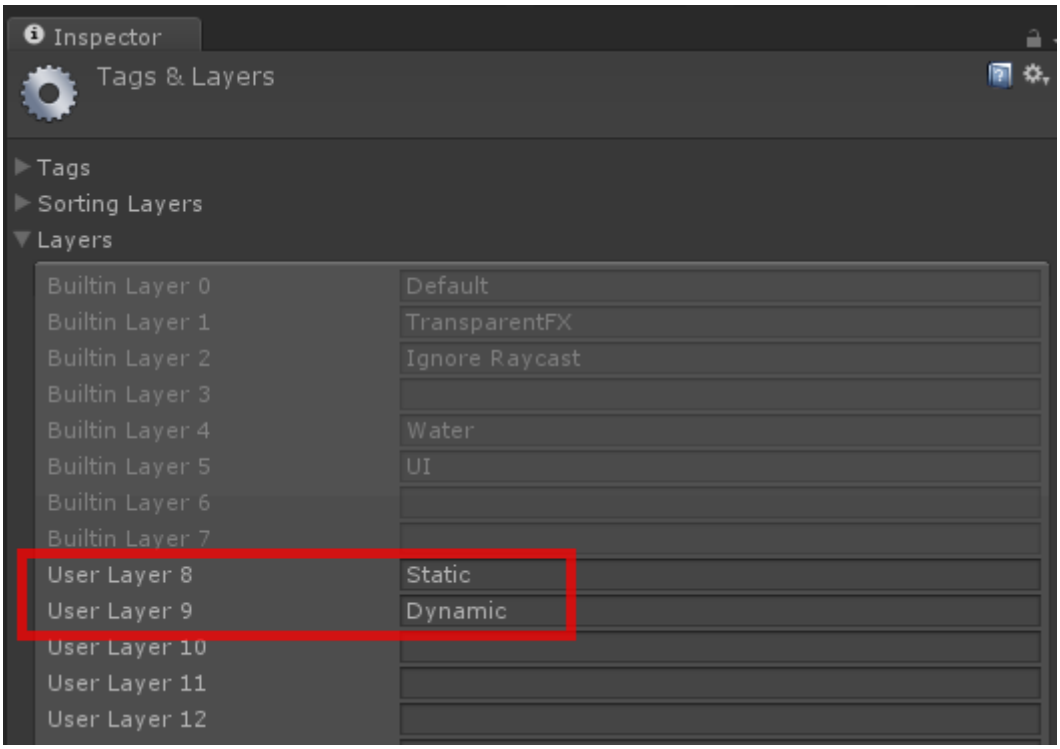
The Force Engine 0.06 Documentation

1 Project Setup

In Project Settings, script execution order is important. TFEngine and TFGUI should execute before other scripts.

Two layers should be reserved by TFEngine, Static and Dynamic, and should preferably be named accordingly:



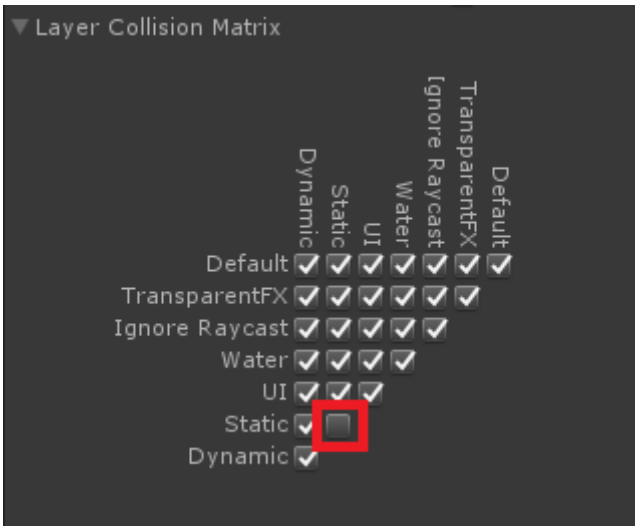
TFColliders with no TFRigidbody automatically set the layer to Static (ground, static scenery).

TFColliders with TFRigidbody automatically set the layer to Dynamic.

If TFEngine handles auto disabling of bodies, then TFRigidbodies below their given auto disabling velocities automatically switch from Dynamic to Static layers when falling asleep, then automatically switch from Static to Dynamic layers when awaken.

The indices of the Static and Dynamic layers are parameters of the TFEngine, default values are 8 and 9.

As an optimisation technique, we don't want to handle collision between static objects; therefore, in the collision matrix of Physics Settings, we untick collision between Static and Static objects.



2 Scene Setup

The scene should contain a GameObject with TFEngine and TFSurfaceParameters scripts.

For stability, it is recommended not to set friction to infinite value.

The following demo scenes are provided:

- **Demo scenes**: self-explanatory
- **HelloWorld**: empty scene with just a floor, TFEngine, TFSurfaceParameters, Hand, to start playing with the engine. The user can drag and drop in this scene prefabs from the Prefabs folder
- **Profile**: serves as a scene for performance profiling

Most demo scenes now have a Physics Hand, a script to drag objects around using the mouse.

3 GameObject Setup

In order for a GameObject to be part of the simulation, the following behaviours should be attached:

3.1 UnityEngine Collider and Rigidbody

These are used for first pass collision detection, to detect that two GameObjects are potentially colliding.

TFCollider will automatically set the Collider to trigger and the Rigidbody to kinematics.

3.2 TFResetScale

The engine partly supports scaling and nesting; this allows for example easy creation and attaching of boxes of various sizes.

This behaviour will bake the scale into the MeshFilter mesh vertices and into the Collider, then set the scale of the GameObject to {1, 1, 1}.

3.3 MeshFilter and MeshRenderer

Used for rendering

3.4 TFCollider

Similar to UnityEngine Collider. This is used to generate detailed collision information.

3.5 TFRigidbody

Similar to UnityEngine Rigidbody. Control of an object's position through TF physics simulation.

3.6 TFMassBehaviour

This behaviour is used to set the mass and moment of inertia of TFRigidbody, according to the TFCollider shape and either final mass or density.

3.7 ShowAutodisabled

This behaviour is used to change the colour of a GameObject, according to if the associated TFRigidbody is sleeping or not. Sleeping is an optimisation technique where bodies at rest are no longer updated.

4 TFTrimesh

It is a triangle mesh collider that uses TFTrimeshData as data.

TFTrimeshData contains all triangles information in an optimised way. It is a ScriptableObject (belongs to the Unity project, not the scene) and can be used by many instances of TFTrimesh.

Current optimisation is the use of an octree, where the root is level 0

TFTrimeshData can be constructed from a Mesh. A lot of the processing is done offline.

1. It is first created using **Tools/The Force Engine/Create TFTrimeshData**,
2. Assign a Mesh
3. Set Min Level and Max Level of recursion in the octree
4. Press **Create from Mesh**

Then you can assign the TFTrimeshData to a TFTrimesh in the scene.

5 TFTerrain

It is a terrain collider that uses TFTerrainData as data.

TFTerrainData contains all triangles information in an optimised way. It is a ScriptableObject (belongs to the Unity project, not the scene) and can be used by many instances of TFTerrain.

TFTrimeshData can be constructed from a Texture.

1. It is first created using **Tools/The Force Engine/Create TFTerrainData**,
2. Assign a Texture
3. Press **Create from Texture** (non-welded vertices) or **Create from Texture Smooth**

Then you can assign the TFTerrainData to a TFTerrain in the scene.

6 What is new

6.1 In Version 0.05 ?

To remove any confusion, focus on new features and allow the engine to be compatible with most (all?) platforms, The Force Engine no longer acts as a wrapper over native Open Dynamic Engine

library. Wrapping is still possible in The Force Engine PRO edition. Native code tends to run approximately 2.5 times faster than current managed code implementation.

Angles in TFJointLimitMotor are now in degrees.

Currently supported colliders are:

- Box
- Capsule (fake, made of spheres)
- Compound (one body, many colliders)
- Plane
- Ray
- Sphere
- Terrain
- Trimesh
- Wheel (kind of torus)

Currently supported joints are:

- Ball
- Contact
- Fixed
- Hinge
- Hinge2
- Slider
- Universal (angular motor)

6.2 In Version 0.06 ?

- Inverting bodies for most joints; now body 0 can be null for the joint to be attached to static environment for easier scene creation.
- New TFRigidbody.OnCollision event
- New TFRigidbody.linearDamping parameter
- New TFDistance joint

7 Stability

To improve the stability of the simulation:

- Decrease fixed time steps
- Increase number of iterations of the solver
- Avoid infinite friction in TFSurfaceParameters

8 Performance

Performance has improved and memory leakage has reduced in **The Force Engine 0.05**.

To measure performance, it is recommended to build a Standalone app.

9 How to create a new TFCollider (The Force Engine, The Force Engine PRO)

To create a new TFCollider, you should derive from TFCollider.

Methods worth overriding:

9.1 InitAfterResetScales()

Called after the scale has been reset. Good time to save the final world size of the collider and use it to adjust the mass of the attached TFRigidbody.

9.2 Bool CanHandleCollision(TFCollider other)

Managed implementation only.

This method should return true if this TFCollider knows how to handle collision against other TFCollider. For collision to take place between two TFColliders, at least one TFCollider should know how to collide against the other one.

9.3 Collide(TFCollider other, List<TFContactGeom> contactGeoms)

Managed Implementation only.

This method should generate detailed contact information for collision with other TFCollider.

10 How to create a new TFJoint (The Force Engine, The Force Engine PRO)

To create a new TFJoint, you should derive from TFJoint.

Methods to override:

10.1 InitAfterResetScales()

Called after the scale has been reset. Good time to save anchor positions.

10.2 getInfo1(ref Info1 info), getInfo2(ArraySegment<Constraint> J)

Please have a look at article *How to make new joints in ODE* from Russell Smith, 2002.

http://www.ode.org/joints.pdf

11 How to translate a C++ file into C#

1. Change your source file extension from cpp to cs
2. Insert the file into your Unity project or Visual Studio solution
3. Fix errors
4. Repeat previous step until all errors fixed
5. Done !

12 Copyrights

13 Support

support@jstarlab.com

In memory of Yves Chaperot, 1925-2017