be used to change particles movement from a straight line into a curved line. This will only produce predictable results if the bullets number is no more than one (i.e. if you make a radial firing pattern it will not be applied to the pattern but to the system as a whole)
- **Curve Range:** How wide the curve is.

Trail: If a trail is added, every fired particle will have a trail to it.

- **Pick Trail:** Will pick a trail material from the folder “trails”, if you would like to add more trails to this list, you can add more materials to the trails folder, located in the art, materials folder. Alternatively you can just pick any material from the box on the right.
- **Remove Trail:** You must use this button to remove a trail material from a particle weapon, otherwise the trail will not be removed.
- **Trail Time:** The time this trail stays active during and will determine how long will this trail be.
- **Trail Width:** How wide the sprite forming this trail.

Munition Weapons

Spawns pooled game objects which can represent bullets, missiles or any type of projectile. Those munitions are pooled by the munition pool player and the munition pool enemy depending on the type of weapon.

Both the player and enemy weapons have the exact same fields, but have been differentiated to account for the different firing directions of the player and the enemies.

Those weapons use bullets or missile prefabs ***(munition refer to both bullets and missiles)*** which need to be premade and stored in the prefabs with unique names and referenced inside the munition weapon. More on bullet and missile in the munition section, but basically they can be created from the edit menu, and will need to be saved as prefabs with unique names. The Same bullet prefab can be used with enemy and player munition weapons, but you need different missiles for the player and enemy weapons. Player missiles are different from enemy missiles because they have different trackers. (Or you can just use the existing munition prefab or duplicate and edit them.

You can create munition weapons from the edit menu, Shmup Boss, weapons and create the type of munition weapon you wish to make. Additionally, you can simply create an empty game object and add to it the weapon munition player or weapon munition enemy scripts and you will get the same result.

Any weapons you create will need to be referenced inside the agent, additionally the agent will need to have a shooter component.

Munition weapons preview button has its limitations:
1- The preview button will only show the result if the prefab has been dropped into the scene.
2- It will not also show any of the advanced rotations options or the rotation of the weapon.
3- It will not show the effect of the scatter angle.
4- The preview is only a static display of how the munition would look like after the weapon has fired for the amount of time you specify in the preview duration.

Basically, the preview button is for you to estimate the correct firing rate and size of the munition.

Trivia: The term munition has been picked because it encompasses projectiles like bullets, and self propelling missiles. Ammunition on the other hand only describes things that do not direct itself like shells.

Requires:
- **Multiplier**
- **Agent to be referenced in (player or enemy)**
- **An agent shooter (player shooter or enemy shooter)**

The difference between player munition weapon fields and enemy munition weapon fields is that the player weapon has multiple upgrade stages while the enemy has only one stage.

Advanced Rotation Options: You most likely won't need to modify any of the advanced rotation options, they have been added to help out in the case of rare occasions when you want more control over your weapons rotations.

- **Is Using Nested Rotation:** When checked, the weapon will use the rotation which is the result of the hierarchy. Unchecked, it will instead use the weapon local rotation regardless of any rotation in the hierarchy. By default, it is unchecked, so that any agent rotation does not affect the weapons.
- **Is Using Input Rotation:** This can be used if you wish to use a single game object to rotate multiple weapons on the same agent without nesting them under the same parent.
- **Input Rotation:** Only used if isUsingInputRotation is checked, the transform you reference here will function as a rotator for this weapon instead of this game object local rotation or nested rotation.
- **Is Using Override Direction:** Use if you do not want to be concerned with any agent or local weapon rotation. Please make sure you do not accidentally check this while keeping the override direction at zero, which means your munition won't have any direction and could stay in place.
- **Override Direction:** Will only be used if isUsingOverrideDirection is checked, this vector will override any other agent or weapon rotation.

- **Stage (In the player there is an array of stages to accommodate for upgrades)**
  - **Rate:** Rate of munition fired per second, higher values means a faster firing rate.
  - **Munition Prefab:** The munition (bullet or missile) which will be fired by the weapon using this stage.
  - **Munition Scale:** Modifies the scale of the fired munition. Please make sure it is not set to zero.
  - **Scatter Angle:** Forces the munition to be fired in a random radial manner, the bigger the angle the more spread out the munition will be.

    Radial Firing
  - **Bullets Number:** Used to fire munition radially, when used please add a value to the arc angle spread or otherwise the munition will be fired on top of each other.
  - **Arc Angle Spread:** Determines how spread out the radial bullets are if the bullets number is bigger than 1.

    Sound
  - **Firing SFX:** The sound effect made when a weapon fires a munition.
    - **Ac:** The audio clip which will be played.
    - **Volume:** The volume at which the audio clip will be played.
  - **Firing SFX Cool Down Time:** This time would force a wait period between making firing sound effects, this is usually needed when you have a weapon with a very high rate of fire which would make the sound effects overlap and become incomprehensible.

Preview Settings (In the player munition weapon, you can additionally pick the weapon upgrade to stage to preview)

- **Preview Duration:** The preview button will show how the munition would appear if it has been fired for this amount of time. Make sure the preview amount is sufficient if you have a low firing rate. Additionally as has been mentioned, the preview is only a static display of how the fired munition would look like which has its limitations.
- **Preview:** Would create a static preview of how the fired munition would look like. (would only work if the prefab has been dropped inside the scene and you munition prefab has been referenced)
- **Delete Preview:** delete the created preview.

Munition

Munition weapons need munition prefabs to be referenced in them to fire anything. This makes the munition weapon a separate concept from its munition.

There are 2 types of munition: bullets and missiles. Each has its different requirements and will be explained in their respective sections. Munitions are pooled by the “Munition Pool Player” and “Munition Pool Enemy”, regardless of the type of munition (bullet or missile). ***Each munition prefab must have its unique name, if 2 munitions have the same exact name, then they would be treated as the same munition and only one will be pooled and spawned/despawned***. Munitions are despawned when they hit a target or when they exit the despawning field.

If you know how to code those 2 types of munition should give you a good foundation to start off with if you want to create your own types of munition.

Both bullets and missiles will need to be pointed upwards in the prefabs so that they can have the correct starting rotation when fired. You can learn more about munition by checking the munition prefabs, in the prefabs folder, munition or by just duplicating existing munition and adjusting it.

Requires:

- **Trigger Collider 2D**
- **Multiplier**

Shared fields between the bullets and missiles:

- **Pool Limit:** How many instances of this munition are pooled, try to estimate the maximum number of munition of this type that will be on screen at the exact same time.
- **Is Independent Of Weapon Rotation:** If checked, this would make the munition rotation not influenced by any rotation of the weapon or agent firing it.
- **Damage:** The damage caused when this munition hits an opposing faction.
- **Lifetime:** For how long will this munition stay active after having been fired.
  This is particularly important in the case of missiles, which might not go out of the despawning field to be despawned, and might instead wander around for very long times in the play field unless despawned through a life time. A value of zero means this munition has no lifetime and will live infinitely, or as long as the game lasts!

Bullets

The same bullet prefab can be used as an enemy bullet or as a player bullet, since they rely on a rigidbody2D for movement which can be set to have a different direction by the munition weapon.

To create a bullet munition you can either, create one from the edit menu, Shmup Boss, Munition then create bullet munition and then add a visual to it. Or you can just create an empty game object, add to it a kinematic rigidbody2D with a trigger collider2D and bullet script. Bullets move in a straight direction and have very few controls.

Requires (In addition to the munition trigger collider 2D and multiplier):

- **Kinematic Rigidbody2D**

Fields in addition to the shared munition fields:

- **Speed:** How fast will this bullet move in a straight direction.

Missiles

Missiles follow their targets and move towards them. A missile fired from an enemy munition weapon will track the player and follow it. A missile fired from a player munition weapon will follow and track any available random enemy in the scene.

Missiles from player munition weapons currently have their limitations and do not have any smart tracking system to follow the closest enemy or the one straight ahead and instead just follow any random enemy who happens to be picked up by mere chance.

The differences between a missile following a random enemy and one following a player are merely two components: the type of tracker and mover.

A missile fired from a player munition weapon should have a “TrackerRandomEnemy” and “Player Missile Mover” (Missile Mover Following Random Enemy)

A missile fired from an enemy munition weapon should have a “TrackerPlayer” and an “Enemy Missile Mover” (Missile Mover Following Player)

To create a missile fired from a player munition weapon and which will follow a random enemy:
1- You can create one from the edit menu, Shmup Boss, create munition then pick the type of missile you need. You will still need to add a visual to it.
2- Alternatively, you could duplicate one of the existing prefabs located in the prefabs, munition, missiles folder.
3-Or you can create an empty game object, add a trigger collider2D, a missile script, a “TrackerPlayer” script and an “Enemy Missile Mover” (Missile Mover Following Player.cs) script if you are creating an enemy missile or a “TrackerRandomEnemy” and a “Player Missile Mover (Missile Mover Following Random Enemy.cs)

Adding the “Munition Hit FX” component is completely optional, but it is only to be expected that a missile should have an explosion, but your missile can be simply a “guided bullet” and not necessarily a missile in artistic terms. You are free to add a munition hit FX or not to. If you do add a munition hit fx, make sure its Fx Go is referenced by something which uses an FX Eliminator (You can just use one of the explosion prefabs or revise the previous section on FX eliminators. (All effects are pooled by the FXPool and to be despawned they need an FX eliminator to put them back into the pool.)

Requires (In addition to the munition trigger collider 2D and multiplier):

- **Tracker Player and an Enemy Missile Mover (MissileMoverFollowingPlayer.cs) if you are building a missile fired from an enemy munition weapon.**

- **Tracker Random Enemy and a Player Missile Mover (MissileMoverFollowingRandomEnemy.cs) if you are building a missile fired from a player munition weapon.**

The missile script in itself does not have any additional fields to what is included in the base munition listed in the munition section.

The missile mover is straight forward but you can read more on it in its section.

Weapon FX

You can add effects related to the weapons, such as a firing effect, and an effect when a munition or particle hits a target. All of those effects are pooled and spawned/despawned by the FXPool, and you will need to assign to them an FX Eliminator. You can feel free to revise the section on FX Eliminators. Or just use or examine the FX explosions prefabs or weapon firing FX prefabs located in the prefabs folder, VFX, Explosions and Weapon FX.

All the different spawned weapon effects needs to have an FX Eliminator applied to their FX game object.

All the different weapon FX are based on the “Armament FX” and share common fields and requirements.

Note: The preview button will only work if the weapon is dropped inside the scene and not inside the prefab window.

Requires:

- **FX Pool**
- **For the spawned FX game object to have an FX Eliminator to it so that it can be spawned.**

Common fields:

- **Fx Go:** The FX game object that will be spawned when the required FX event occurs. (This need to have an FX Eliminator applied to it)
- **Modify Scale:** The scale change of the spawned FX.
- **Offset Rotation:** Will move the spawned FX away from its spawning position by this value.
- **New Rotation:** The new rotation of the spawned FX.

- **Preview:** Will create a preview in the scene, this preview can only be visible if you create it when the prefab is dropped inside the scene and it will not be visible inside the prefab window.
- **Delete Preview:** Delete the preview just created from the preview button.

Weapon Fire FX

This component will spawn an effect when the weapon fires, this effect can be applied to any type of weapon (munition and particle weapons).

Requires (In addition to Weapon FX requirements):

- **Weapon**

In addition to all common fields of the Weapon FX:

- **Is Fx Constrained To Weapon:** Will add a constrained point modifier to the spawned FX and link it to the weapon so that it appears to be connected to the weapon and not stay in place after being spawned.

Munition Hit FX

This applies to munition bullets and missiles and will spawn an FX when the munition hits a target.

Requires (In addition to Weapon FX requirements):

- **Munition (Bullet or Missile)**

In addition to all common fields of the Weapon FX:

- **Hit SFX:** The sound effect played when a munition hits a layer it collides with.
    - **Ac:** The audio clip that will be played when the munition hits a target.
    - **Volume:** The volume of the audio clip played.

Particle Hit FX

This will spawn an effect when a particle fired from a particle weapon hits a target.

Please note that this will also spawn an effect when the particle hits the particle destruction field, if you plan to use this effect, try to stick to the distance option in the particle weapon for destroying the bullet and avoid using the particle destruction field.

Requires (In addition to Weapon FX requirements):

- **Particle Weapon (Player or Enemy)**

In addition to all common fields of the Weapon FX:

- **Is Using Hit Angle Rotation:** When checked, the spawned FX will have an angle related to the angle which it hit the target with.
- **Angle Offset:** Only used if isUsingHitAngleRotation is checked, this will add or increment degrees to the resulting spawned FX rotation which is the result of the hit angle.
- **Hit SFX:** The sound effect played when a munition hits a layer it collides with.
    - **Ac:** The audio clip that will be played when the munition hits a target.
    - **Volume:** The volume of the audio clip played.

Weapon Components

You could feel free to add any components to your weapon and not specialized components like the weapon rate controller, these components can be the circular rotator or the focus rotator which can give a considerable amount of versatility for your weapons.

The only specialized weapon component now available to add is the weapon rate controller.

Weapon Rate Controller

A weapon modifier which will change the firing rate of the weapon based on a timer. This means that you can multiply the rate by zero for a few seconds (completely stopping the weapon from firing), then multiply it by 5 for a few more seconds, etc.. This can be very helpful especially if you are building a weapon for an enemy boss.

This can be applied to any type of weapon, particle or munition, player or enemy.

Requires:

- **Weapon**

- **Timed Rates:** These rates will be multiplied with the weapon stage original rate. Each in turn.
  - **Duration:** For how long will this stage last.
  - **Rate Multiplier:** The value the weapon's rate will be multiplied with.
- **Is Looped:** If checked, your timed rate array will execute again after its finished and repeat infinitely. If unchecked the weapon will remain at the final timed rate.

Movers

There are quite a number of movers in this pack, the vast majority of movers are meant to be used with the enemies, others are for munition and others for the pickup and player.

Player Mover

Uses input which is stored in the static “PlayerInput.cs” class to find the direction and keeps the player inside the boundaries.

Requires:

- **Player Input (which is saved by the input handler)**
- **Play Field (To know where to allow the player to move)**
- **Multiplier**

- **Speed**
- **Max Speed:** The maximum speed the player can have after speed pickups.

Enemy Movers

A number of different movers can be applied to enemies, these movers can determine what type of waves they are used in and how they are spawned. These movers are generally divided to ones who are side spawnable and those who are not.

- Side spawnable movers are: simple mover, AI mover, magnet mover side spawnabe and missile mover side spawnable.

- Non side spawnable movers are: curve mover and waypoint mover.

The directional mover is an exception as it is does not require to be spawned by a finite/infinite spawner and can be used with any object

A side spawnable mover is as the name suggests, spawned from one side of the play field (Top, bottom, etc..) and has an initial direction which will be set by the wave, a side spawnable mover spawned from the left, will be heading rightward. It cannot be headed to the left as well or you will never see it.

Non side spawnable movers on the other hand are spawned from a starting point, this point can be placed anywhere. Both the waypoint mover and curve mover have a set of points that define their movements and the first point determines where they spawn from and are not defined by a particular side.

All enemy movers require:

- **Multiplier**

Simple Mover

Moves enemies in a single direction that can either be straight, diagonal or with a curved bend.
This mover is used by the side spawnable wave and its main direction defined by the spawning side. If it is spawned from the left side; it will have a rightward direction, if spawned from the top; it will have a downward direction etc..

Requires:

- **To be spawned from a spawner while inside a side spawnable wave so that its initial direction can be set.**
- **Play field.**

- **Speed**
- **Is Following Direction:** If set to true, this mover will rotate the game object it is moving, this is applied to the Z axis.
- **Diagonal:** This value will give a diagonal rotation to the direction of movement. Please be conservative in the values you use especially if in conjunction with the bend value.
- **Bend:** Will give a curved bend effect to the direction of movement. Please be conservative in the values you use especially if in conjunction with the diagonal value.
- **Is Drawing Gizmos:** Will draw a line representing the path this mover will take (Will use the default level direction)
- **Path Length:** The length of the line representing the path the mover will take. If you keep this at zero you will be unable to see any paths.

AI Mover

Perhaps the term AI is stretching it a bit, but basically this mover will do random movements based on the position of the player and its interaction with the enemy. If the player enters the line of sight of the AI mover, the AI mover will move in a random direction with a timed delay, whenever the player enters that line of sight again, the AI mover will again move thus giving the feeling for the enemy trying to evade the player.

The AI mover Will start with an initial move based on the spawn side, afterwards it will either wait or when the player is straight ahead of it, it will perform a random direction move (maneuver move).

You need to be quite careful about the values you input with this mover and test them somewhat until you reach something you feel is satisfactory.

Requires:

- **Level (So that it can access the player mover)**
- **To be spawned from a spawner while inside a side spawnable wave**
- **Play field.**

- **Speed**
- **First Move Distance:** After being spawned, the AI mover will move from its spawning position in its initial direction to take position in the playfield. This is the distance it will move from its spawning position.
- **Maneuver Distance:** The distance it will move after it has found the target ahead.
- **Time Before Maneuver Distance:** Time after the AI mover has found the target ahead and before it makes the maneuver move.
- **Time For Next Maneuver Distance:** Wait time after the AI mover has reached it target and before it can start looking for the target ahead once again.
- **Avg Collider Width:** This width will determine when will this mover consider that its target (the player) is ahead and this in turn will determine when it will start making the maneuver move.
  You can think of this width as the inaccuracy in the line of sight. The target doesn't have to be straight ahead for the AI mover to consider its target is ahead, if it's within this width it will consider it ahead.

Offset Settings

- **Offset Top:** This will determine the movement zone of the AI mover, the top value will determine how far from the top margin can this mover come close to.
- **Offset Bottom:** This will determine the movement zone of the AI mover, the bottom value will determine how far from the bottom margin can this mover come close to.
- **Offset Left:** This will determine the movement zone of the AI mover, the left value will determine how far from the left margin can this mover come close to.
- **Offset Right:** This will determine the movement zone of the AI mover, the Right value will determine how far from the Right margin can this mover come close to.

Gizmo Settings

- **Is Drawing Gizmos:** Draws the mover average collider width

Magnet Mover Side Spawnable

This mover can be used with enemies which are typically mines, and are spawned by a side spawnable wave data in a finite/infinite spawner. It will simulate the effect of being attracted to the player when it comes closer. It is best used with an enemy detonator (mine) since it will constantly seek the player. This uses the player tracker to determine how to move.

The direction of movement of the agent when idle (target is not within magnet radius) is determined by the spawn sides (initial direction).

Requires:

- **Tracker Player**
- **To be spawned from a spawner while inside a side spawnable wave**
- **Play field.**

- **Speed:** Idle speed in the initial spawn direction.
- **Is Following Direction:** If set to true, this mover will rotate the game object it is moving, this is applied to the Z axis.
- **Magnet Speed:** The speed which the mover will use when within the target magnet radius.
- **Magnet Radius:** When the mover enters this target radius, it will move using the magnet speed and will start moving towards the target.

Missile Mover Side Spawnable

This mover can be used with enemies which are spawned by a side spawnable wave data in a finite/infinite spawner. It will simulate the effect of a missile being fired towards the player, but in this case it will have enemy vitals, fx and weapons, it is best used with an enemy detonator (mine) since it will constantly seek the player. This uses the player tracker to determine how to move.

Requires:

- **Tracker Player**
- **To be spawned from a spawner while inside a side spawnable wave**
- **Play field.**

- **Speed**
- **Is Following Direction:** If set to true, this mover will rotate the game object it is moving, this is applied to the Z axis.
- **Turn Speed:** How fast a missile will turn towards its target.

Waypoint Mover

Moves in straight lines towards pre-specified points with options to stop at the points or loop. This mover is used in a waypoint wave data (non side spawnable wave) and its first spawning position is determined by first point regardless of the play field size.

To create a waypoint for the waypoint mover, you will need to increase the waypoints array size and either manually change their position from the scene by the handles, or by inputting numbers inside their position slots in the inspector.

Requires:

- **To be spawned from a spawner while inside a waypoint wave**

- **Speed**
- **Mode:** How the mover behaves when it has reached the last point.
    - **Stop:** Stops the mover after it has reached the final waypoint.
    - **Continue Loop:** After the mover has reached the last point, the mover will continue the loop from the loop back to point waypoint.
    - **Loop Back and Forth:** After the mover has the last point, the mover will move in reverse until it reaches the loop back to point waypoint, when it does. It will go in a forward direction again, then in reverse, etc..
- **Loop Back To Point:** Any point before that will be discarded from the continue loop and loop back and forth mode options.
- **Waypoints:** The points which the mover will pass through and for how long will it stop at each one.
    - **Wait Time:** The time the mover will wait at this position.
    - **Position:** The position the mover will move to.

  Gizmos: Draws the waypoints positions and the path lines connecting them.
- **Handle Size:** The size of the circle representing the waypoint so you can move them. Please make sure this is not set to zero if you want to see the handles.

Curve Mover

Will move the object along a bezier curve path that you have created. This mover is used in a curve wave data (non side spawnable wave) and its first spawning position is determined by the first curve point regardless of the play field size.

To create a new curve point, you must hold shift and first mouse click in the scene view. You will need to adjust the handles of the curve. You can delete a point using the delete last segment button.

The curve mover despawns the enemy it is attached to when it reaches the final point in the curve path.

Requires:

- **To be spawned from a spawner while inside a curve wave.**
- **To be applied to the game object holding the enemy component.**
- **Enemy Pool as it despawns the enemy when it reaches the final point**

- **Speed**
- **Is Following Direction:** If set to true, this mover will rotate the game object it is moving, this is applied to the Z axis.

- **Handle Size:** Controls the size of the circles representing the curve control points.

Independent Movers

Directional Mover

A directional mover is very similar to a simple mover, only difference is that it does not have to be spawned from a wave, instead you can pick the initial direction.

A directional mover can be applied to anything and does not have any requirements.

- **Speed**
- **Is Following Direction:** If set to true, this mover will rotate the game object it is moving, this is applied to the Z axis.
- **Diagonal:** This value will give a diagonal rotation to the direction of movement. Please be conservative in the values you use especially if in conjunction with the bend value.
- **Bend:** Will give a curved bend effect to the direction of movement. Please be conservative in the values you use especially if in conjunction with the diagonal value.
- **Is Drawing Gizmos:** Will draw a line representing the path this mover will take (Will use the default level direction)
- **Path Length:** The length of the line representing the path the mover will take. If you keep this at zero you will be unable to see any paths.

Pickup Mover

A pickup does not necessarily need a mover, but a most common use would be to make the pickup get attracted to the player as if it were a magnet, the magnet mover is the most suitable candidate for this behaviour.

Magnet Mover

Mover which will move towards whatever target is set in the tracker in an incremental magnet like fashion. It will basically be attracted to the target and move towards it at a constant speed.

Requires:

- **Tracker (If you are using it as pickup mover, this should be the player tracker)**

- **Speed:** Idle speed in the initial spawn direction.
- **Is Following Direction:** If set to true, this mover will rotate the game object it is moving, this is applied to the Z axis.
- **Magnet Speed:** The speed which the mover will use when within the target magnet radius.
- **Magnet Radius:** When the mover enters this target radius, it will move using the magnet speed and will start moving towards the target.

Munition Movers

Currently there are 2 types of munition, bullets and missiles. Bullets are moved using a rigidbody2D and do not need any movers, missiles on the other hand require a missile mover and a tracker to follow a target, missiles fired from an enemy weapon track and follow the player and are different from missiles fired from a player weapon which track and follow a random enemy.

When you create a missile you will need to add to it the needed type of missile mover and tracker.

Both types of missile mover scripts are based on the missile mover script and share the exact same list of fields. The only differences between those scripts are the speed multiplier and facing angle.

Note1: The missile mover may appear to be moving at different speeds depending on the direction of movement, but it's only an optical illusion! If you have a scrolling background, if the missile is moving in its direction, it will appear to be slower than when it is moving perpendicular to it.

Note2: The missile mover is different from the magnet mover. Missile mover always moves in its current direction. The magnet mover on the other hand is constantly attracted to its target regardless of its direction. If we receive enough positive feedback on this pack we might add more types of munition movers or quickly adapt the magnet mover as a missile (projectile) mover of some sort.

- **Speed**
- **Is Following Direction:** If set to true, this mover will rotate the game object it is moving, this is applied to the Z axis.
- **Turn Speed:** How fast a missile will turn towards its target.

Missile Mover Following Player (Enemy Missile Mover)

This mover is for a missile fired from an enemy and will seek the player using the player tracker.

Requires:

- **Tracker Player**

Missile Mover Following Random Enemy (Player Missile Mover)

Note: This missile mover does not attempt to follow any enemies which are invincible, this is particularly helpful in the case of a multi-phased boss, where some phases are active but are not damageable from the start (i.e. invincible).

Requires:

- **Tracker Random Enemy**

Trackers

Several components, such as the magnet mover, missile mover, or focus rotator need to know what to aim at or follow, the tracker will provide the information they need, it contains information about the tracked target position, the vector and distance to it. It also raises an event whenever a target is reached.

There are two types of trackers, one which tracks the player and which is used with missiles following the player (enemy missiles), magnet movers, etc.. And another type which tracks a random enemy currently active in the scene and can be used with missiles following random enemies (player missiles) or a focus rotator focusing on the enemies.

Both the tracker player and tracker random enemy do not contain any fields which you can manipulate.

Tracker Player

Uses the player as its target and finds the vector to it, direction, distance and when it has reached it or not. This is typically used with enemy detonators (mines), magnet mover pickups and missiles following the player

Requires:

- **Level (To find the player component used with it)**

Tracker Random enemy

Uses a random enemy as its target and finds the vector to it, direction, distance and when it has reached it or not.

Movers Components

These components add features or rely on accessing mover components. A mover component is any class which inherits from the “Mover.cs” base class.

These can add a rotation, change the sprite or the speed and are explained below

Requires

- **Mover**

Roll By Level Direction

Rolls the game object when it moves sideways in a vertical level or up and down in a horizontal level. This component is useful if you want to give some variation to your agent by rotating them depending on their movement direction.

But please note this script simply rotates the object based on its movement in one axis, In a vertical level it will simply rotate in one direction if the object moves to the right and to the opposite direction if it moves to the left and in a horizontal level, it will rotate in one direction if it moves up and the opposite if it moves down.

It doesn’t take into account any accumulated movement average or the overall direction of movement. If you feel this behaviour is not sufficient for your needs, alternatively you can always animate your objects or modify the component.

Note: The roll by level direction component must be added to an object with a mover, attempting to add it before adding a mover will give you an error as it attempts to add the abstract base class of the mover.

Requires (In addition the mover components requirements)

- **Level**

- **Roll Speed:** The rotation speed on the roll axis which happens when a mover is moving sideways in a vertical level or up and down in a horizontal level. (this value can also be negative)
- **Limit Angle:** The maximum rotation angle that will be caused by this component. A value of zero means it will not be limited and the game object will keep rolling until it stops moving.

Mover Speed Control

Changes a mover's speed after it has spawned based on the curve information you input. You draw a curve and you choose how you want it to loop and it will multiply the speed accordingly. The X-axis of the curve determines the time in seconds and the Y-axis determines the value the speed is multiplied with.

Note: The mover speed control component must be added to an object with a mover, attempting to add it before adding a mover will give you an error as it attempts to add the abstract base class of the mover.

Gradual Sprite Swap

Flips through a series of sprites based on the agent direction. This works only on one movement axis at a time, horizontal or vertical.

When you are in a vertical level, only the left and right directions will be valid for changing frames, but in a horizontal level; the up and down directions are the ones which will be resulting in a sprite change. Hence the use with vertical/horizontal levels headers below.

Note: This sprite swapper, and the 2 swappers next, work on individual frames and not animation. Shmup Boss does not contain a component which starts a particular animation based on a movement direction. You can make animations using the gradual sprite swapper, but it only accepts one axis (2 directions) and will take a series of frames rather than an animated sprite.

Requires (In addition the mover components requirements)

- **Sprite Renderer**

- **Mover:** The mover whose direction will be used to swap the sprite.
- **Sprite Renderer:** Sprite renderer which will be swapped by the images you input.
- **Time Between Frames:** How much time it takes to flip from one sprite frame to the next.

Use With Vertical Levels

- **Left Frames:** The frames that will be swapped through when movement is to the left in a vertical level.
- **Right Frames:** The frames that will be swapped through when movement is to the right in a vertical level.

Use With Horizontal Levels

- **Up Frames:** "The frames that will be swapped through when movement is upward a horizontal level.
- **Down Frames:** The frames that will be swapped through when movement is downward a horizontal level.

Sprite Swapper By Four Direction

Will swap the sprite image of a sprite renderer based on movement in four directions.

Use this script if you are using a 2D sprite for the player or enemy and you would like the sprite to change automatically when the player or enemy moves. This can give you a really nice arcade look and feel to your game and change your static sprites into something alive.

Requires (In addition the mover components requirements)

- **Sprite Renderer**

- **Mover:** The mover whose direction will be used to swap the sprite.
- **Sprite Renderer:** Sprite renderer which will be swapped by the images you input.
- **Up, Down, Right and Left:**
    - **Sprite:** The sprite which will be shown when the mover moves in a direction.
    - **Is Flipped Vertically:** will flip the sprite vertically.
    - **Is Flipped Horizontally:** will flip the sprite horizontally.

Sprite Swapper By Eight Direction

This swapper is exactly the same as the above swapper (By Four Direction) with the exception that it gives you additional directions to change the sprites..

Rotators

These components can be added to any object to modify its rotations.

Simple Rotator

Rotates the object by user input values.

- **X Rotation Speed:** Rotation speed on the X axis.
- **Y Rotation Speed:** Rotation speed on the Y axis.
- **Z Rotation Speed:** Rotation speed on the Z axis.

Circular Rotator

Simple rotator for rotating an object on the Z-axis.

- **Turn Speed:** Rotation change speed on the Z-axis.

Rotation Stabilizer

A very simple component to make sure that the rotation of the game object it is attached to is unchanged by any hierarchy rotations.

Focus Rotator

Rotates the game object on the Z axis to be pointing towards a tracked target.

The focus rotator is a great tool for objects to aim at targets, it could make your simple path weapon direct its bullet at enemies, or you can add it to your player to make an interesting weapon system, or to just place it on enemy visuals or game objects to direct their focus and rotation.

In order for the focus rotator to know what to aim at, it needs a tracker. Please add the needed type of tracker (i.e. TrackerPlayer.cs or TrackerRandomEnemy.cs etc..) If no tracker has been added it will be assumed that this FocusRotator is targeting the player and is aiming towards it.

Requires:

- **Tracker**

- **Turn Speed:** Rotation change speed on the Z-axis.

Main Menu

The components that make up the main menu are in themselves very simple, and you can basically just use and modify the existing main menu scene which is located inside the Space Hero and Space Hero Infinity demos.

But there are few things to pay attention to before attempting to build any game.
All scenes which are saved inside your current Shmup Boss project, have been saved and prepared to be used as a stand alone level (not using the main menu). To make a level be used in the main menu you only need to change these simple variables:

- In the Level component, pick the player source as By Game Manager.
- In the Multiplier component, pick the difficulty level source By Game Manager.
- In the Input Handler input method and auto fire option choose by Game Manager as well.

Save the scene (and preferably build the lighting from the window, lighting settings, generate lighting) and you are done.

Now you can just go to the game manager script, set the indices for your build (main menu and your level scenes), make sure that you have player prefabs listed in your player selections at the game manager, and that you have your avatar indices for the player selection in the main menu UI and that's it.

The approach of separating the main menu from the levels (The by game manager option), gives you the flexibility to include your own levels easily in your own main menu or separate package. It also makes debugging easier as it will disconnect the main menu settings from the levels.

Below we will examine the different components that go into the main menu. Again all you need to do to build a main menu is to duplicate one of the existing scenes available inside the space hero demos.

Game Manager

Handles information which needs to remain across levels and the main menu, such as player selection, saved data and settings. It also handles moving from one scene to another. The game manager is a persistent singleton which does not get destroyed when moving between scenes.

Even if not using a main menu, the game manager will still be created in a level to control things such as sound settings, but you will not need to change any of its variables, if building a main menu then you will need to set up the level indices.

In a standard situation, your main menu scene should have a build index of 0, your first level 1 and your last level is the number of levels you have. Those are set in Unity's build settings and should be referenced in the game manager.

Force Aspect Ratio Background

- **Screen Creator Option:** The background option when you build vertical levels for desktop and the force aspect ratio setting is enabled.

If Option Is Color

- **Background Color:** This is only used if you picked color as a screen creator option and you actually force the aspect ratio. The background color that will cover the empty area on a desktop screen if you

If Option Is Image

- **Background Sprite:** This is only used if you picked background as a screen creator option and you actually force the aspect ratio. The background image that will cover the empty area on a desktop screen if you have forced the aspect ratio for a vertical desktop level.

Scene Indices

- **Main Menu Scene Index:** The build index of the main menu scene. In build settings, the build index is the number next to the scene in the scenes in build.
- **Min Level Scene Index:** The build index of the first level of the game. In build settings, the build index is the number next to the scene in the scenes in build.
- **Max Level Scene Index:** The build index of the last level of the game. In Unity build settings, the build index is the number next to the scene in the scenes in build.

Player

- **Player Selections:** A list of all possible player selections in the game. The amount you put here needs to match what you have in the UI, player menu, avatar canvases. If you are only using one type of level in your game, such as vertical for example you will not need to input a player for the horizontal version.
    - **Vertical:** The prefab which will be used when playing a vertical level, you do not have to reference one if all of your game levels are horizontal. This prefab must have a player component on it.

- **Horizontal:** The prefab which will be used when playing a horizontal level, you do not have to reference one if all of your game levels are vertical. This prefab must have a player component on it.

Events

- **Event System:** If you do not have an event system already in the scene, you can easily create one by right clicking in hierarchy: UI -> Event System.

Main Menu UI

To simplify debugging and expanding the main menu, the main menu UI has been separated into multiple components. All of these components are intended to be used on the same game object.

The easiest thing would be to of course just use the main menu UI prefab located in the prefabs folder, UI and see if you need to modify anything there.

These components are:

Main Menu

Holds references for all the other sub menus such as levels, player select, settings, quit etc.. This will control enabling/disabling the sub menus and going back to the main menu.

I have taken the approach of a generic canvas array so that you are able to add your component or canvas for new functions of the main menu.

Requires:

- **Game Manager**

- **Start Button:** This button will start the first level.
- **Main Menu Canvas:** The main menu canvas that this script controls. This canvas will be activated and deactivated based on what buttons are pressed.
- **Canvases:** All of the canvases that refer to sub menus which are part of this main menu.
    - **Canvas Name:** This canvas name is only used for organizational purposes in the editor.
    - **Canvas:** The canvas which will be activated when the button is pressed.
    - **Activating Button:** The button which will activate the canvas when pressed.
    - **First Selected Button:** The button which will be first selected when the canvas is activated.
- **Coins Text:** This is just to test the accumulation of coins by the player, coins are currently not used to activate anything but make it possible for users to expand this pack and include more features.

Levels Menu

Holds buttons that will allow access to the game levels depending if they have been completed or not.

If you are using an infinite spawner level (A single level), you might want to hide the button leading to the level menu and only keep the start button.

The levels menu uses a scroll view canvas to add level buttons to them, the number of level button is dependent on the game manager level scene indices.

Requires:

- **Game Manager**

- **Scroll View Content:** This scroll view will be propagated with the levels buttons and its size adapted to how many levels in the game there are.
- **Level Button Template:** The button used here will be duplicated to create the rest of the levels buttons.
- **Back Button:** The button which will disable this levels canvas and enable the main menu canvas.
- **Space Between Levels Buttons:** All the levels will be listed in the scroll view, this space determines the distance between each button.
- **Level Button Height:** Uses the height used by the level button adding to it some margin.
- **Scrollview Margin:** The margin for the scroll view which will show the levels buttons.

Player Menu

Sub menu in the main menu to enable selecting the player. ***The number of player selection must match the player selections array in the game manager.***

Requires:

- **Game Manager**

- **Accept Button:** Will assign the currently active player as the selected player and take you back to the main menu.
- **Back Button:** Will take you back to the main menu without assigning any selected player.
- **Next Button:** Displays the next player selection.
- **Previous Button:** Displays the previous player selection.
- **Avatar Canvases:** The canvases which will be displayed when the user scrolls through the player selections, the selected canvas array index will determine the selected player. ***The size of this array must be exactly the same as the player selection in the game manager.***

Settings Menu

Sub menu in the main menu which contains controls for changing game settings such as volume, input method and difficulty.

Requires:

- **Game Manager**
- **Main Menu Audio Manager**

- **Music Slider**
- **Sound FX Slider**
- **Input Method Next Button:** The next button for displaying the different options for the input method (currently available are controls or pointer.)
- **Input Method Previous Button:** The previous button for displaying the different options for the input method (currently available are controls or pointer.)
- **Input Method Text:** The text showing if controls or pointer is used which are selected by the next and previous buttons.

- **Difficulty Next Button:** The next button for displaying the different options for the difficulty.
- **Difficulty Previous Button:** The previous button for displaying the different options for the difficulty.
- **Difficulty Text:** The text showing the difficulty level which is selected by the next and previous buttons.
- **Auto Fire Toggle:** This toggle will enable/disable auto fire mode which makes the player shooting continuously or activated by the fire button (or touch if pointers input method selected).
- **Back Button:** Will take you back to the main menu after saving the settings.

Quit Menu

A sub menu for quitting the game or going back to the main menu.

- **Yes Button:** The button which when pressed will take you out of the game.
- **No Button:** The button which when pressed will take you back to the main menu.

Main Menu Audio Manager

Gives the ability to control the main menu music and sfx via a master audio mixer and mixer groups, and it also enables selecting different audio sound effects for the different main menu UI events in addition to the main menu background music.

Requires:

- **Game Manager**
- **The master mixer which is located in the resources folder**

- **Background Music:** The background music that will be looped and controlled by the music volume in the settings.
- **UI Audio Clips:** An array of all possible audio clips that can be played by the main menu UI. Please be careful not to give two audio clips for the same main menu UI event.
    - **Volumetric AC :** The audio clip that will be played when a main menu UI event occurs.
        - **Ac:** Audio clip which will be played.
        - **Volume:** The volume it will play at.
    - **Main UI Event Trigger:** The main menu UI event that will cause an audio clip to play.

Support

You can always get in touch with us through E-mail: info@ebalstudios.com / https://www.ebalstudios.com/contact or preferably through the forums: https://forum.unity.com/threads/1030870/

The forum thread: https://forum.unity.com/threads/1030870/ will be the place where I will post information about video tutorials, development, feedback and updates and you might want to check it out every once in a while.

FAQ

Usage

- I've changed and customized this package, and I am planning to build a game I am going to sell, monetise or use ads in. Is it Okay?
  Yes.

- Can I use the music, sound effects, 2D illustrations or 3D models you have included with this package for this other game I am making?
  Yes. But what you can't do is post that music, sound effects, 2D illustrations or 3D models for free somewhere, or sell it explicitly.
  Use these items inside your products like a game but do not make them a product on their own,

- Can I use some of your scripts in a game I am making?
  Yes. But do not share them or use them in any packs for the asset store.

- Do I have to credit you in my game?
  No, but it would be nice if you did!

Enemies

- I can't see enemy waves?
  - Layer index is very important, just make sure you have a value higher than whatever other background objects you have in the scene.
  - Double check that the movers are working. During gameplay, see in your scene hierarchy when the wave and enemies spawn, pause the game and see where they are located and why they are not showing.

- I can't hit the enemy?
  - Do you have a collider?

- Enemies pop up in the scene?
  - Use the offset parameter or if you are using an enemy with the patrol mover, move the first way point out of the scene

Weapons

- I can't see any bullets?
  - Check the bullet size or emission rate inside the weapon script.
  - Check if you referenced the weapon in agent and that you have a shooter component.

- I just set a particle weapon but I can't see any bullets in the scene view?
  - Try just selecting any other scene object, then selecting the weapon again to reset it.

- I can't change the weapon settings?
  - The weapon settings change the particle system placed in the same game object, if you have the particle system settings opened then the script can't change these settings. Close the particle system and now you will be able to change the settings in the weapon script.

Background

- I add background layers but can't see them?
  - Most likely it's an index thing, double check your index number and if something is obstructing the layers in the scene view.

- Background objects layers pop up in the scene?
  - Use the offset parameter

- Background objects are not being despawned?
  - Make sure a trigger 2D collider is applied to your prefabs.

General Issues

- Hitting retry gives me a black screen or moves me to a totally different scenes?
  - For the retry button to work, the level needs to be set in the build settings, check your build settings and include your scene there.

- I place something in the scene but I can't see it?
  - Watch and see if it's within the camera field especially in the z position. The index in this package controls the z position and without it you would have to correct it manually.

- How to change input controls?
  - This template uses the standard Unity GetAxis function and should be accessible for changing input, for more info on GetAxis you can check:
    https://unity3d.com/learn/tutorials/topics/scripting/getaxis?playlist=17117
    And:
    https://docs.unity3d.com/ScriptReference/Input.GetAxis.html

    The strings used as input are stored in ProjectInput.cs, currently default input strings are used but if you wish to change them you will need to change the project input script.

- After selecting the player in the player select page, inside the level the player is unchanged?
    - In the level component for each level, did you select the player reference by Level or Game manager? (it should be by manager).

- High score is not being saved?
    - High score is saved only when you properly build a game with a main menu, otherwise it won't be saved when playing single levels. It needs to store the high score in a levels array which also stores which level are completed or not and is strongly connected to a game manager that exists inside the main menu scene.

Terms of Use

This package can be used in any commercial or non commercial video game, you may edit it and release your own game or use one of its components in any of your applications.

You are thus allowed to use this script or any of its components in your own video game. This also includes the artwork, music and sound effects included with this package.

You are not allowed to re-distribute the package in its entirety or any of its elements or assets in any shape or form. You are not allowed to use the scripts, artwork or music in an asset or pack you plan to publish on the Unity asset store or on any other website, and you may not give them away for free or for a price.

For example, you may modify this script and build a game and release it on the app store or the play store. You may use the spaceships, mine models, or music in your game. But you cannot give away the models, scripts or package for free or for a price on the asset store, any other website or through any other means.

Shmup Boss© name and logo are a property of Ebal Studios and cannot be used for any other products and services whether free or paid.

WWW.EBALSTUDIOS.COM
https://assetstore.unity.com/publishers/24304

Huge Thanks For The Music & SFX Provided By
WWW.KLEMELKSTUDIOS.COM

Special thanks go to unity users: OwlHaus, TriplePAF, AndersonGameBr, chingwaifunggames, LeftTurnWorkshop, imm-unity, TheClicketyBoom and Phantasie Your reviews were the encouragement we needed to carry on and actually finish this second version at a time which we had doubt that we should continue.

And more thanks go to unity forums users: Didier Charles, FlyVC, anilcool, DrOcto, Dhruv, Adam Jones, Robert Podosek, Neil Bartlett and everyone on Shmup Baby's and Shmup Boss's Unity WIP forums threads for their support. Your patience and words meant quite a lot to us.