Thank you for choosing this package! If you have any questions or difficulties during your work, please do not hesitate to contact us anytime:

dinv.info@gmail.com

We hope you enjoy your work with our package. **We also would be grateful if you could leave your feedback or review, it is extremely important to us!**

This package contains space shooter game template, which allows you to create your own game using ready-made graphics and customizable script. Use this manual to build your own Scene, create Enemies, define Shooting and so forth.

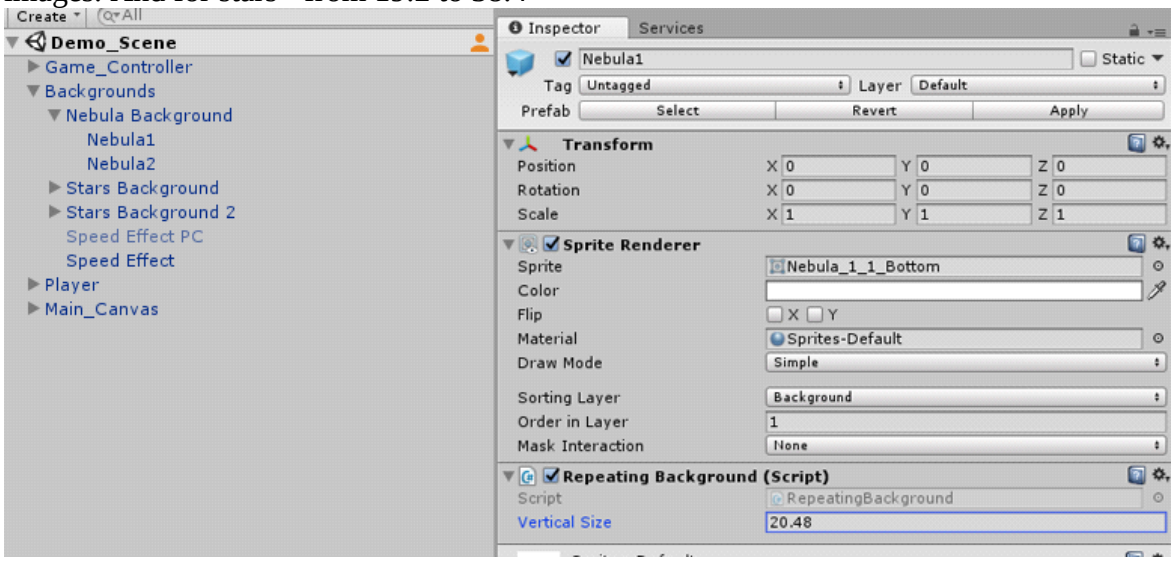
The package contains Demo-Scene with a turnkey level that demonstrates the possibilities of the game.

And now we will explain you how to create your own Scene.

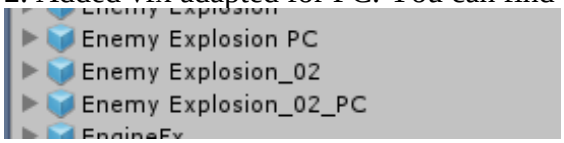
Update 1.2

1. Added background sprites adapted for PC. You can find them in "Sprites - Background" folder.

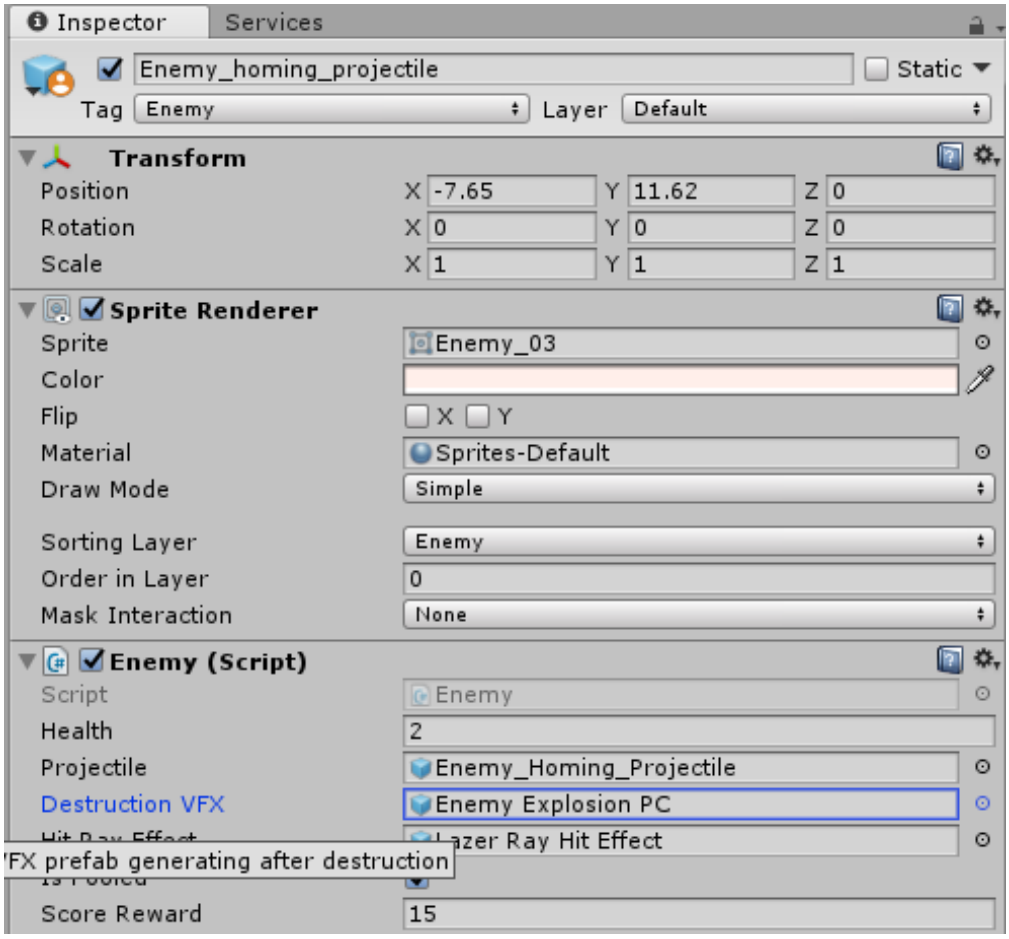
If you want to use them, go to prefab "Backgrounds" on scene, choose object Nebula1 for example, and replace the spite. Then drag and drop them in the game scene to fit them correctly.
Also, change the value of "vertical size" in the inspector from 20.48 to 40.96 for Nebula images. And for stars - from 19.2 to 38.4



2. Added vfx adapted for PC. You can find them in "Prefabs - VFX" folder.



To apply them, choose enemy prefabs, and change explosion prefab like on screenshot below:

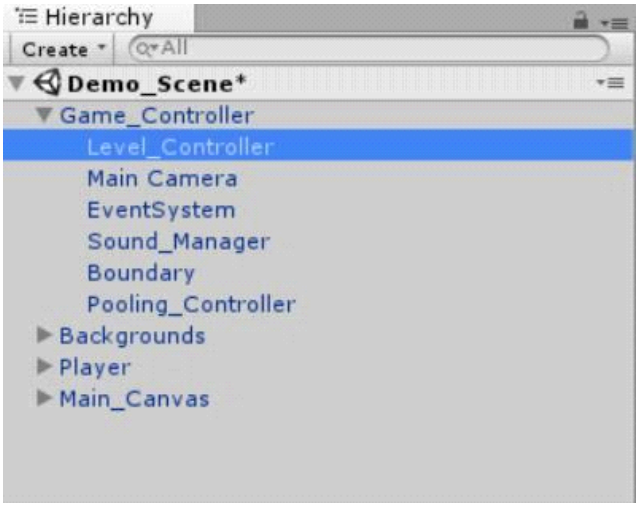


Also, you can turn on the Speed Effect for PC on the scene.

How to create a Scene

To build a level, you need four elements:

- ‘Game_Controller’
- ‘Backgrounds’
- ‘Player’
- ‘Main_Canvas’



All ready elements for the game are located in the “Prefabs” folder. What each of these elements is needed for?

‘Game_Controller’ contains objects, which control the game.

‘Backgrounds’ object contains backgrounds for the game and controls background layers.

‘Player’ object controls the object of a player.

In **‘Main_Canvas’** is located information about scores, player’s health and any other elements you want to add.

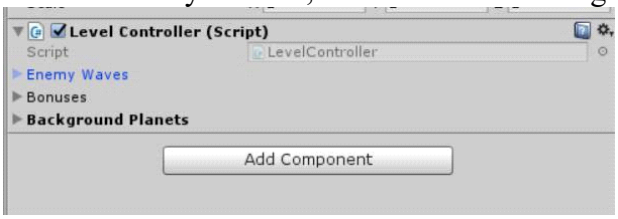
How functions ‘Game_Controller’

‘Game_Controller’ object contains a script that counts player’s score and health points and draws this information to the ‘Main_Canvas’. Also to the ‘Game_Controller’ object are attached other objects, such as:

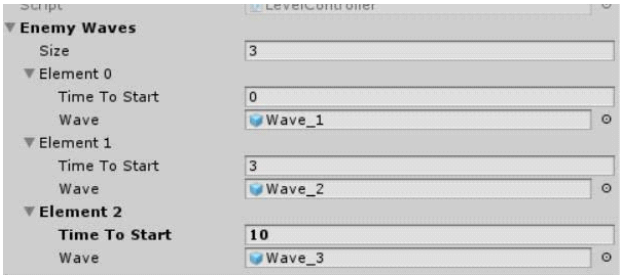
- **‘Main_Camera’**;
- **‘Sound_Manager’** - an object that plays all sounds in the game;
- **‘Boundary’** object indicate objects that cross the line of the game field, destroys or deactivates them;
- **‘Pooling_Controller’** contains the list of the Prefabs, like Enemies or Projectiles and pools them on command or if necessary;
- **‘Level_Controller’** controls game process of a given level, in particular produces Enemies, Bonuses and Background Objects.

How to launch a level

To launch a level, you need to set fields of ‘**Level_Controller’** object, namely ‘Enemy Waves’, ‘Bonuses’ and ‘Background Planets’.



In the ‘**Enemy Waves’** category you need to specify the number of the enemy waves, which would appear in this level. Also you need to indicate the starting point from the beginning of the game, when each wave appears (in seconds).

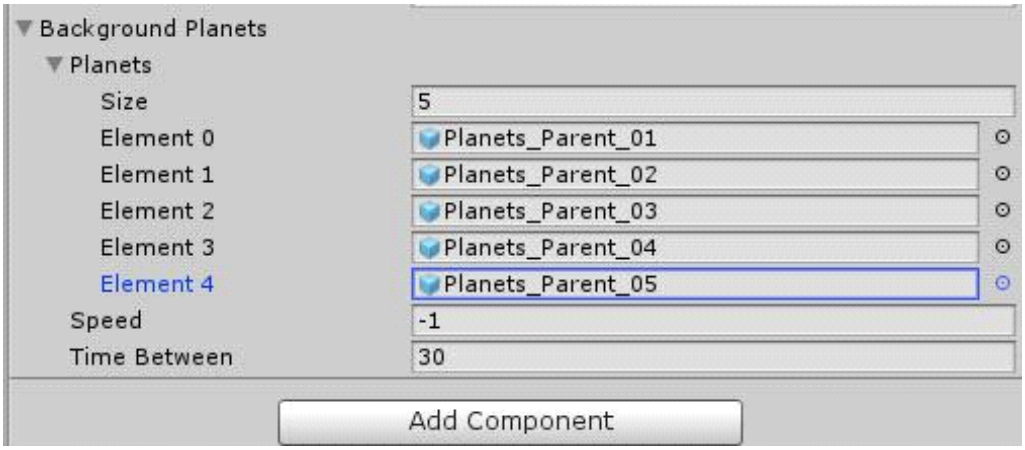


Previously prepared waves are added from the folder ‘Prefabs’ – ‘**EnemyWaves**’.

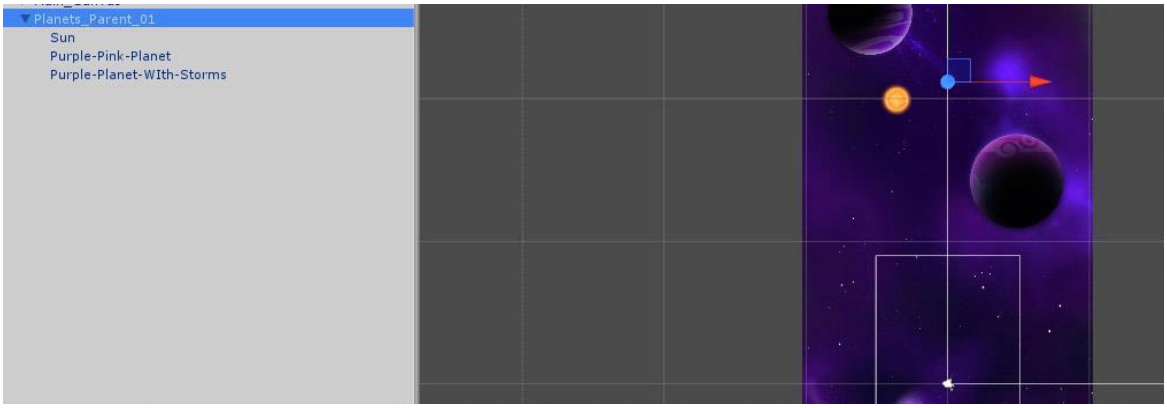
In the ‘**Bonuses’** category are specified prefabs of the bonuses, and frequency with which bonuses are appearing (per second).

In the ‘**Background Planets’** category is specified a list of planet compilations, the speed with which they are moving and frequency with which a new compilation is appearing (per second). You can create your own planet compilation: just copy one of the objects in a ‘**Planets compilations’** folder and change the objects in its hierarchy.

In ‘**Planets compilations’** folder you find five premade prefabs, with name "Planets_Parent". You can choose them in ‘Level_Controller’ like on screenshot below:

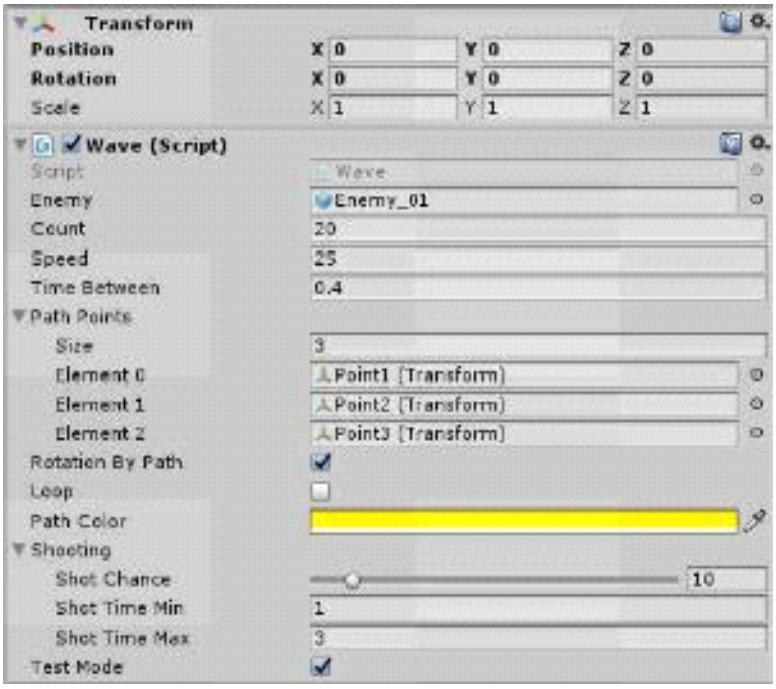


Important: Make sure to put all the "Planets_Parent" objects you create above the main camera:



How to create Enemy Wave

To create a new Wave you can take one of the prefabs from the folder '**Enemy Waves'** and duplicate it. It contains a script '**Wave'**, which generates a certain number of enemies during a certain period of time and runs them along a certain path.



The fields of the '**Wave'** script:

- '**Enemy'** – prefab of an Enemy that will be generated during this Wave
- '**Count'** – the number of enemies in this Wave
- '**Speed'** – the speed with which an Enemy completes the path
- '**Time Between'** – a time period between the emergence of the Enemies
- '**Path Points'** – points which an Enemy will pass during the completion of the path
- '**Rotation Bypass'** – whether an Enemy would rotate during the completion of the path
- '**Loop'** – if loop is active, after the completion of the path, an enemy would be re-generated at the starting point;
- '**Path Color'** – a color that will mark the path in the Editor;
- '**Shooting'**:

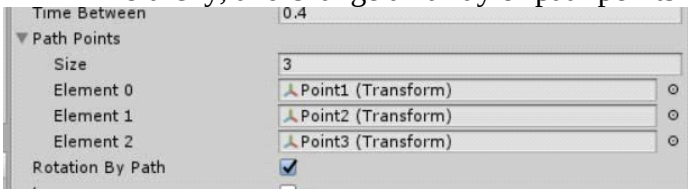
    'Shot chance' – probability of the enemy shot during the completion of the path, and minimal and maximal time when an enemy would attempt a shot.

- '**Test Mode'** – if test mode is active, a wave will be generated constantly with the 3-sec interval. You can use the test mode for a new wave customization.

You can customize a Wave in a 'Play Mode'. Place a wave into the Scene and deactivate Level_Controller to disable the appearance of other Scenes. Turn on 'Scene View' and setup 'Play Mode'. You can change points' position in a hierarchy of wave object.

You can change types of enemies, a number of enemies and other parameters to see changes of the wave in a live mode. Name the wave and save it in Prefabs.

If you want to change the number of path points, add or delete a point in a wave hierarchy, and change an array of path points in the Inspector.



When you created all the waves you needed, you can add them to the Level_Controller.

'Player' Customization

The object '**Player'** includes three scripts:

- **Script 'Player'** stores information of player's health points, processes damage, sends the information about player's health to

Canvas.

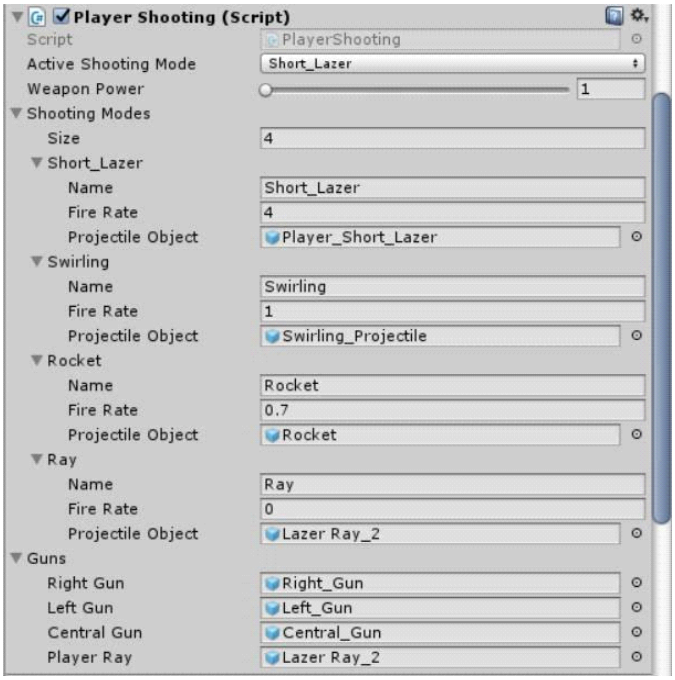
The fields of the '**Player' script**:

- **Health Indicator** – reference to the image of player's health scale to Main Canvas;
- **Damage Frequency** – how long after receiving damage points the player won't get new damage points;
- **Health Indicators Sprites** – sprites of health scales
- **Shield** – object shield in player's hierarchy (deactivate it before game is started)
- **Destruction FX** – prefab of video effect that is generated after player's death.

- **Script 'Player Moving'** controls player's movement, sets borders of his movement. Depending on chosen Handling Type it moves a player.

- **Handling Type** – depending on chosen handling type, the player moves on a touch position or on a touch position with offset.
- **Borders** – an indent from each border of the screen, which the player cannot pass.

- ‘**Player Shooting’** Script defines the mode, in which the player shoots, and with what frequency.



The fields of the '**Player Shooting'** Script:

- **Active Shooting Mode** – the current mode of player's shooting
- **Weapon Power** – power of player's weapon, which influences shooting (by default from 1 to 4)
- **Shooting Modes** – enumeration of available shooting modes with the following fields:
  - Name of shooting mode
  - Fire Rate – frequency with which the player shoots (the higher the more often)
  - Projectile Object – prefab of projectiles of shooting mode.
- **Guns** – guns and ray objects in player hierarchy (deactivate Ray before the game is started)

How to change game objects' parameters

Almost any prefab object in the game, such as Enemy, Projectile etc. can be changed. You can find a prefab of an object in a 'Prefabs' folder and change its parameters. For instance, you can change the speed of the enemy projectile or

enemy sprite, or the power of player's weapons. Just find the needed prefab in the folder and change its fields.

Backgrounds

‘**Backgrounds’** object contains background’s layers. Every layer moves along the camera with the set speed. Accordingly, each layers has two sprite objects, and rearranges them, creating the effect of endless movement.

You can change background sprites, however for the correct endless scrolling effect, do not forget to set the value of ‘Vertical Size’ field, which equals to the size of the new sprite along the Y axis.

Also object ‘**Backgrounds’** contains the object ‘**Speed Effect’** – particle system which also can be customized.

Sound Manager

Sound manager controls all sound in the game.

To use a background music add your clip to the 'Background Music' field and it will play when your game starts.

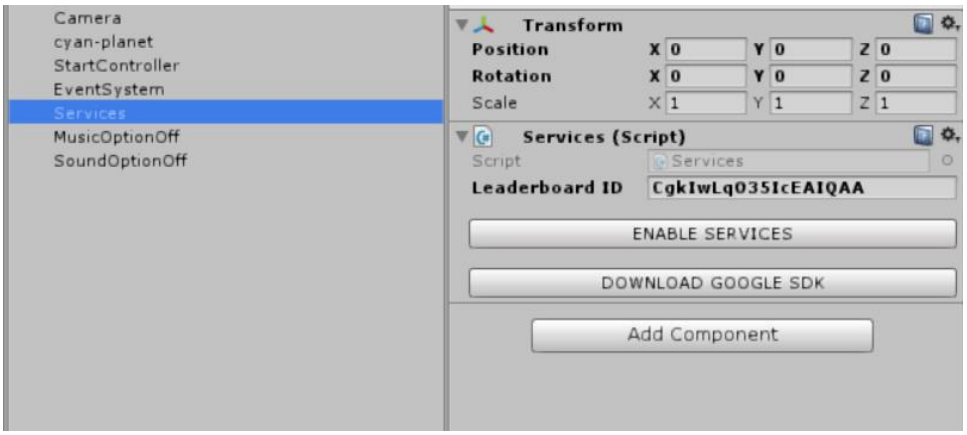
Add the audioclip to appropriate field to play a needed sound.

If you want to add a new sound, you can add a new AudioClip to the attached '**Sound Manager Script'** and play it from any other script using method SoundManager.instance.PlaySound("<your sound>");

Leaderboard

If you're going to use leaderboard in your game follow these steps to include this option:

- Add prefab "Services" to the very first scene.



- In case you create a game for Android platform press the button "DOWNLOAD GOOGLE SDK" to download sdk plugin.
- Press the button "ENABLE SERVICES". If you're not going to use leaderboard disable it to avoid any errors.
- Insert ID of your leaderboard.
- Done! The score amount will be automatically sent to the leaderboard right after "Game Over".

Scripts

The pack contains a number of different scripts, and inside every script you will find the description of the steps of the code. You are able to change parts of the code according to your needs. If you have any questions or suggestions please contact us: DinV.info@gmail.com

Ads Manager

If you want to add advertisement to project, see the file "Ads Manual" which you can find in the "Documentation" folder.