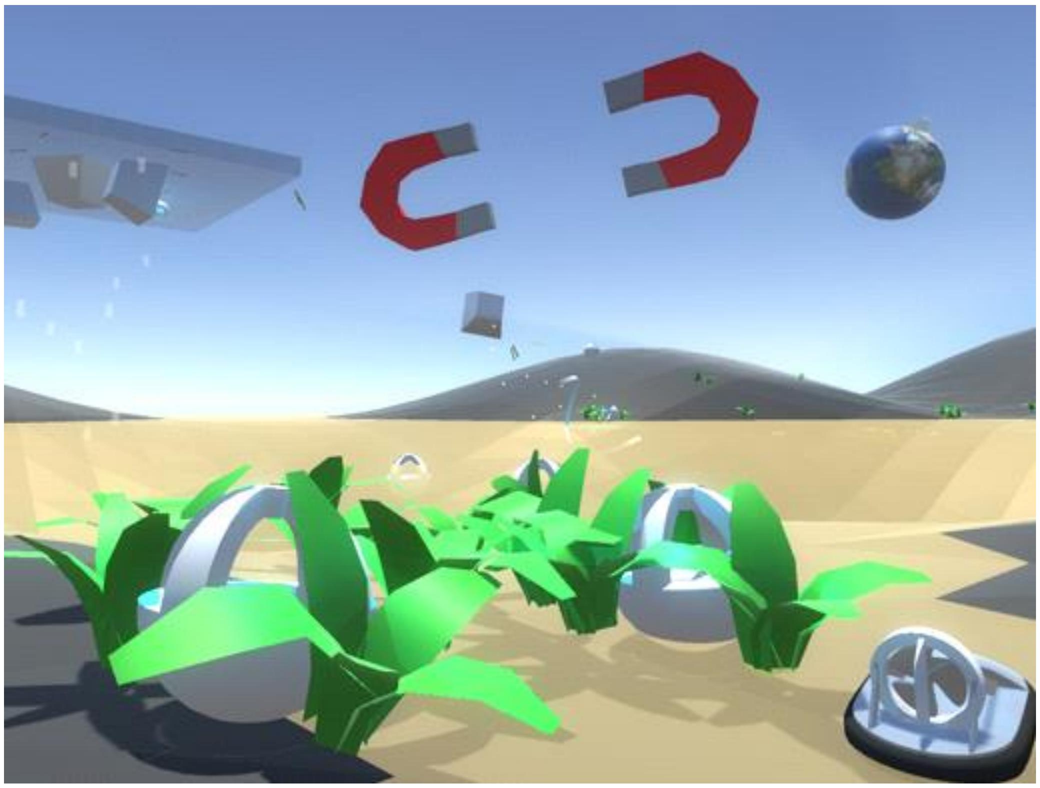
Forces and Physics Lab



You will be able to create forces of nature in your scene, with almost no effort. You can create Gravity fields, Thrusters, Magnets and have fluids and gasses through Soft-collision!

I am ready to give you my best support!

assets@robreijnen.nl

Content:

Quickstart

All Forces & Physics lab features work right out of the box. Go to ***GameObject > Forces & Physics Lab*** and choose to create one of the Objects. Or check the example scenes and see how the included Prefabs are set up.

Features

Each core script can be added as a component to a GameObject independently. And adds several features. They are all located in: ***Forces & Physics Lab > Scripts***

All scripts that share the same base script, do interact with eachother. Plus features that state otherwise (f.e. *Affect all rigidbodies*).

Let me lighten that up a bit: Physics Object is the base for Gravity Object and Directional Gravity. So both child scripts are also treated as a Physics object. A Gravity object interacts with Physics objects, and therefore also interacts with other Gravity Objects. Because they derive from it. This is done because they "share" the same system. It also makes it more accessable for you to build new classes on this. You could create your own child class that derives from Physics Object to inherit it's features and make it "fit in with the system".

Physics Object(base)

GameObject > Forces and Physics Lab > Physics Object > Pick a shape

A Physics object adds several capabilities to your Rigidbody object when using Gravity Objects and Directional Gravity. You can use it to easily change your Rigidbody's center of mass. Align it to a gravity force. And enable Force Processing.

Can Be attracted: Specifies wether this object can be attracted by GravityObjects.

Align to Force: If true, this will rotate itself to point with it's Center of Mass towards the gravity that is attracting it. (For example: This can be used to make a player stand straight up in a spherical world.)

Align Strength: The strength of the aligning force.

Align Limit: How much the objects rotation may differ from being aligned to the force.

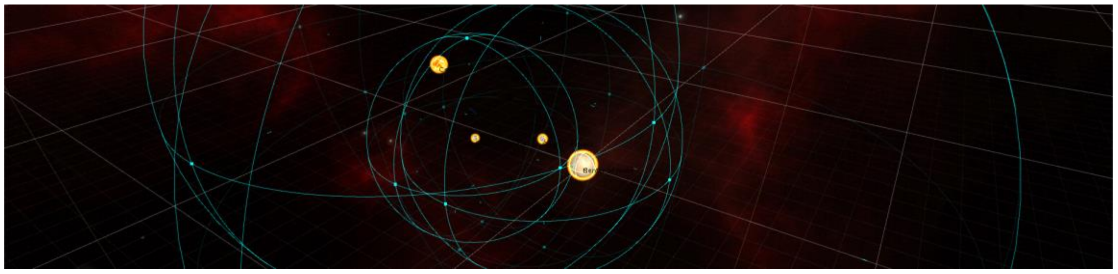
Enable Force Processing: When enabled and multiple forces apply to this object. The object will only apply the highest priority force. And discard the others.

Center of Mass: The relative position of the Center of Mass.

Gravity Object(base : PhysicsObject)

GameObject > Forces and Physics Lab > Forces > Gravity Object

A Physics object but with a gravity force. The center of mass has the most gravitational pull. Optionally, you can make it attract all RigidBodies. All in it's Forcefield get attracted.



Gravity Multiplier: This is multiplied by the RigidBody mass to define how much gravitational pull it has.

Can Be attracted: Specifies wether this object can be attracted by other GravityObjects.

Affect all rigidbodies: Specify wether it affects all rigidbodies. Or only objects with a Physics object component.

Align to Force: If true, this will rotate itself to point with it's Center of Mass towards the gravity that is attracting it. (For example: This can be used to make a player stand straight up in a spherical world.)

Align Strength: The strength of the aligning force.

Align Limit: How much the objects rotation may differ from being aligned to the force.

Disable Rotation Locking on other: If set to true. This gravity field will not Align to force on other objects.

No Decay: Specify wether the gravity field has a decay or not.

Enable Force Processing: When enabled and multiple forces apply to this object. The object will only apply the highest priority force. And discard the others.

Non Alloc: If enabled. The component will create no memory garbage. But a buffer is mandatory.

Non Alloc Buffer: The maximum number of objects that can be affected by this component.

Force Shape: The shape of the force field. All objects in this zone will be attracted.

Field Radius: The maximum radius of the Spherical force field.

Field Size: The size and shape of a Cubic force field.

Be aware that the gravitational decay is not specified by this radius/size, but by distances. In the real world this radius/size would be infinite, but forces will become negligible at longer distances.

Center of Mass: The relative position of the Center of Mass. Objects will get pulled to this point.

Priority: Use with Force processing. Lower priority forces will be discarded when applying force to an object, that has Force processing enabled.

Forcemode: Use ForceMode to specify how to apply a force.

Included layers: Specify wich layers to exclude from the force.

Directional Gravity(base : PhysicsObject)

GameObject > Forces and Physics Lab > Forces > Directional Gravity Object

Pull's GravityObjects towards a direction and has a center of mass. The center of mass has the most gravitational pull. Optionally, you can make it attract all RigidBodies. All in it's Forcefield get attracted.

Gravity Multiplier: This is multiplied by the RigidBody mass to define how much gravitational pull it has.

Direction: The direction to pull Objects towards (Direction will be normalized).

Can Be attracted: Specifies wether this object can be attracted by other GravityObjects.

Affect all rigidbodies: Specify wether it affects all rigidbodies. Or only objects with a Physics object component.

Align to Force: If true, this will rotate itself to point with it's Center of Mass towards the gravity that is attracting it. (For example: This can be used to make a player stand straight up in a spherical world.)

Align Strength: The strength of the aligning force.

Align Limit: How much the objects rotation may differ from being aligned to the force.

Disable Rotation Locking on other: If set to true. This gravity field will not not Align to force Direction on other objects.

No Decay: Specify if the gravity force decays over distance. This is useful if you want an equal directional gravity in a zone.

Enable Force Processing: When enabled and multiple forces apply to this object. The object will only apply the highest priority force. And discard the others.

Non Alloc: If enabled. The component will create no memory garbage. But a buffer is mandatory.

Non Alloc Buffer: The maximum number of objects that can be affected by this component.

Force Shape: The shape of the force field. All objects in this zone will be attracted.

Field Radius: The maximum radius of the Spherical force field.

Field Size: The size and shape of a Cubic force field.

Be aware that the gravitational decay is not specified by this radius/size, but by distances. In the real world this radius/size would be infinite, but forces will become negligible at longer distances.

Center of Mass: The relative position of the Center of Mass. Objects will get pulled to this point.

Priority: Use with Force processing. Lower priority forces will be discarded when applying force to an object, that has Force processing enabled.

Forcemode: Use ForceMode to specify how to apply a force.

Included layers: Specify wich layers to exclude from the force.

Density Object(base)

GameObject > Forces and Physics Lab > Forces > Density Object

A density object component adds a Density value to an object. It interacts with Global Water and Soft Collision Components.

Density: The Density of the object's material.

Soft Collision(base : DensityObject)

GameObject > Forces and Physics Lab > Forces > Soft collision

Suitable for creating particles, fluid, gas or some other soft collision object. The Soft-collision objects will add outward pressure to colliding objects and itself. The soft collision object can not collide with static colliders.

Density: The Density of the object's material.

Non Density Objects: Specify what objects without Density will do when colliding. They can be pressed outward or be ignored by this Soft collision Object. When set to sink, the Softcollision object will move away from the colliding object.

Pressure multiplier: How strongly the object will be pushed out of the density object.   In respect to both object's density.

Pressure Damping: The pressure falloff. The closer an is to being out of the DensityObject's boundary. The more the outward pressure is decreased. This value specifies how much that decrease is applied. A value of 1 to make the outward pressure exactly 0 when the object is extactly on the border.

Disable Density Calculation: If true: every colliding object will be treated as a 'Non Density object'. Even when it has a density component.

Collide with Softcollision: Specify wether it will collide with other soft collision objects.

Forcemode: Use ForceMode to specify how to apply a force.

Included layers: Specify wich layers to exclude from the force.

Global Water(base : DensityObject)

GameObject > Forces and Physics Lab > Forces > Global Water

Global water is used for water with an upward/global gravity. Use a collider that is set to Trigger. A BoxCollider is Recommended. This scripts makes all RigidBodies entering the Trigger zone float upward on it's surface.

If the entering RigidBody's gameObject has a DensityObject component > The water will calculate how the Object will float or sink.

Density: You could set this to 1 (or 0.997 to be precise) if it's going to be just water.

Non density Object: Specify wether the entering gameObject will float or sink, if it has no DensityObject component.

Surface height: The height of the surface of the water. Or in other words: how deep the water is. Measured from the lowest point of the collider (along the y-axis). This can also be edited by dragging the upward arrow in the scene view.

Pressure multiplier: How strongly the object will float towards the desired height. This does not specify how high an object will float. A higher value will make the water lift heavier objects faster, but still in respect to the density. A sinking object will still sink.

Pressure Damping: The pressure falloff. The closer an object is to the water surface. The more the upward pressure is decreased. This value specifies how much that decrease is applied. A value higher than 0 is needed to stabilize objects floating in the water. A value of 1 to make the upward pressure exactly 0 when the object's height is exactly on the water surface.

Minimum Pressure: The minimum density difference between two density object's.

Raycast Pressure: This will apply rotation to the floating objects. Objects will always rotate to lay-flat on the water. You could turn this of if you use (for example) spherical objects, since they don't need to rotate will floating. One raycast per floating object is used.

Forcemode: Use ForceMode to specify how to apply a force.

Included layers: Specify wich layers to exclude from the force.

Magnetic Properties(base)

GameObject > Forces and Physics Lab > Forces > Magnetic Properties

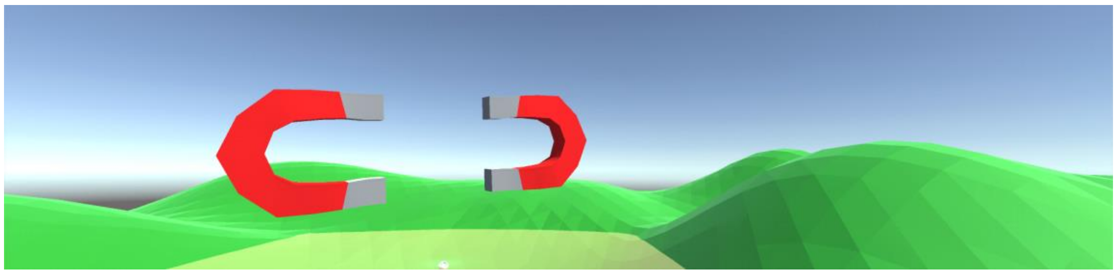
Gets pulled by magnets, but does not affect other objects. Put this on objects that represent metal, to make them magnetic.

Can Be attracted: Specifies wether this object can be attracted by Magnets.

Bipolar Magnet(base : MagneticProperties)

GameObject > Forces and Physics Lab > Forces > Bipolar Magnet

Pull's Magnets towards itself and has two magnetic poles to interact with other Magnet's. The poles have the most gravitational pull. Interacts with other magnets to repel the same poles, and attract the opposites.



Magnetism Multiplier: This is multiplied by the RigidBody mass to define how much magnetic attraction it has.

Can Be attracted: Specifies wether this object can be attracted by other Magnets.

No Decay: Specify wether the magnetic field has a decay or not.

Non Alloc: If enabled. The component will create no memory garbage. But a buffer is mandatory.

Non Alloc Buffer: The maximum number of objects that can be affected by this component.

Field Radius: The maximum radius of the Spherical force field.

Be aware that the magnetic decay is not specified by this radius/size, but by distances. In the real world this radius/size would be infinite, but forces will become negligible at longer distances.

Northern End: Relative position of the Northern magnetic pole.

Southern End: Relative position of the Southern magnetic pole.

Forcemode: Use ForceMode to specify how to apply a force.

Included layers: Specify wich layers to exclude from the force.

Rigidbody Windzone

GameObject > Forces and Physics Lab > Forces > Rigidbody Windzone

Applies a windforce to all RigidBodies. And can be driven by a standard Windzone component to match it's values. It is created to kind of match the build-in windzone.

Master: If a windzone is specified this component will become a slave to the standard windzone. The mode, Main, Turbulence, PulseMagnitude and PulseFrequency will be driven by the standard windzone.

Multiplier: This value is multiplied by the Main value.

Radius: The radius of the windzone.

Mode: Defines the type of wind zone to be used (Spherical or Directional).

Main: The primary wind force.

Turbulence: The turbulence wind force.

PulseMagnitude: Defines how much the wind changes over time.

PulseFrequency: Defines the frequency of the wind changes.

Non Alloc: If enabled. The component will create no memory garbage. But a buffer is mandatory.

Non Alloc Buffer: The maximum number of objects that can be affected by this component.

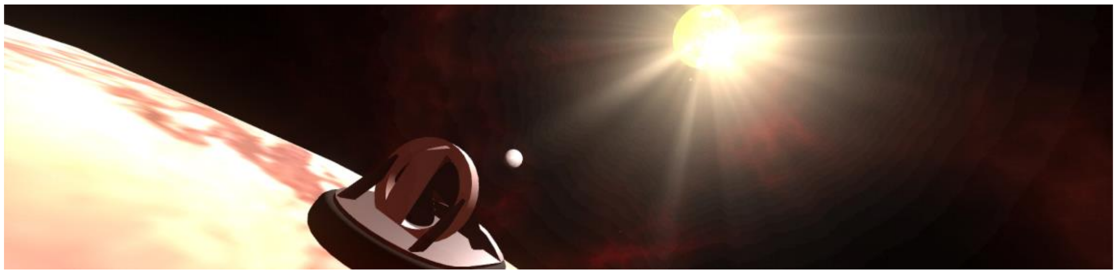
Forcemode: Use ForceMode to specify how to apply a force.

Included layers: Specify wich layers to exclude from the force.

Thruster

GameObject > Forces and Physics Lab > Forces > Thruster

Adds a force to the bound rigidBody. Can also hover above objects. On top of this you can make the thruster apply opposite forces to near Rigidbodies with a Thruster Blastzone.



Mode:

> **Rocket Power:** Adds a force to the bound rigidBody.
>
> **Hover:** Adds an amount of force to the bound rigidBody to make it hover above a surface.

Enabled: Specify wether the Thruster is enabled.

Current Power: The current amount of power in percentages.

Bound RigidBody: The rigidBody to wich the forces of this thruster are applied.

Force: How much force will be applied.
Note: In hover mode this is more like: Specify with how much force you will try to achieve the desired hover height.

Hover Height: How high above a surface should it hover?

Hover Damp: The force falloff. The closer the bound RigidBody is to the desired hover height. The more the force is decreased. This value specifies how much that decrease is applied. A value of 1 to make the force exactly 0 when the object is extactly on the hover height.

Forcemode: Use ForceMode to specify how to apply a force.

Thruster Blastzone

GameObject > Forces and Physics Lab > Forces > Thruster

Adds an opposite force to objects inside the blast radius of it's Thruster component. The blast decays over distance.

Blast Multiplier: The thruster component's original Force is multiplied by this.

Force Limit: The maximum strength of a force. This could provide you some stability, if necessary.

Wobble: Adds a fast wobble to the blast force. For a more windy appearance on objects.

> **Wobble Turbulence:** The strength of the wobble
>
> **Wobble Frequency:** How fast the wobble is.

Blast Radius: The maximum blast radius. Does not affect the Blast decay. Can be changed in scene view by dragging the handles of the blast radius sphere.

FPL namespace

There is also the PFL_Forces namespace that contains several classes to create forces, that are automatically calculated, can be saved and passed around. Before executing them. These class methods can be used in your custom scripts. By typing: `using FPL_Forces;` on top.

A simple example on creating your own gravity script without too much hassle:

```
using UnityEngine;
using FPL_Forces;

public class SuperSimpleGravity : MonoBehaviour {

//Specify these in the inspector
public Rigidbody rigidbodyself;
public Rigidbody rigidbodyother;

//A variable to store the force in and all it's properties.
GravityForce force = new GravityForce();

//This scripts makes one rigidbody attract another with gravitational forces.

    void FixedUpdate ()
         {

         force = new GravityForce();

         //Calculate the gravity that should be applied when interacting between these two rigidbodies.
         force.CalculateGravity(rigidbodyself, rigidbodyother, 1);

         //Apply the force
         force.AddGravity(rigidbodyother, ForceMode.Acceleration);

         }

}
```

Below are all the classes and method for this namespace described

class **GravityForce**

You need two rigidbody objects to calculate gravity between them

Public Methods:

CalculateGravity

```
public void CalculateGravity(Rigidbody rbSelf, Rigidbody rbOther, float GravityMultiplier =
1f, bool NoDecay = false,Vector3 dir = default(Vector3))
```

Calls all methods needed to calculate the force. If direction is left blank it will attract towards itself.

AddGravity

```
public void AddGravity(Rigidbody rb, ForceMode Forcemode = ForceMode.Acceleration)
```

Adds the calculated gravity to the given rigidbody.

AddRotationalLock

```
public void AddRotationalLock(Rigidbody rb, ForceMode Forcemode = ForceMode.Acceleration,
Vector3 OtherCenterOfMass = default(Vector3),float Strength = 1f, float Limit = 0f)
```

Rotates the other objects center of mass towards itself.

ClearSettings

```
public void ClearSettings()
```

Reset the forces calculated values

class **MagneticForce**

Needs two rigidbodies that each have two magnetic-pole positions relative to itself.

Public Methods:

CalculateMagnetism

```
public void CalculateMagnetism(float MagneticMultiplier, Rigidbody rbSelf, Rigidbody rbOther,
bool NoDecay = false, Vector3 SelfNorth = default(Vector3), Vector3 OtherNorth =
default(Vector3), Vector3 SelfSouth = default(Vector3), Vector3 OtherSouth = default(Vector3))
```

Calls all methods needed to calculate the force.

AddMagneticForce

```
public void AddMagneticForce(Rigidbody rb, ForceMode Forcemode = ForceMode.Acceleration)
```

Adds the calculated force to the given rigidbody.

ClearSettings

```
public void ClearSettings()
```

Reset the forces calculated values

class **DensityForce**

Gets the density of two rigidbodies. To calculate if there is any pressure between them, when colliding. If the first rigidbody has a higher density then it will press the second rigidbody outward. The pressure will lessen as it is closer to the edge of the collider. In addition you can specify what happens without setting density values.

Public Methods:

CalculatePressure

```
public void CalculatePressure(Rigidbody rbSelf, Rigidbody rbOther, Collider coll, float
PressureMultiplier = 1f,  float Density1 = -1f,float Density2 = -1f, bool F = true,  Vector3
HighestDensityPoint = default(Vector3), float PressureDamp = 1f, float minimumPressure = 0f)
```

Calls all methods needed to calculate the force.

CalculatePressureGlobal

```
public void CalculatePressureGlobal(Rigidbody rbSelf, Rigidbody rbOther, Collider coll, float
PressureMultiplier = 1f, float Density1 = -1f, float Density2 = -1f, bool F = true, float
PressureDamp = 1f, float minimumPressure = 0f, float SurfaceHeight = 10f)
```

Calls all methods needed to calculate the force. An adaption of the previous method used for global water. (That will push stuff upward specifically.) This will also set StaticUpward to true.

AddPressure

```
public void AddPressure(Rigidbody rbOther, Rigidbody rbSelf, ForceMode Forcemode)
```

Adds the calculated force to the given rigidbody. Or to itself, depending on the Float value.

AddRotatingPressure

```
public void AddRotatingPressure(Rigidbody rbOther)
```

Only works with StaticUpward set to true. Gets the deepest vertices of the rigidbody en applies pressure from there. Floating stuff will rotate to lay flat on the surface.

ClearSettings

```
public void ClearSettings()
```

Reset the forces calculated values