2D Physics Kit

Requirements: Unity 5.6.0 or higher.
Support: frostinglg@gmail.com
If you have any questions or suggestions, please, don't hesitate to write me.

2D Physics Kit contains of phenomenal physics elements for making a game based on Unity Physics2D. It's really easy to use and customize, just build it on your device and try! Also feel free to use it inside editor, it works like a charm! Every element of the package is customizable inside the editor without touching a single line of code. With this asset you can quick start making your own 2D game. Just add your sprites and spend a few minutes to set up behavior for each physics element and enjoy!

Coming soon

Our plugin is in constant development. This means that gradually will be added new 2D physics elements, will feature new styles of the visual design of objects for different styles of games.

Here is a list of development:

- Rocket Launcher - done
- Shell templates - done
- Gun templates - done
- Falling Platform - done
- Disappearing Platform - done
- Huge refactor - done
- Trajectory prediction - done

Traps will be added in next update

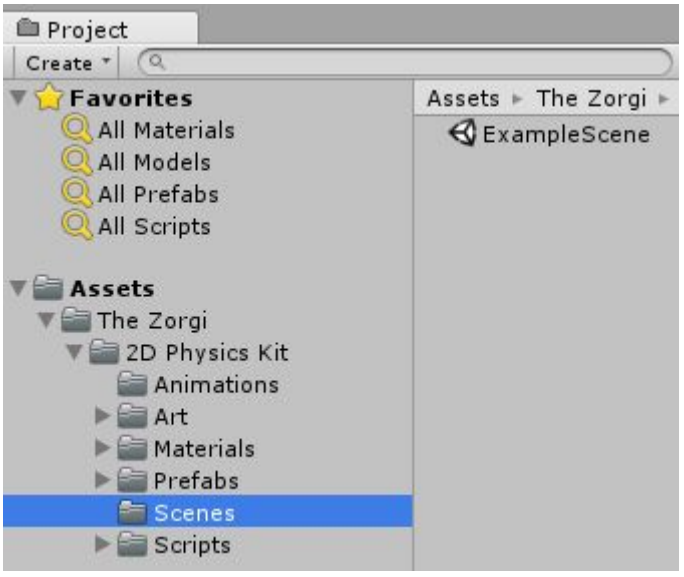
- Explosion trap
- ForceExplosion trap
- Smoke trap
- Lighting detection

This list will be constantly updated with new technologies that will minimize your effort to create interesting and exciting game.

If you have any questions or suggestions, please, don't hesitate to write me - frostinglg@gmail.com.

Example scene

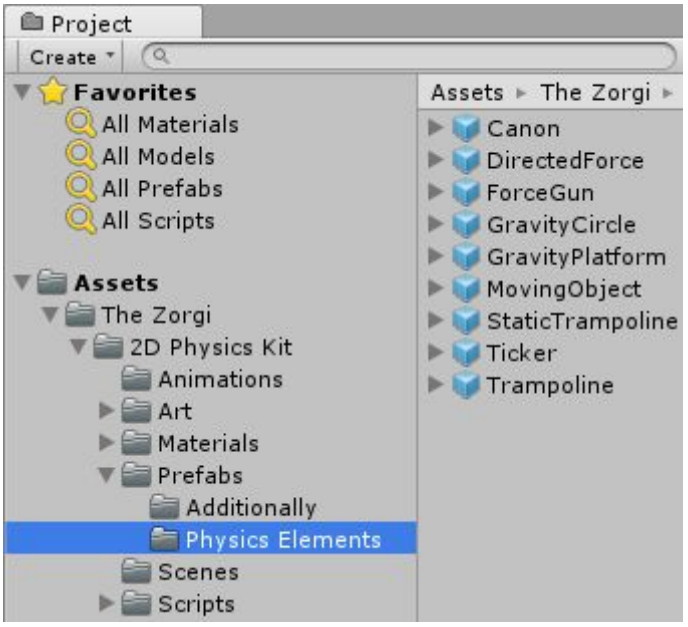
You can find example scene here: *Assets -> The Zorgi -> 2D Physics Kit -> Scenes.*



First start guide

This part is going to explain how to work with a (prefab) physics element from scratch, tough it's recommended to start by **making a copy of the demo scene** and experimenting with it.

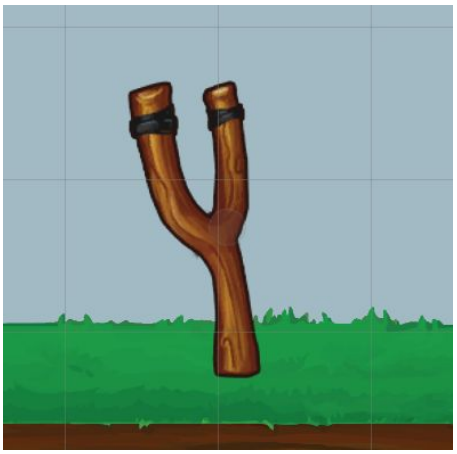
1. Create a new scene.
2. Play the scene, you will see how work our prefabs with default directions and predefined variables.
3. From the project inspector drag & drop some prefab, which you will use for experiment.



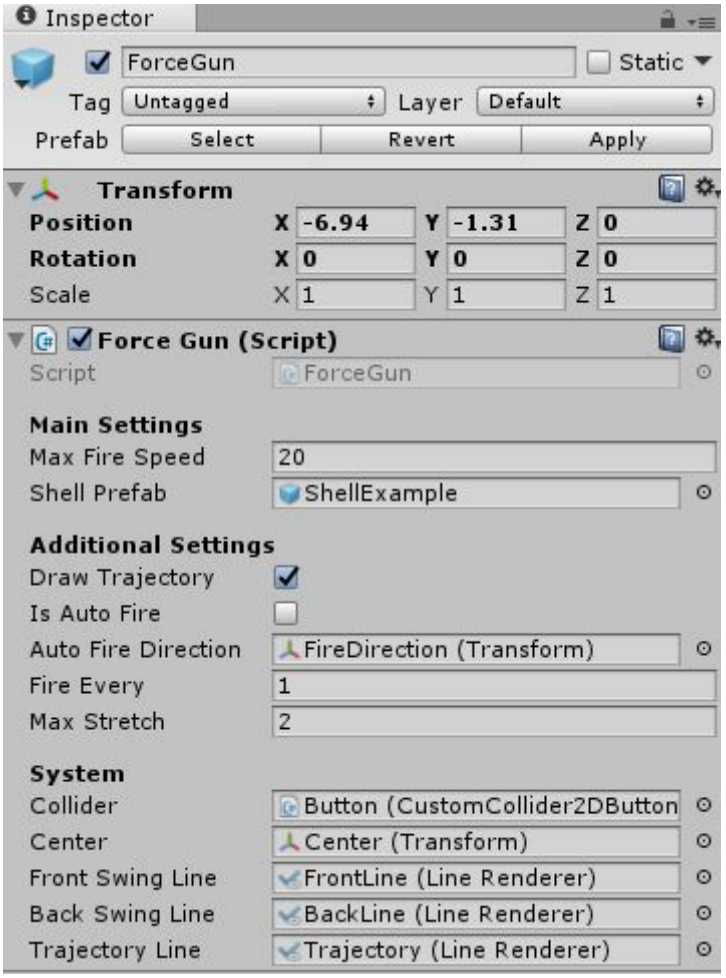
4. Start customizing the physics object inside the inspector as you prefer, change sprites materials, edit sizes, add new layers, etc.

API reference

Force Gun



Force Gun - object that can throw "bullet" prefab automatically or by user handle. Looks like slingshot from very known game about unhappy birds.



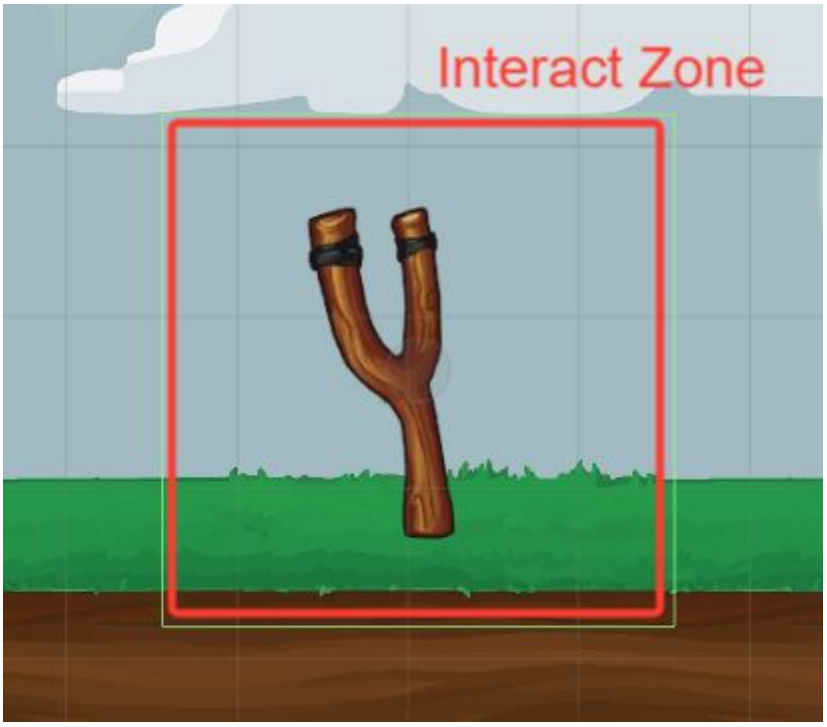
Parameters:

1. Max Fire Speed - bullet speed with max stretch of sling.
2. Shell Prefab - object that will be used like a shell.
3. Draw Trajectory - trajectory toggle.
4. Is Auto Fire - auto fire toggle.
5. AutoFire Direction - autofire direction, using only direction without vector magnitude.
6. Fire Every - time between shots (in seconds).
7. Max stretch - max stretch of sling.
8. System.

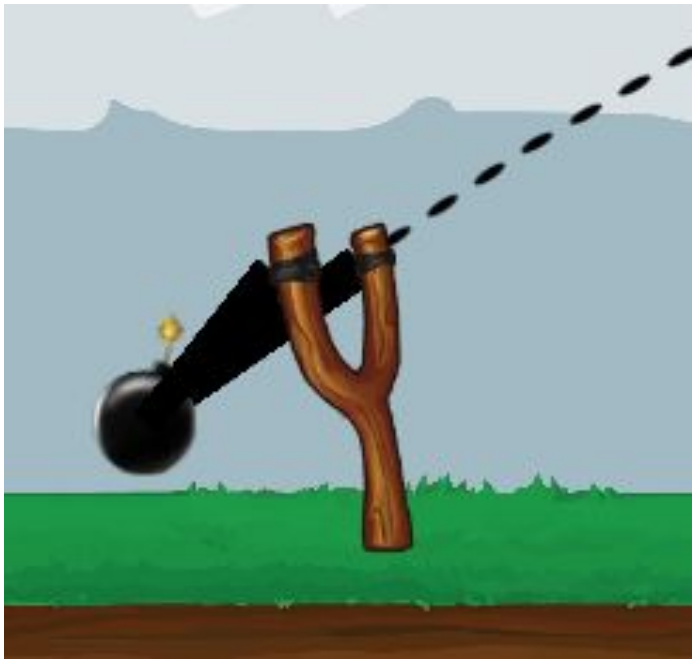
Components:

1. FireDirection - aiming fire direction in automatic fire mode.
2. Center - bullet default position, center of Force Gun physics.
3. Button - zone that can be hitted to interact.
4. Visual - visual part of Force Gun (sprites, trajectory, slings).

Usage:

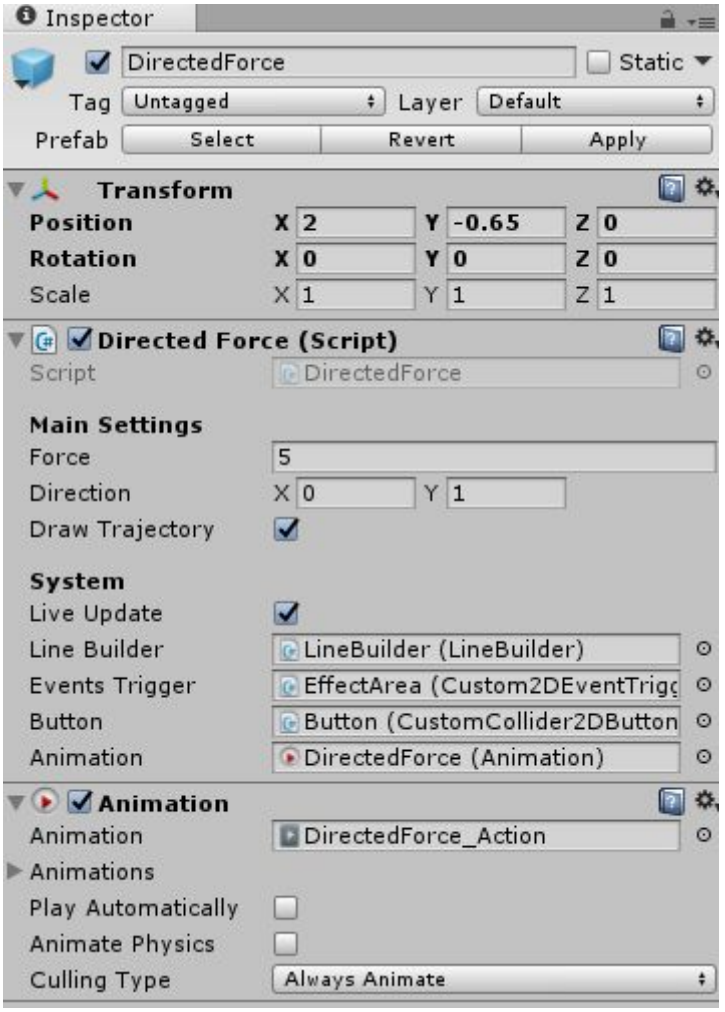


1. Click to load a shell.
2. Choose direction and release.



Direction Force

Direction Force - object that affects all objects in "effect area" with force N on (X,Y) direction.



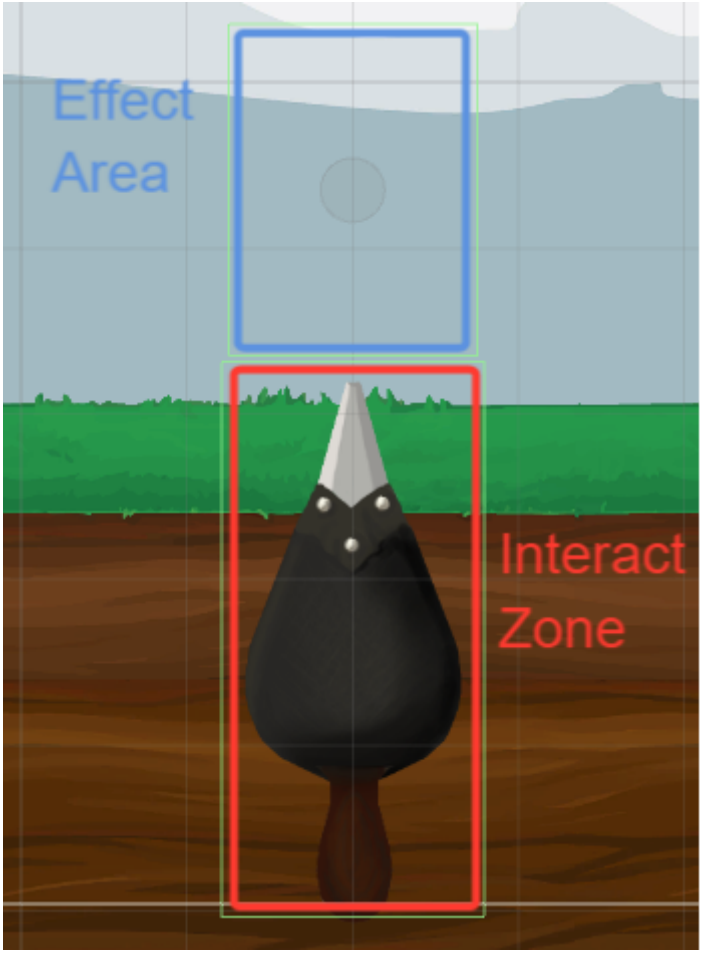
Parameters:

1. Force - force amount that affects objects in effect area.
2. Direction - force direction.
3. Draw Trajectory - trajectory toggle.

Components:

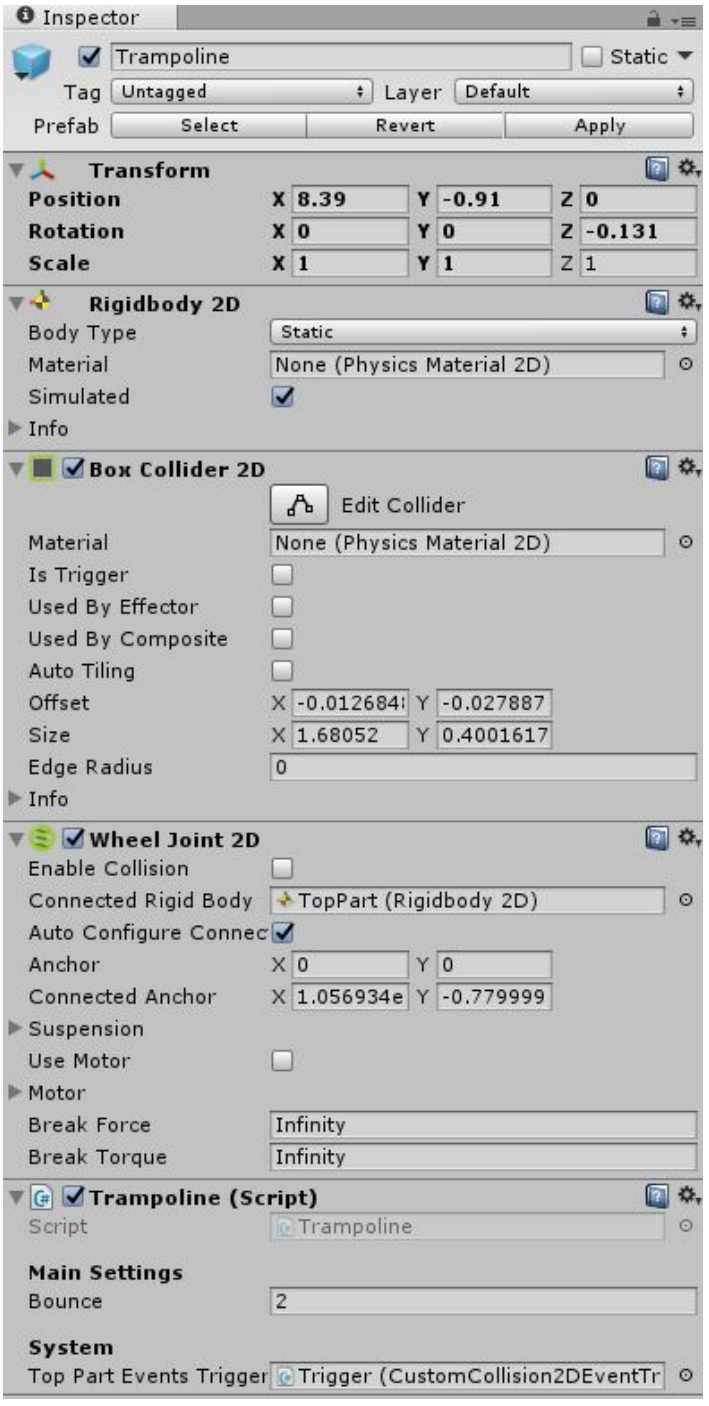
1. Effect Area - area where objects can be affected.
2. Button - zone that can be hitted to interact.
3. Visual - visual part of Direction Force (sprites, particle, line that shows effect area).

Usage:



Trampoline

Trampoline - object that bounces all objects that collide with top part.



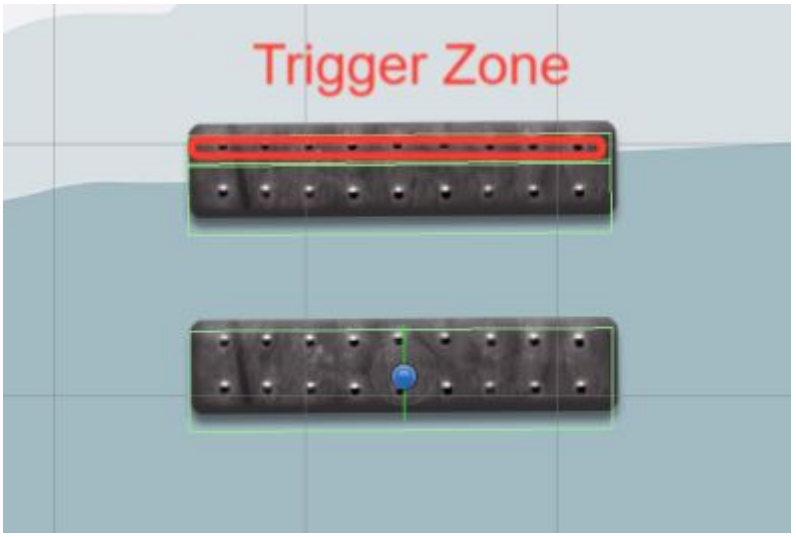
Parameters:

1. Bounce - bounce parameter (1 - natural).

Components:

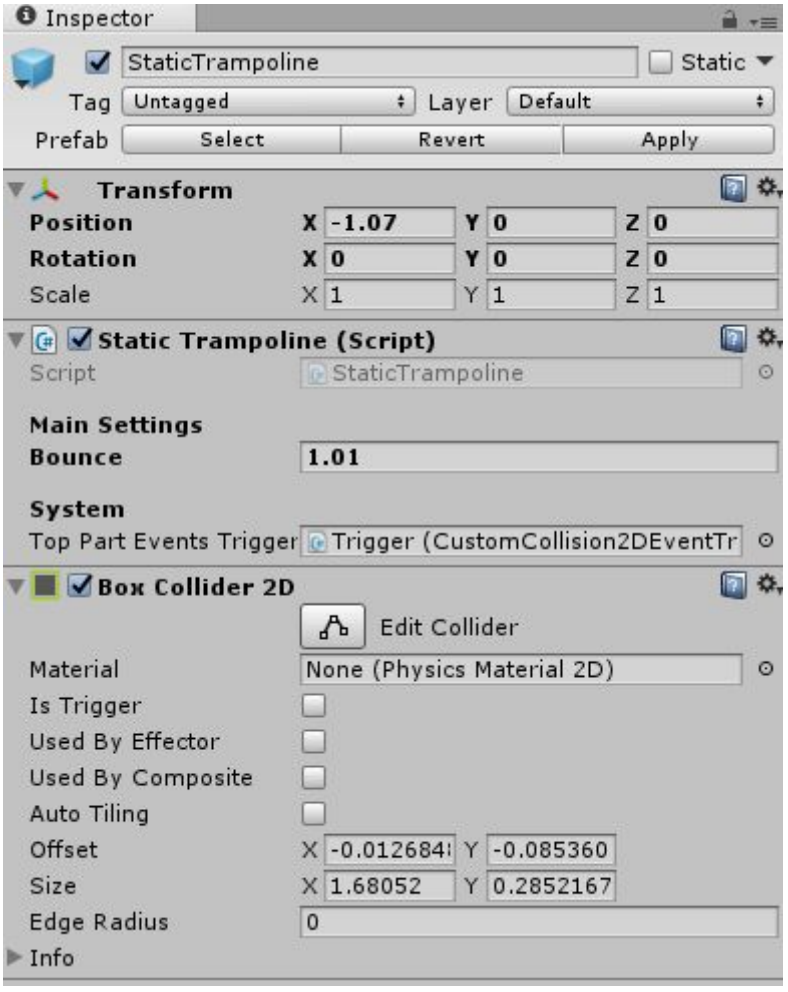
1. Visual Bottom - bottom part sprite.
2. Top Part - top part of trampoline that contains trigger and top sprite.

Usage:



Static Trampoline

Static Trampoline - analog of trampoline but it's static and has only 1 part.



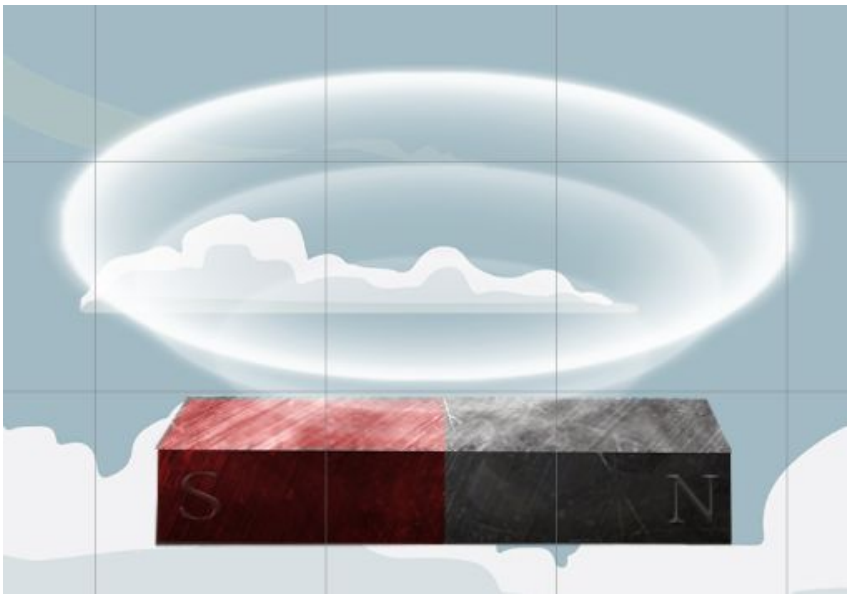
Parameters:

1. Bounce - bounce parameter (1 - natural).

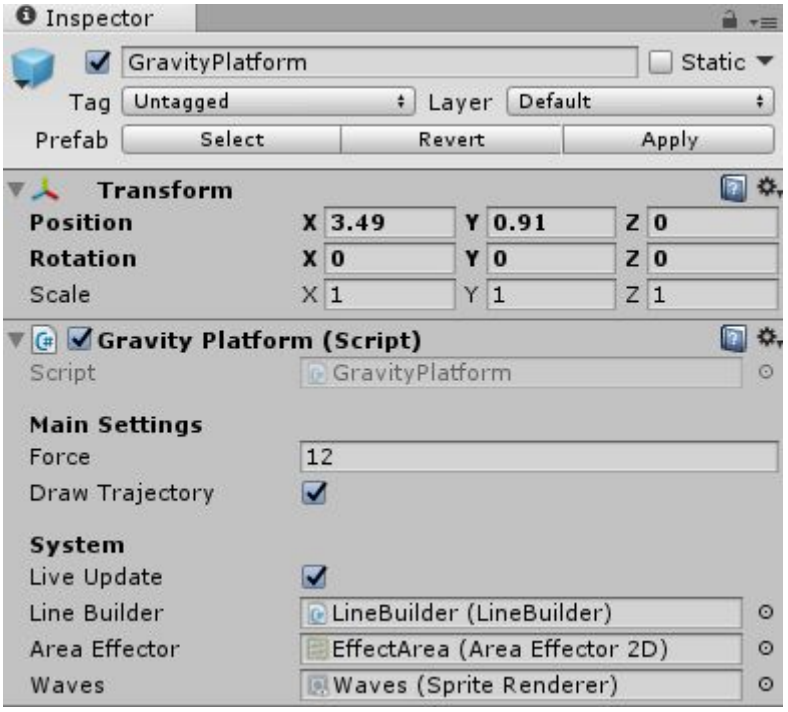
Components:

1. Trigger - bounce zone.
2. Visual - sprite.

Gravity Platform



Gravity Platform - platform that constantly affects objects in area with vertical force (gravity).



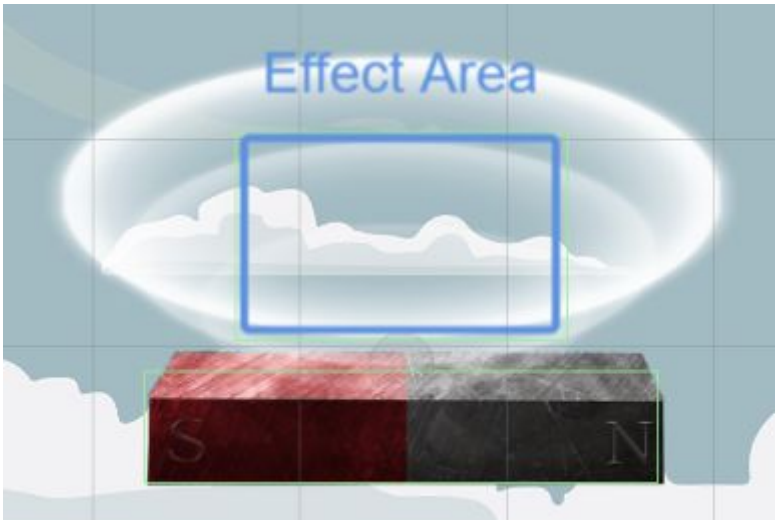
Parameters:

1. Force - force parameter (**9.8 - natural gravity, bigger - objects move on top, smaller - fall down faster**).
2. Draw Trajectory - trajectory toggle.

Components:

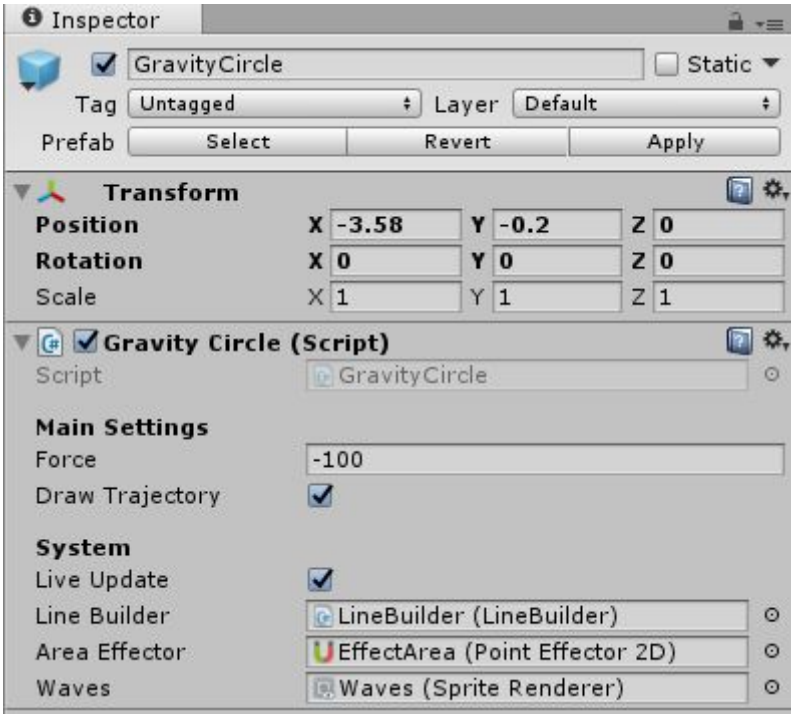
1. Effect Area - zone where objects can be affected with additional gravity.
2. Platform Collider - platform collider.
3. Visual - visual part of Gravity Platform (effect zone, waves, platform sprite).

Usage:



Gravity Circle

Gravity Circle - gravity circle that constantly affects objects in circle area with radial force.



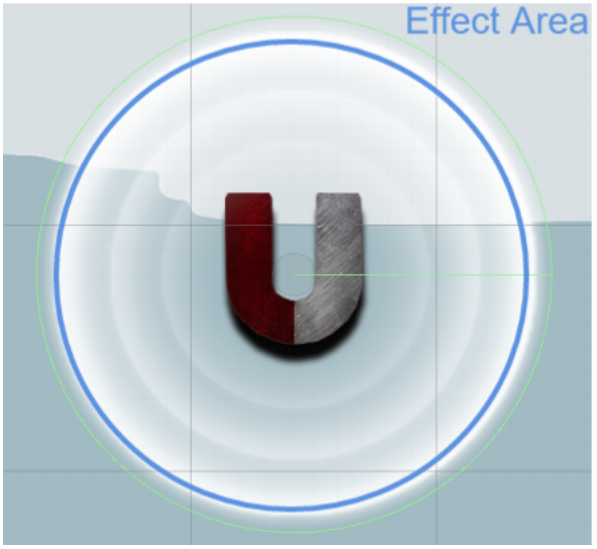
Parameters:

1. Force - force parameter (**<0 - to center, >0 - outside**).
2. Draw Trajectory - trajectory toggle.

Components:

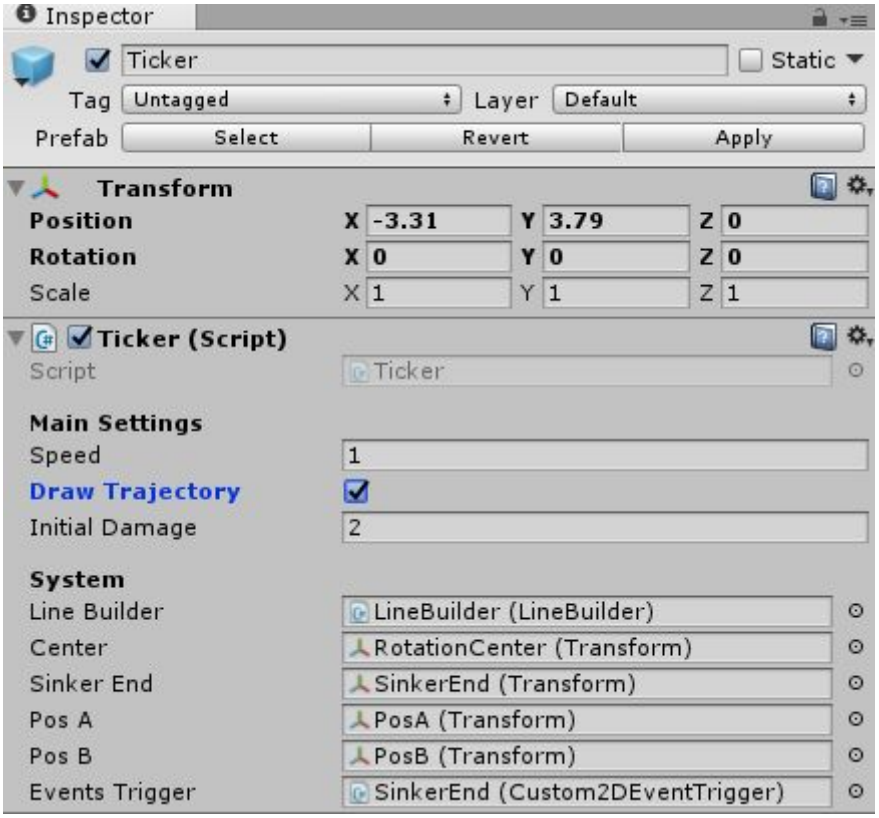
1. Effect Area - zone where objects can be affected with radial force.
2. Visual - visual part of Gravity Platform (effect zone, waves, magnet sprite).

Usage:



Ticker

Ticker - periodic ticker that moves with N speed between 2 points.



Parameters:

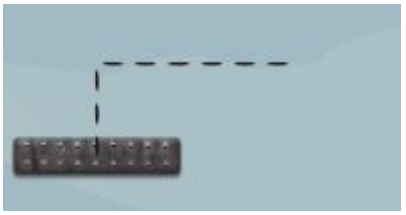
1. Speed - move speed.
2. Draw Trajectory - trajectory toggle.
3. Initial Damage - the damage which player will get after falling into a trap

Components:

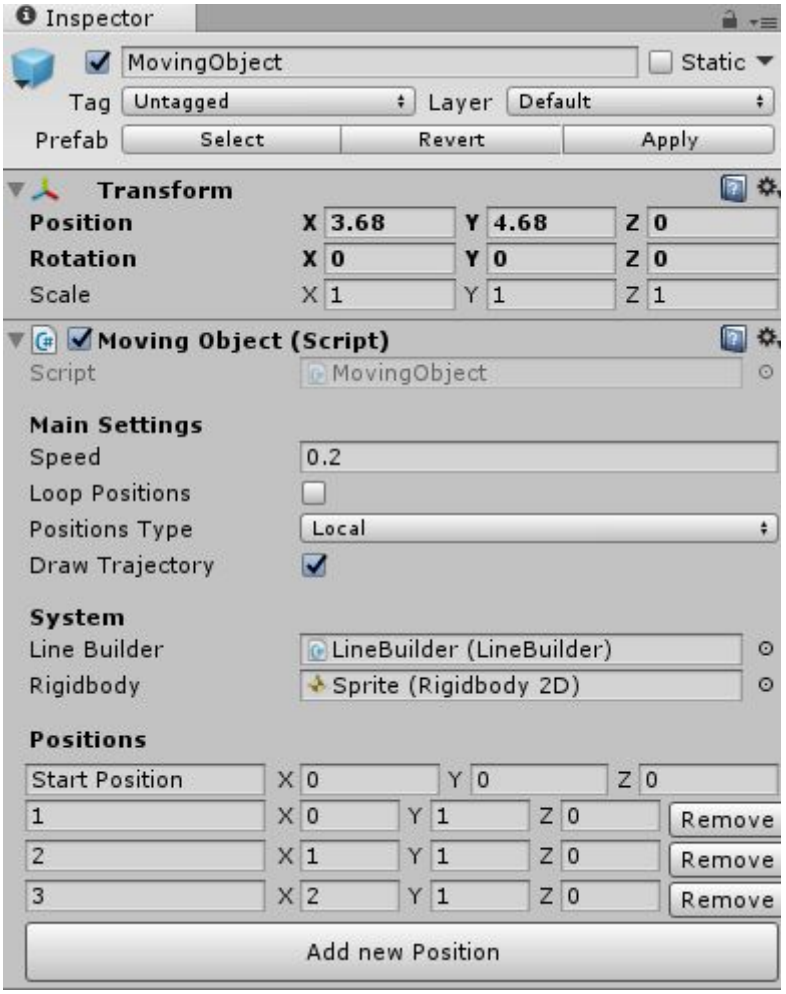
1. Rotation Center - center of ticker rotating.
2. PosA - position of ticker start.
3. PosB - position of ticker end.

4. Visual - trajectory of moving zone.
5. Events Trigger - axe blade now deals damage (look Initial Damage above) to the character.

Moving Object



Moving Object - platform that moves by setted positions (loop or not).



Parameters:

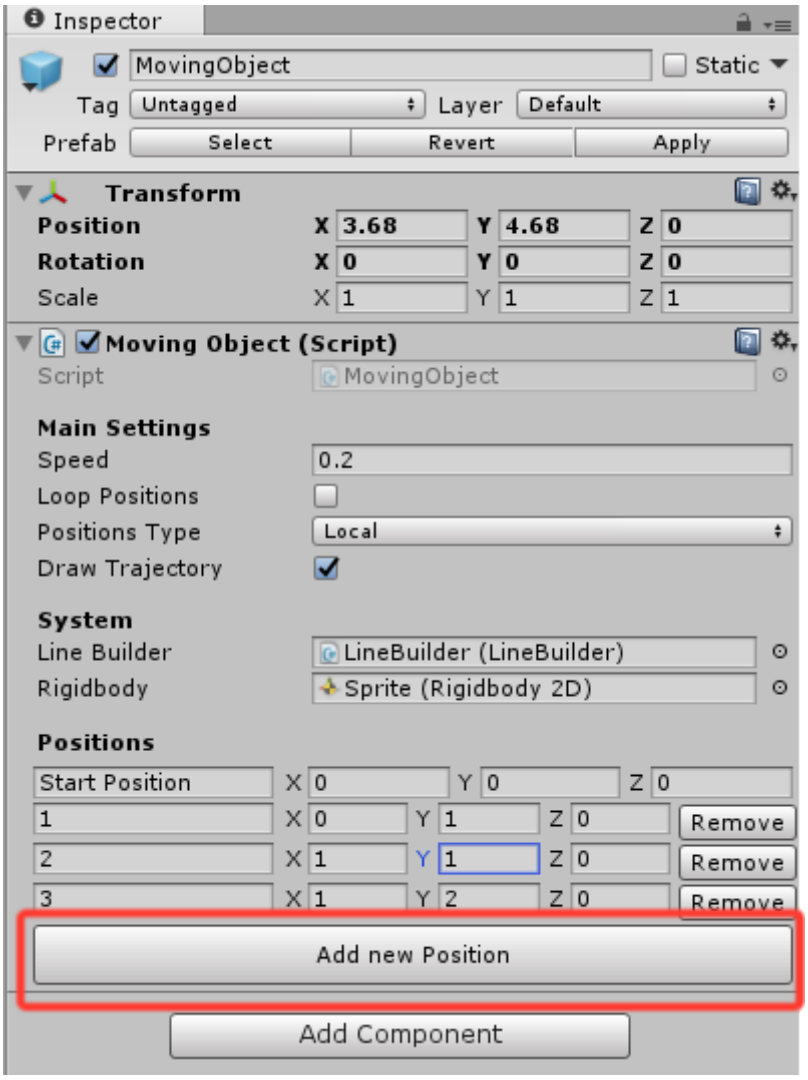
1. Speed - move speed.
2. Loop Positions - loop toggle, after the last position platform moves back or to the first position.
3. Position Type - positions type in space.
4. Draw Trajectory - trajectory toggle.
5. Positions - move positions.

Components:

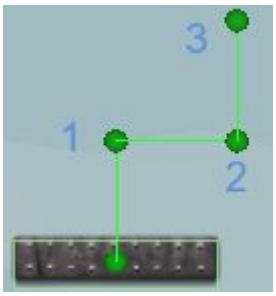
1. Visual - sprite and move trajectory.

Usage:

You can add new position for moving object by using appropriate button in the Inspector.

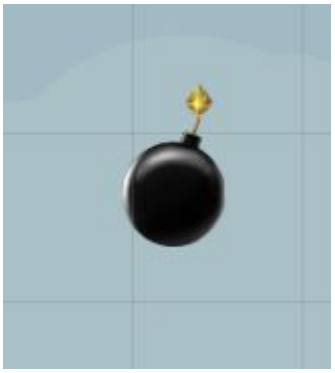


For example We added three positions for moving object. It will move in sequential order to each of the following positions.

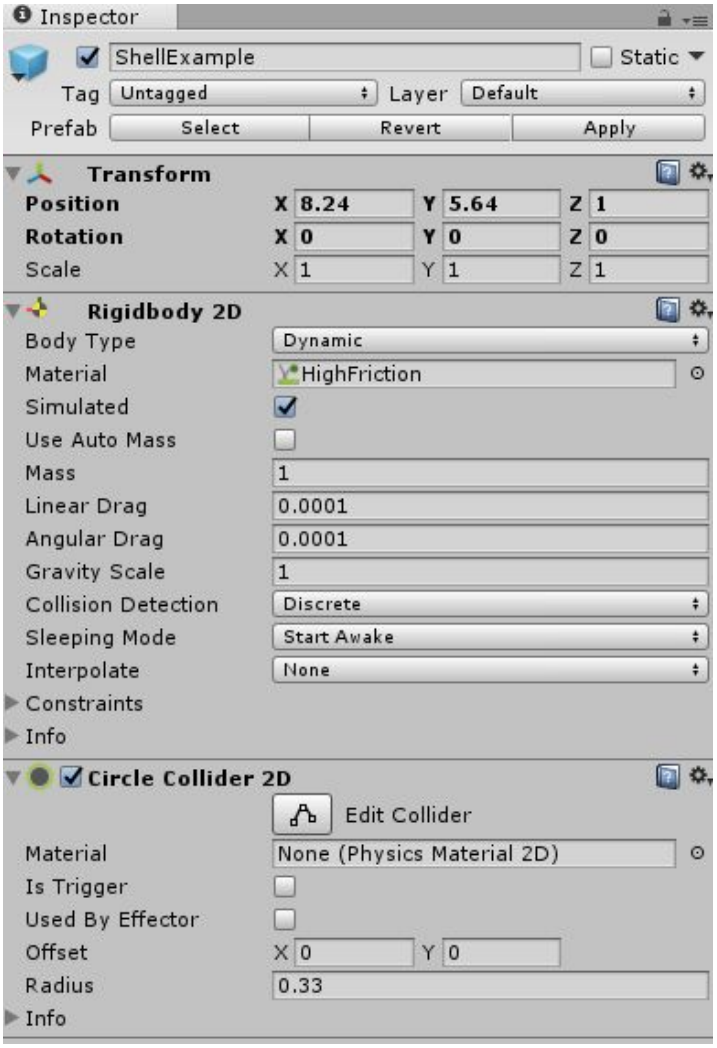


By default Unity disable Gizmos of objects on the scene. So you need to enable it to display the new created positions:

Shell Example



Shell Example - object that will be used like a bullet.



Creating trajectory

If you want to create your own physics element, but you don't know how to create trajectory - use our helper LineBuilder.

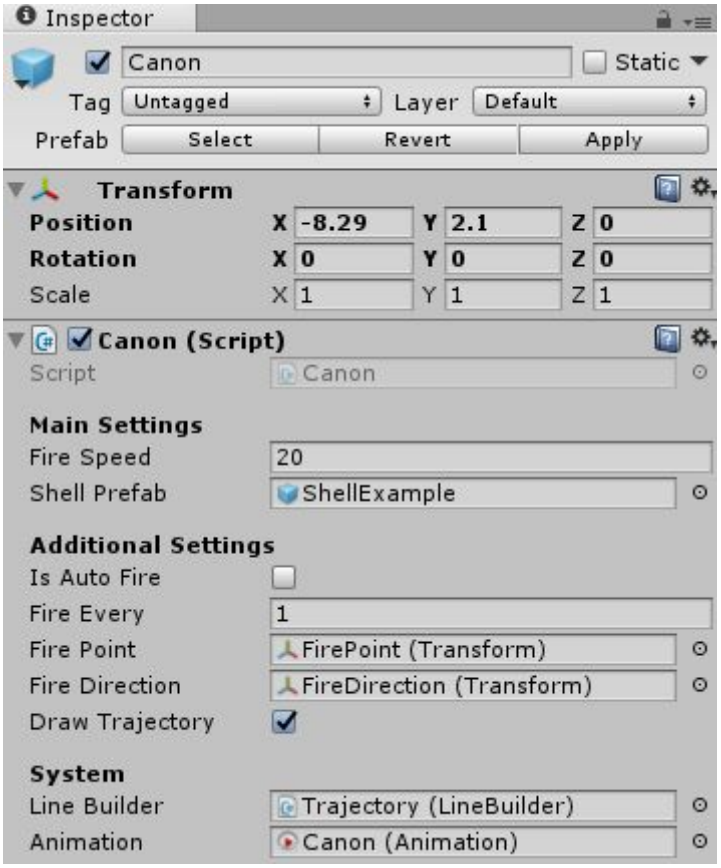
LineBuilder - tool, that uses Unity basic component - LineRenderer. It doesn't have Editor interface. Use it only via script by the calling method:

public void DrawLine(Vector3[] positions, bool isWorldSpace)

This tool may be used by anyone, it's just a small life helper.

Canon

Canon - fires shells in right direction. It works in two modes: Manually and Automatically.



Parameters:

1. Fire Speed - speed of a bullet.
2. Shell Prefab - shells fired by a cannon.
3. Is Auto Fire - switch firing modes.
4. Fire Every - when "**Auto mode**" has been chose the cannon will fire every (specify number) seconds.
5. Fire Point - position from where the shells fly.
6. Fire Direction - direction of the shells fly
7. Draw Trajectory - shows / hide trajectory of shells fly.

Components:

1. Effect Area - zone where canon can be affected with Mouse Click in order to fire shells.
2. Visual - sprite and move trajectory

Usage:

In order to modify behaviour of the shells fly you should change two parameters - "Fire Point" and "Fire Direction" respectively.

Fixes(will be available with Update 1.1):

Special thanks to one of my customers. This guy helped me find the bug by using Canon with Perspective Projection of Main Camera.

****Has been fixed bug due to using Perspective Projection of Main Camera***

Update 1.1

Good day, Friends! For some time I could not release the updates of the plugin due to the excessive workload. However, now I am full of strength and motivation to continue to develop this asset. Thanks to those who have already purchased the 2D Physics Kit. You give me more space to develop, and that is why I decided to devote a lot of time to this product. Soon I will upload a new version of the plugin, which will contain new physical objects for your interesting games. Every month there will be a new update, that is, during the year a huge number of interesting objects will be added for the interaction with your game world.

Features

- ***Has been added new physics elements type - Traps***
- ***Added 3 new physics elements***
- ***Charge - this trap works like electric object that moves by setted positions***
- ***Moving Staircase - this physics object gives acceleration to any object that falls on it***
- ***Unlocked Key - this physics object is responsible for collecting keys count***

Updates

- ***Ticker now dealing damage when axe blade contacts the player***
- ***Added a interfaces for convenient and flexible use physics elements. Now you know where put your code to add animation, music, sound, etc.***
- ***Was made code refactor. Now it more convenient and looks like more clean.***

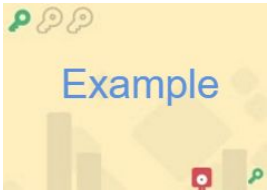
Fixes

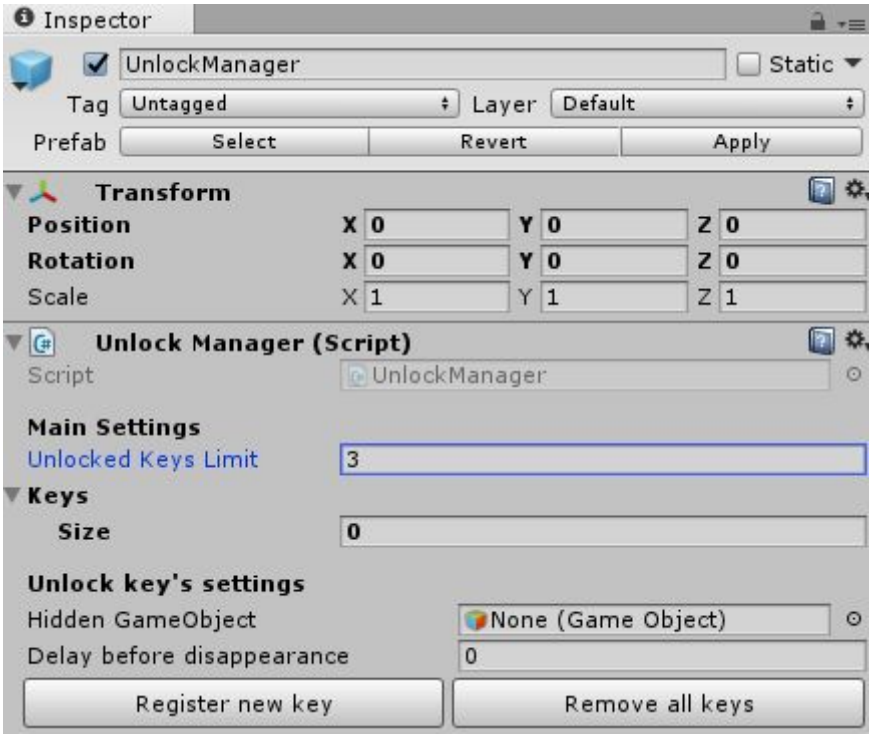
****Has been fixed bug with Canon due to using Perspective Projection of Main Camera***

Unlock Key

Unlock Key - is responsible for collecting keys count. This physics element operates the keys-work logic. By using it you can easy open door, unlock chest, disable laser or any another object on the scene.

Provided screenshot below is a example of functionality that can be easily implemented in your game.





Parameters:

1. Unlocked Keys Limit - is a keys' count that player must collect in order to unlock some functionality.
2. Keys - Created keys array. Script will keep each key in the array.
3. Hidden GameObject - is responsible for hiding(disable or destroy) gameObject when all keys are collected. Drag & drop necessary gameObject.
4. Delay before disappearance - is responsible for the time during which the generated key will be (hidden) destroyed. During this time you can play the animation, sound, give an achievement, and so on.
5. Register new key - creates new key with specified delay before disappearance.
6. Remove all keys - removes all keys from current Unlock Manager.

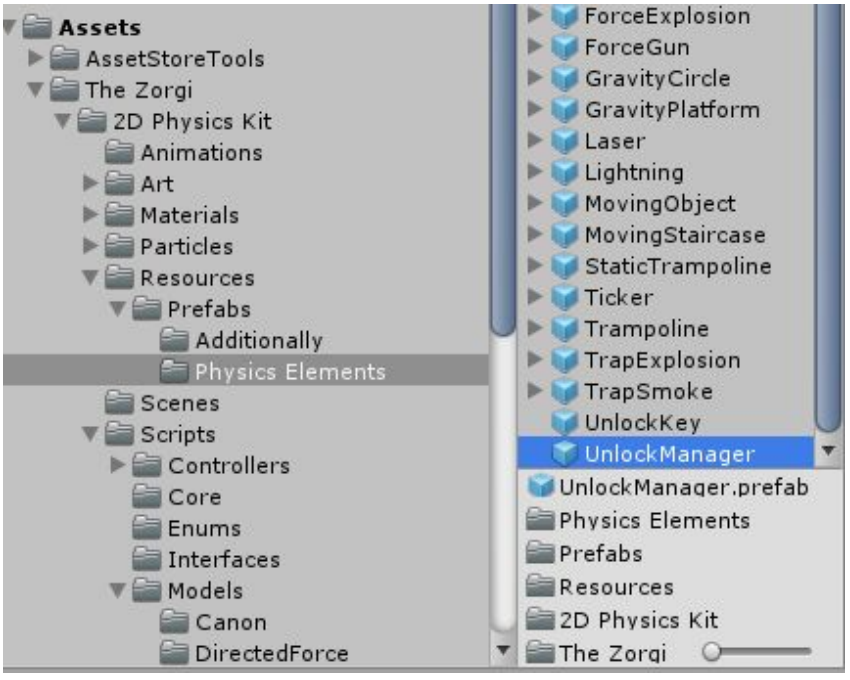
Usage:

Unlock manager very easy to use. In order to start using it you should:

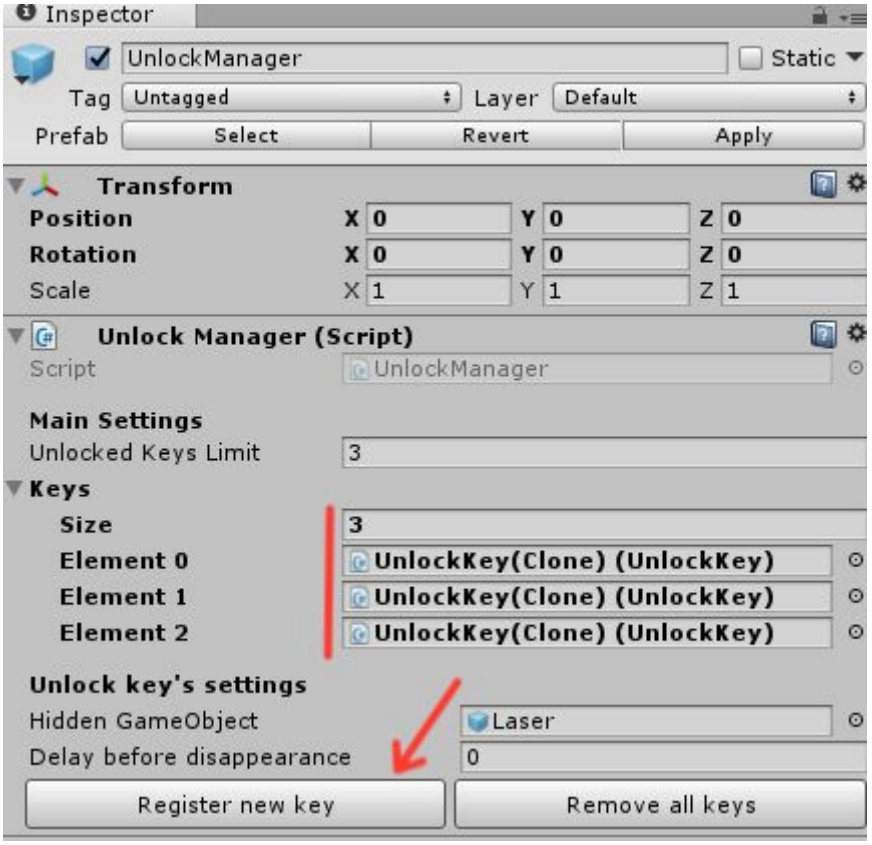
- Specify keys' count limit
- Create necessary keys count. **It must be equals keys' count limit.** In other case you will have an error.
- Point the position out for each key, wherever you want it to be placed

Representation of usage:

Find UnlockManager's prefab in the project hierarchy. Then drag & drop it on your scene wherever you want.



Next step is creating keys:



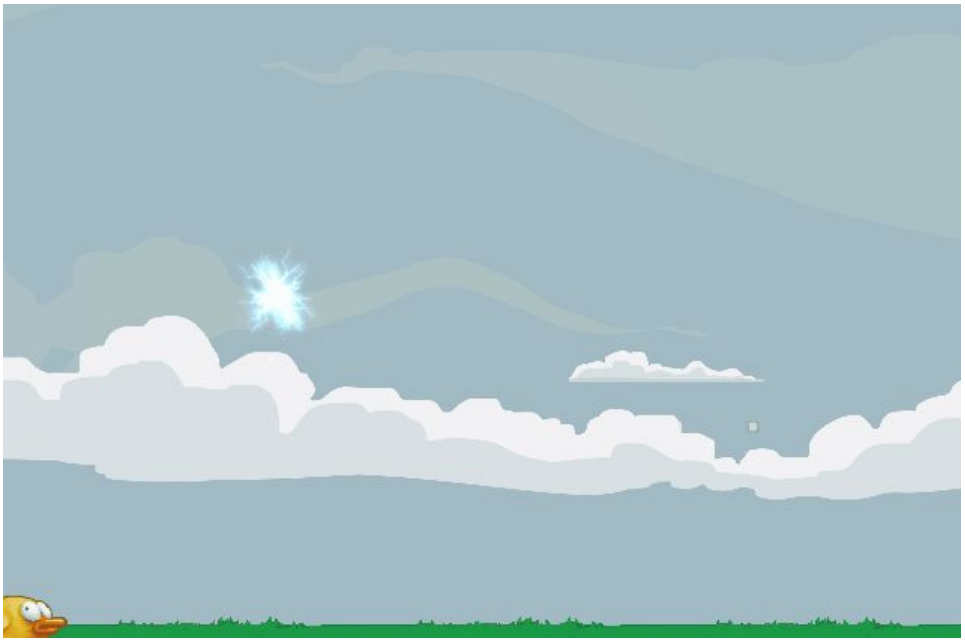
As you can see on screen, We have created **3** keys and set unlocked keys limit **3**. **Keys can lie anywhere on your scene**. Feel free to move it wherever you want. When all keys will be collected GameObject(as example it's a laser) will be hidden(in your case you can destroy object, hide, etc). That's all!

Charge

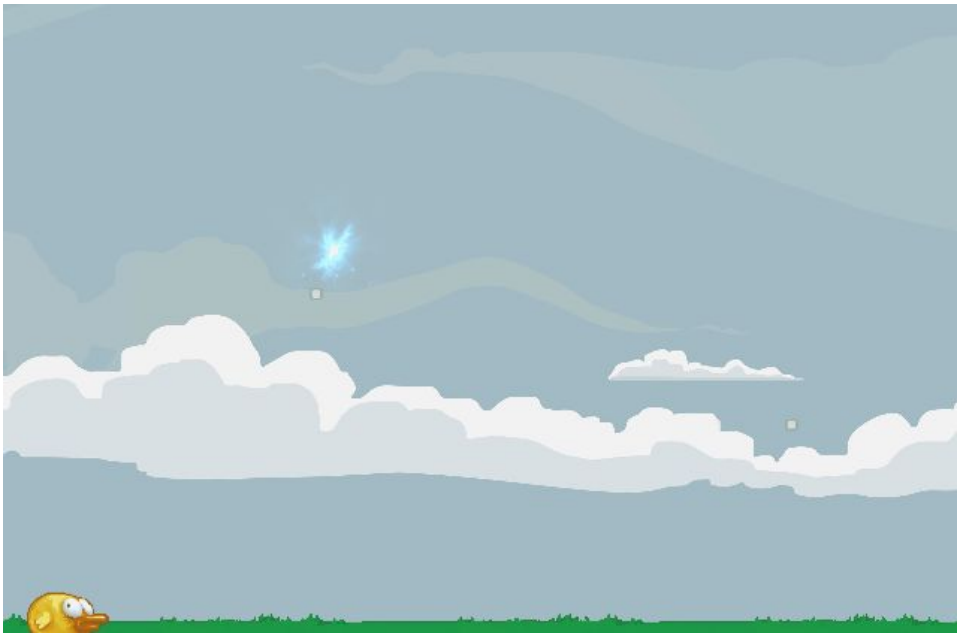
Charge - electric object that moves by setted positions. It has two movement type(*Between to points* and *Circle*).

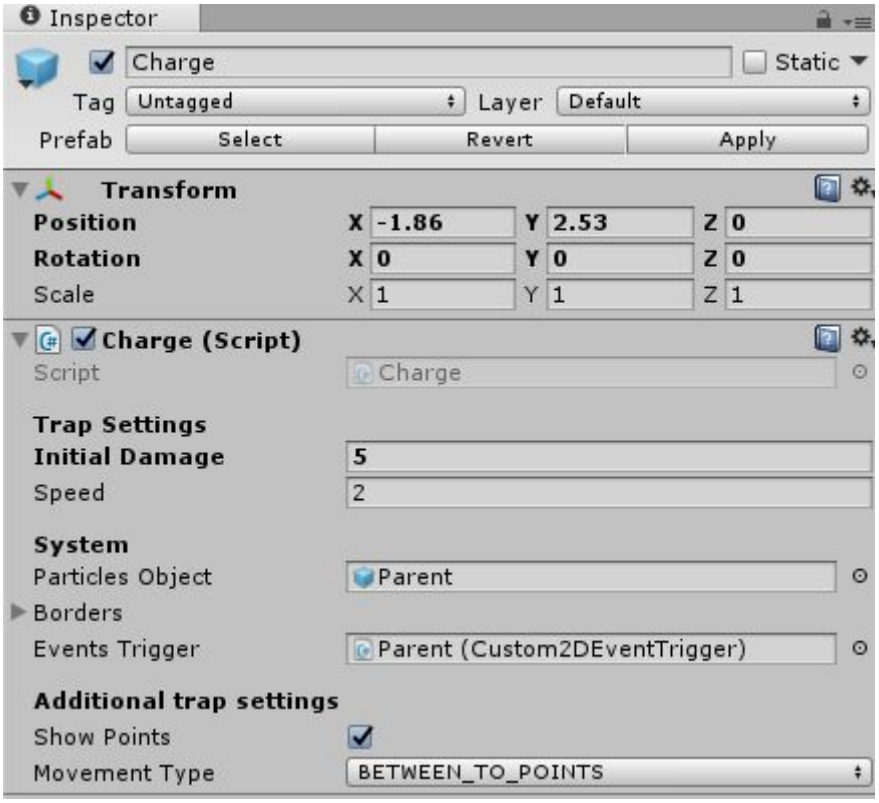
Gif's representation:

Movement type - Between to points:



Movement type - Circle:





Parameters:

1. Initial damage - the damage which player will get after falling into a trap
2. Speed - current speed of the charge

Components:

1. DefaultPos - start position of the Charge.
2. DestinationPos - destination position of charge movement.
3. Parent - particles of the charge object.

Note: If you will choose movement type Circle, charge will move around the circle through the points as borders.

Additional Parameters:

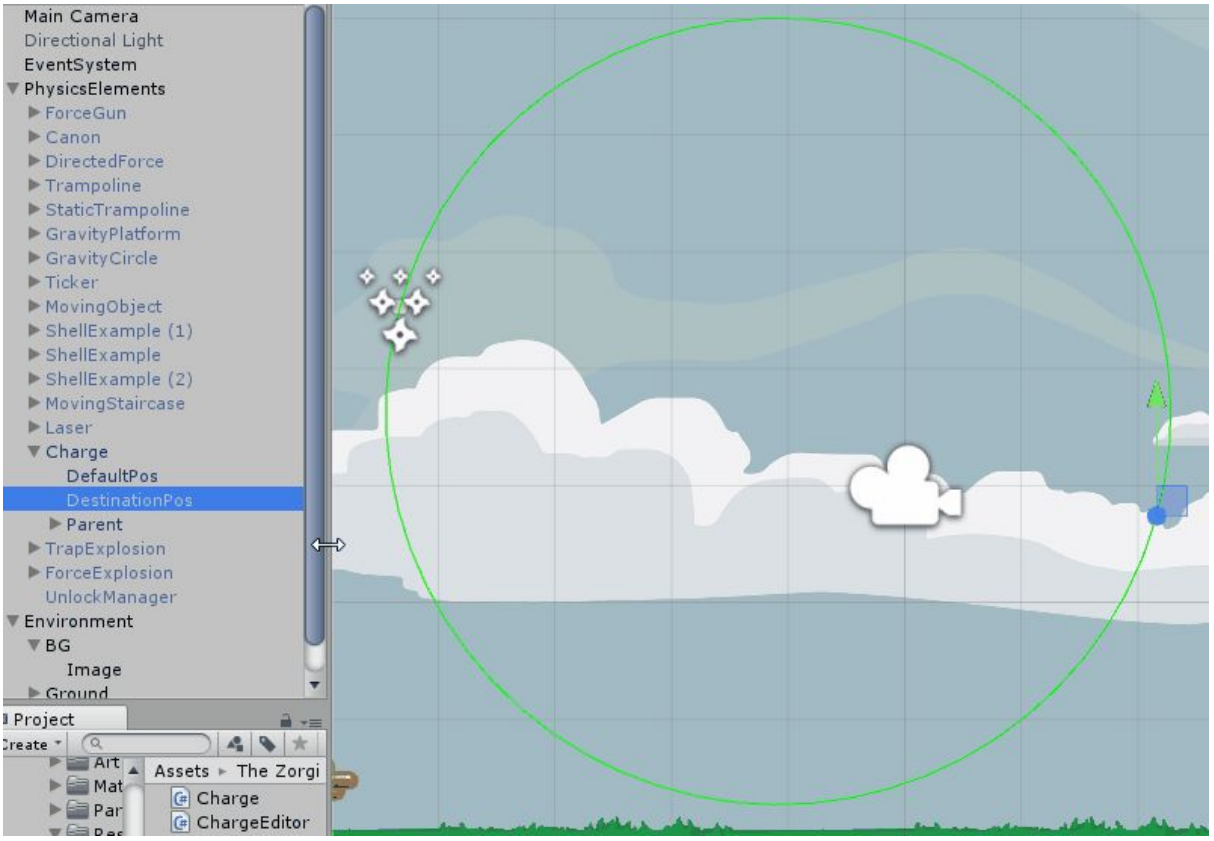
1. Show borders - showing or hiding borders of the charge
2. Movement type - changes movement type of charge
3. Direction type(**is available only when chose movement type Circle**) - changes direction of charge movement

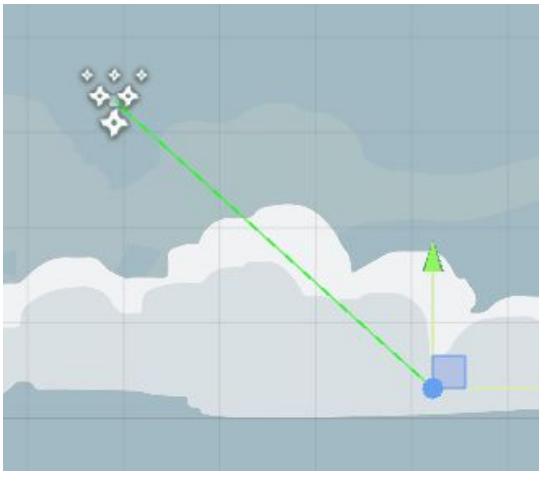
Usage:

Use charge is quite easy!

1. Point the position out where you want to place Charge in your scene
2. Change distance between borders(Use destination border. Look gif below)

Changing distance:

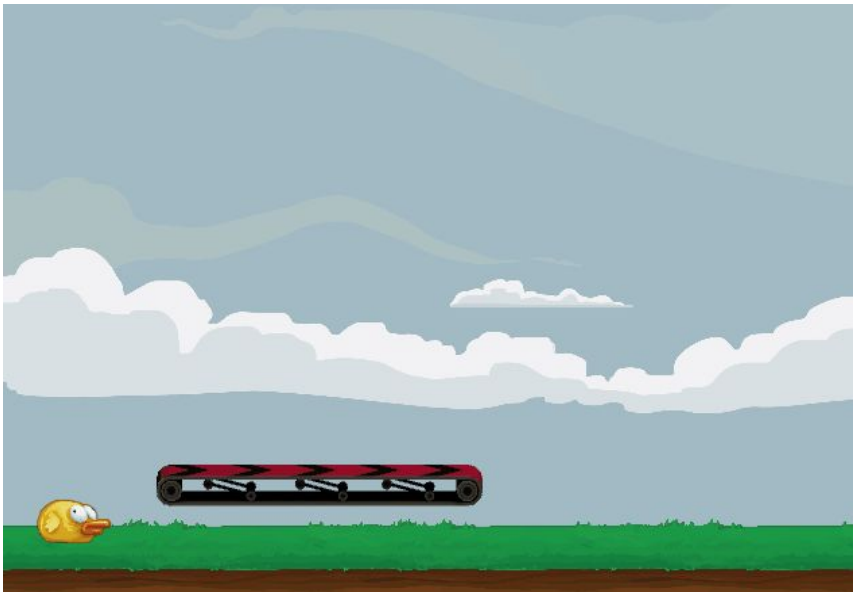


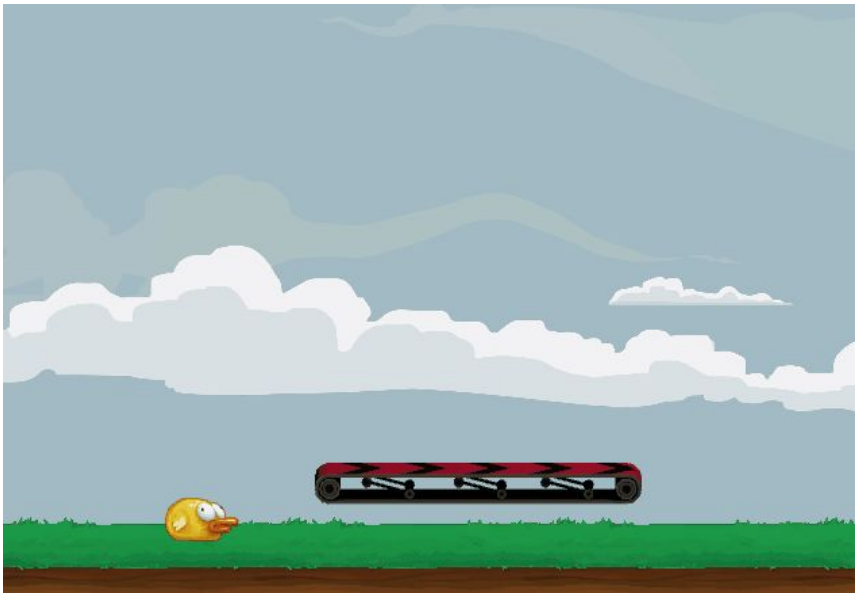


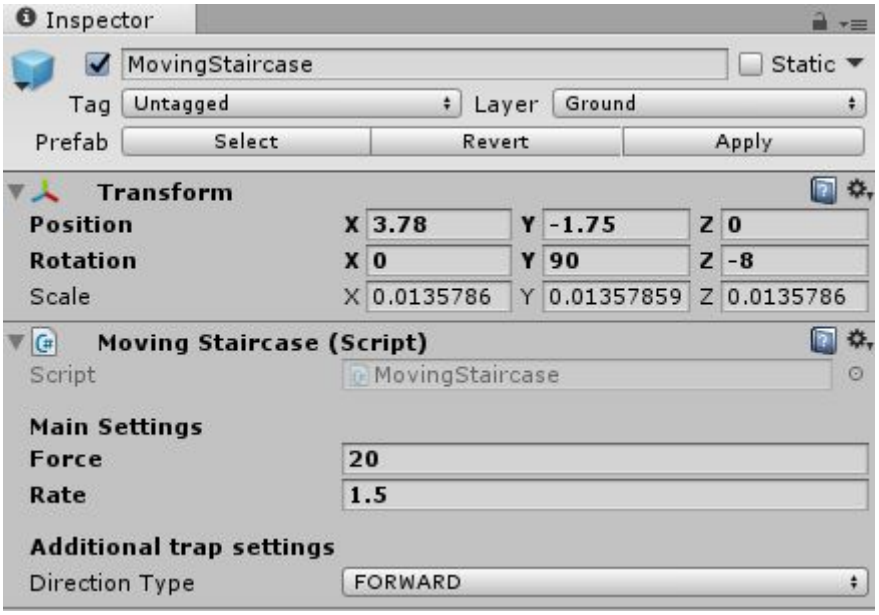
Moving Staircase

Moving Staircase(escalator) - 3D object converted into 2D object. This object gives acceleration to any object that falls on it. You can use it for overcoming obstacles, for crossing distances between objects or like slowdown trap.

Gif's representation:







Parameters:

1. Force - Add force to the object
2. Rate - current rate speed of animation material of the Moving staircase

Components:

1. Rigidbody 2D - required for adding Force to the Object

Additional Parameters:

1. Direction - changes direction where Moving Staircase will add force an object.

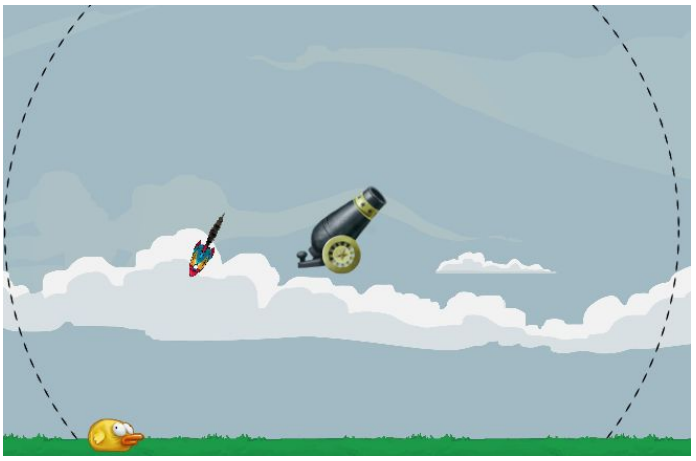
Usage:

Use charge is quite easy!

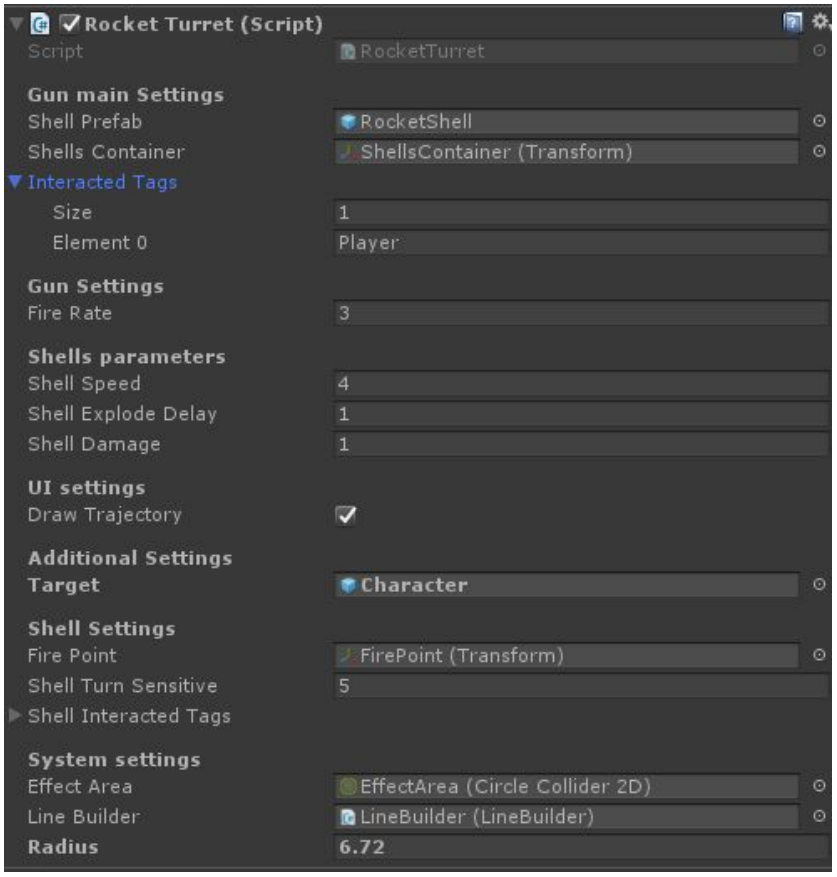
1. Point the position out where you want to place Moving staircase in your scene
2. Rotate if it needed

Update 1.2

Rocket Launcher(turret)



Rocket Launcher - fires shells in player direction. It fires rocket into player since he has been detected in zone attack of rocket launcher.



Parameters:

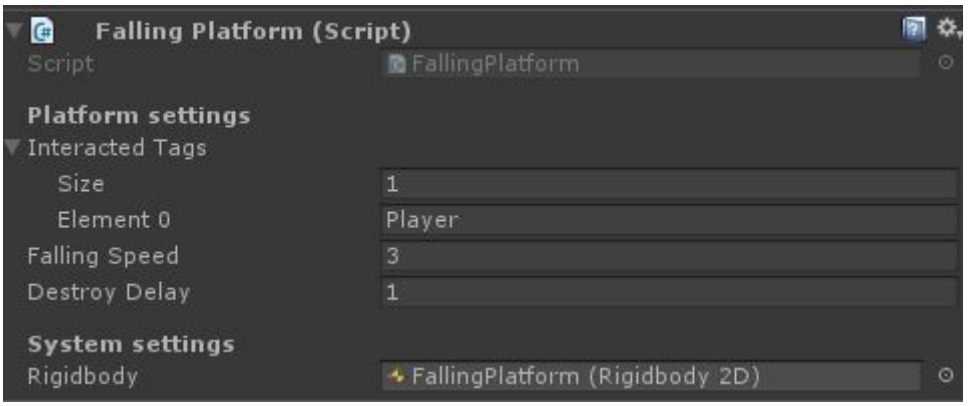
1. *Shell Prefab* - Rocket shells fired by a rocket launcher.
2. *Shells Container* - Gameobject which contains each fired rocket.
3. *Interacted tags* - I have decided to detect player by tag. Array keeps information about tags, due to which rocket launcher decides own targets by raycast.
4. *Fire rate* - Fire rate of the rocket launcher
5. *Shell speed* - Shell speed
6. *Shell Explode Delay* - Delay before destroying rocket shell gameobject
7. *Shell Damage* - Shell damage
8. *Draw Trajectory* - Shows / hides zone attack of the launcher.
9. *Target* - gameobject or place selected as the aim of an attack.
10. *Fire Point* - Position from where the shells fly.
11. *Shell Turn Sensitive* - Shell turn sensitive
12. *Shell Interacted tags* - I have decided to detect interaction target by tag. Array keeps information about tags, due to which rocket shell decides targets by trigger or collision. If it will be empty, shell will be interact with everything.
13. *EffectArea* - collider of the body rocket launcher
14. *Radius* - Zone attack of the rocket launcher

Usage:

1. Point the position out where you want to place Rocket launcher in your scene
2. Rotate fire point if it needed
3. Specify tags for shell and target.

Falling Platform

Falling Platform - falling down after contact with Player tag or another specified name of tag.



Parameters:

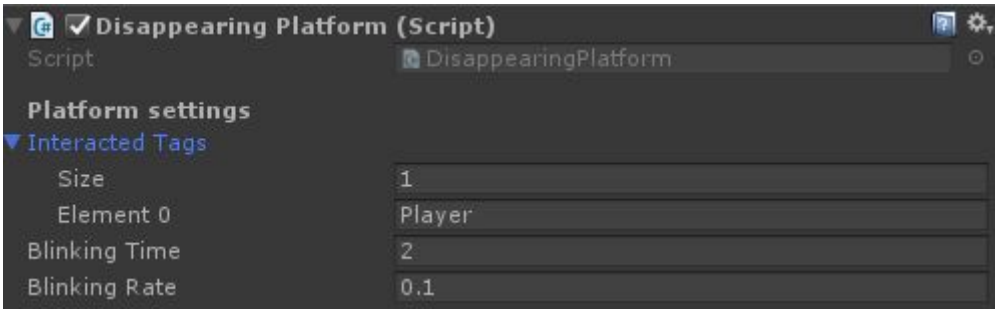
1. *Interacted tags* - Array keeps information about tags, due to which platform decides when starts to fall. If it will be empty, shell will be interact with everything.
2. *Falling speed* - Falling speed
3. *Destroy delay* - Delay to destroy after contact with tags of objects

Usage:

1. Point the position out where you want to place falling platform on your scene
2. Specify tags for collision target.

Disappearing Platform

Disappearing Platform - disappears after contact with Player tag or another specified name of tag.



Parameters:

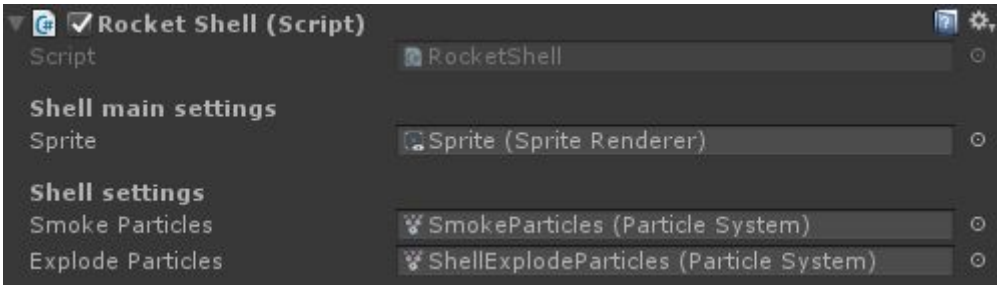
1. *Interacted tags* - Array keeps information about tags, due to which platform decides when starts to fall. If it will be empty, shell will be interact with everything.
2. *Blinking time* - time before destroying platform
3. *Blinking rate* - blinking rate

Usage:

1. Point the position out where you want to place disappearing platform on your scene
2. Specify tags for collision target.

Rocket, Canon, ForceGun shells

Shell of the rocket launcher. Plugin contains 3 types of shells - *rocketShell, canonShell, forceGunShell.* Each of them works with abstract shell which helps contact with specified target by tag. More flexible settings of shell you can find in each gun, to whom do they belong.



Parameters:

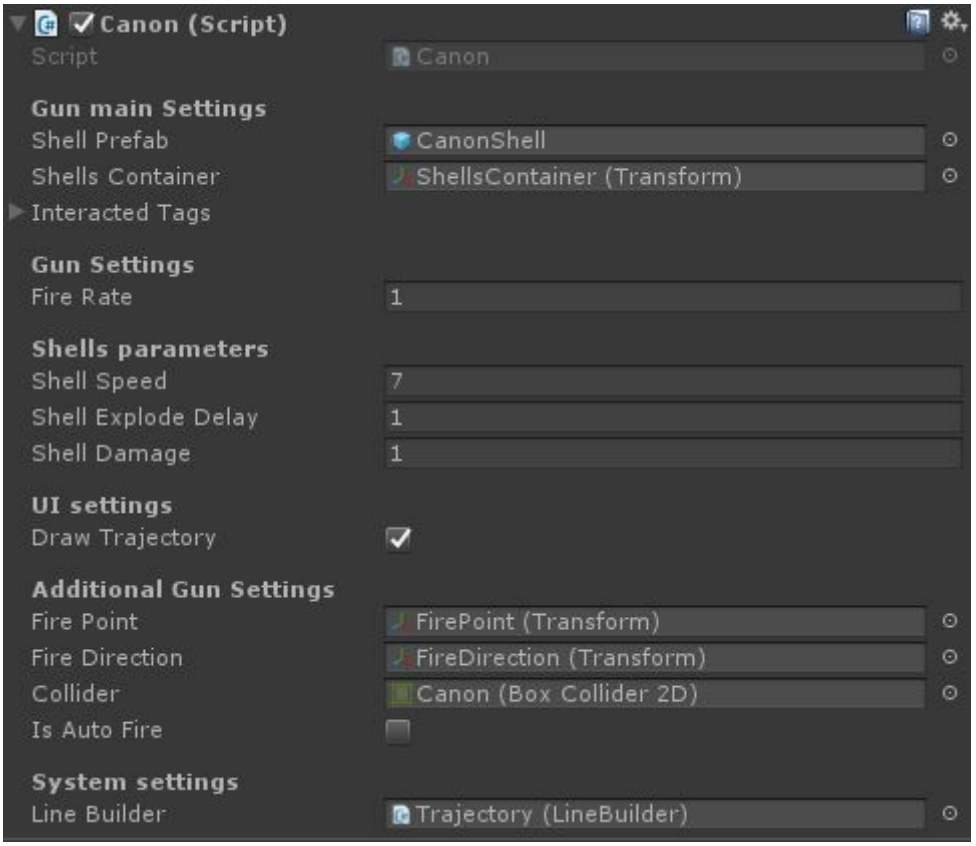
1. *Sprite* - Sprite of shell.
2. *Smoke particles* - Displaying smoke fire of rocket
3. *Explode particles* - Displaying explode particles after colliding with something

Usage:

1. Drag and drop this prefab on Rocket Launcher component as shell prefab.

Canon(Modified since Update 1.2)

Canon - fires shells in right direction. It works in two modes: Manually and Automatically.



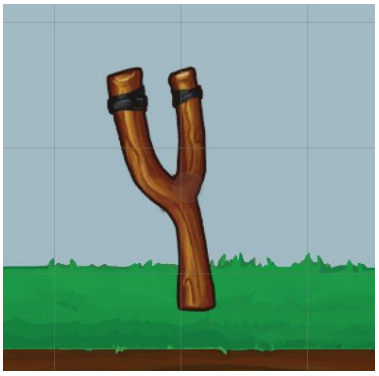
Parameters:

1. *Shell Prefab* - Canon shells fired by a canon.
2. *Shells Container* - Gameobject which contains each fired canon shell.
3. *Interacted tags* - I have decided to detect player by tag. Array keeps information about tags, due to which canon decides own targets by raycast.
4. *Fire rate* - Fire rate of the canon
5. *Shell speed* - Shell speed
6. *Shell Explode Delay* - Delay before destroying canon shell gameobject
7. *Shell Damage* - Shell damage
8. *Draw Trajectory* - Shows / hides zone attack of the canon.
9. *Fire Point* - Position from where the shells fly.
10. *Fire Direction* - Direction of the shells fly.
11. *Collider* - Body of canon
12. *Is Auto Fire* - switch firing modes.

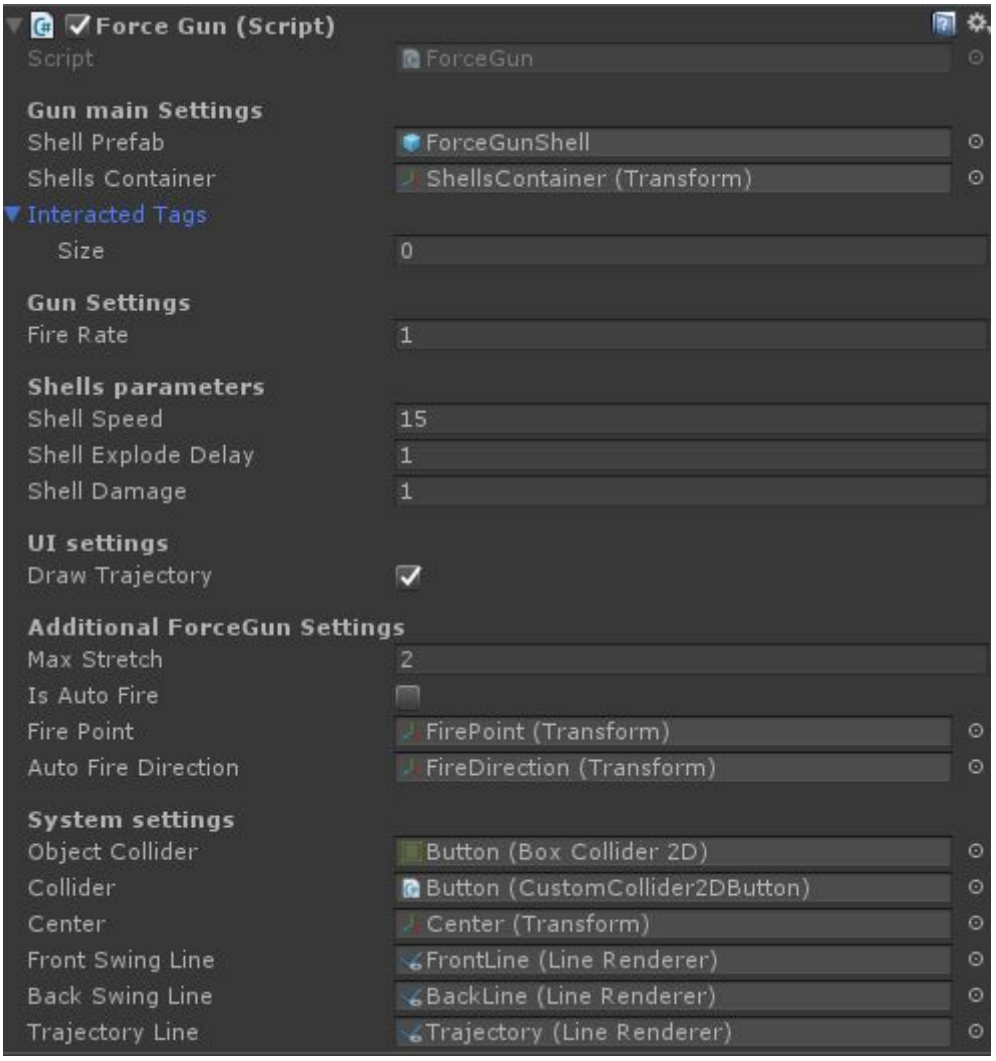
Usage:

1. The same as previous version

Force Gun(Modified since Update 1.2)



Force Gun - object that can throw "bullet" prefab automatically or by user handle. Looks like slingshot from very known game about unhappy birds.



Parameters:

1. *Shell Prefab* - ForceGun shells fired by a canon.
2. *Shells Container* - Gameobject which contains each fired ForceGun shell.
3. *Interacted tags* - I have decided to detect player by tag. Array keeps information about tags, due to which ForceGun decides own targets by raycast.
4. *Fire rate* - Fire rate of the ForceGun
5. *Shell speed* - Shell speed
6. *Shell Explode Delay* - Delay before destroying ForceGun shell gameobject
7. *Shell Damage* - Shell damage
8. *Draw Trajectory* - Shows / hides zone attack of the ForceGun.
9. *Max stretch* - max stretch of sling.
10. *Is Auto Fire* - switch firing modes.
11. *Fire Point* - Position from where the shells fly.
12. *Fire Direction* - Direction of the shells fly.

Usage:

1. The same as previous version

If you have any questions or suggestions, please, don't hesitate to write me - frostinglg@gmail.com.