Physics based animations

- Julien Vulliet (Vux)
- Natan Sinigaglia (Dottore)

Physics based animations

- Introduction
- Terminology
- Box2d Introduction
- Box2d in Depth
- Pratical exercises

Physics based animations

Introduction

Physics based animations

Introduction

Why using a physics engine?

- Realistic Motion
- Objects Interaction (Collisions)
- Efficient handling of Multiple Objects
- Create constraints between Objects

Physics based animations

Terminology

- Time steps
- ABB,OBB
- Shape,Body,Joint
- Broadphase,Narrowphase
- Sleeping Bodies
- Tunneling, CCD, TOI

Physics based animations

Box2d

http://www.box2d.org

- Box2d is an open source 2D physics engine, primarily designed for games.
- Written by Erin Catto (physics developper at Blizzard)
- Licensed under zlib: http://en.wikipedia.org/wiki/Zlib_License
- Rigid body engine. Object geometry can't be altered.
- Contains most modern physics engine features (Broadphase,CCD,Compound Bodies,Collision filtering,Joints.
- Initially written in C++, but ports are available in Flash,Java,Objective C, C#...

Physics based animations

Box2d for VVVV

- Box2d plugin is a Wrapper against Box2d (c++ version). It's not a port.
- For license, go to plugins/licenses/license-box2d.txt
- Uses a top-bottom/top-bottom approach for object creation/interaction
- Wraps pretty much all box2d features (apart the few which requires code)

IMPORTANT NOTE:

As box2d (and any physics engine) will struggle with small values, it is really not recommended to use objects of an area of less than 0.1

This will cause some errors (due to floating points), with very low inertia for example (no restitution)

Good practice is to have a projection with a uniform scale of 0.1 so we can used decently sized objects.

Physics based animations

Box2d in depth

Physics based animations

World

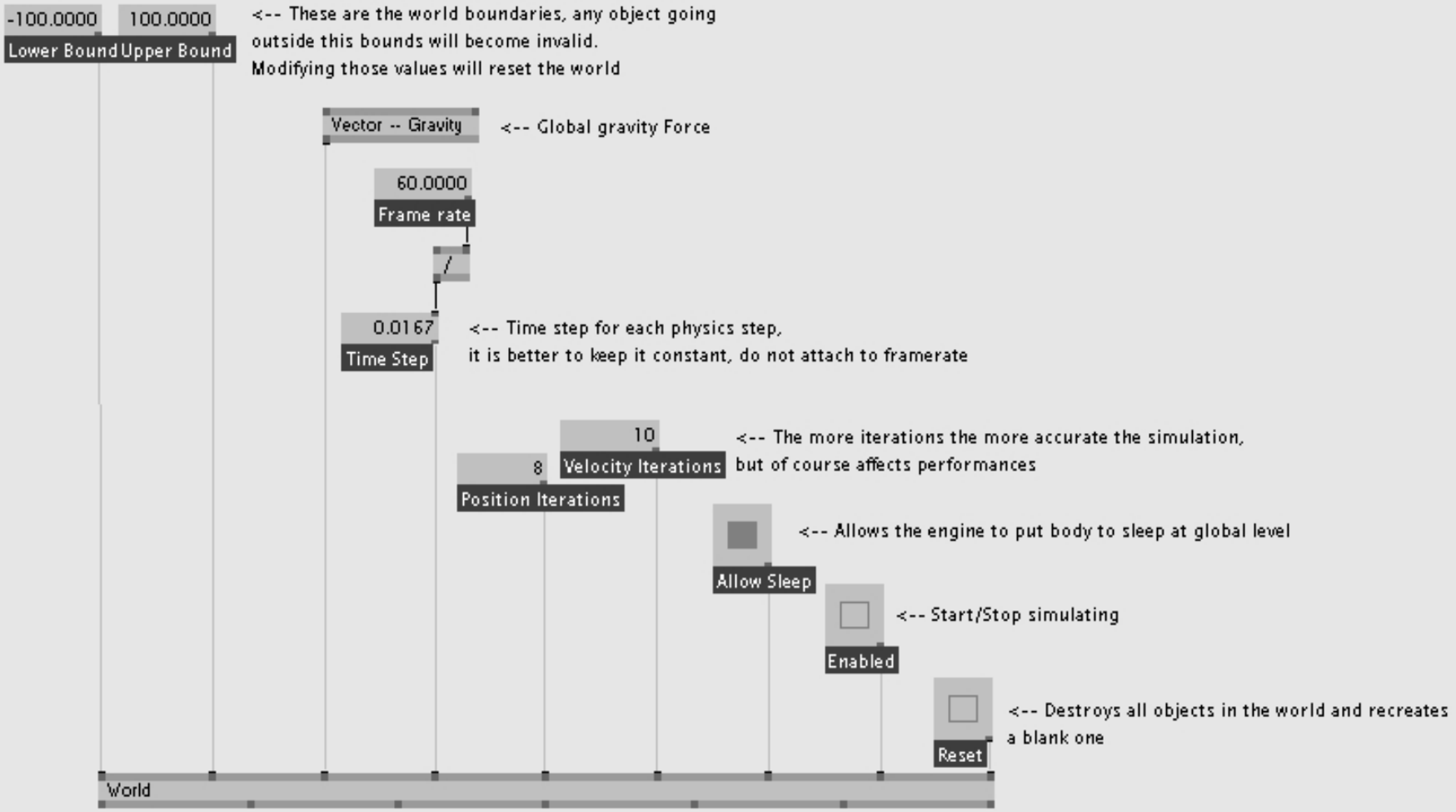
This is the core object for box2d nodes.

It handles:

- Physics simulation
- Object management (Shape updates/Destruction)
- Collision checks
- Prepare objects so their data can be retrieved.

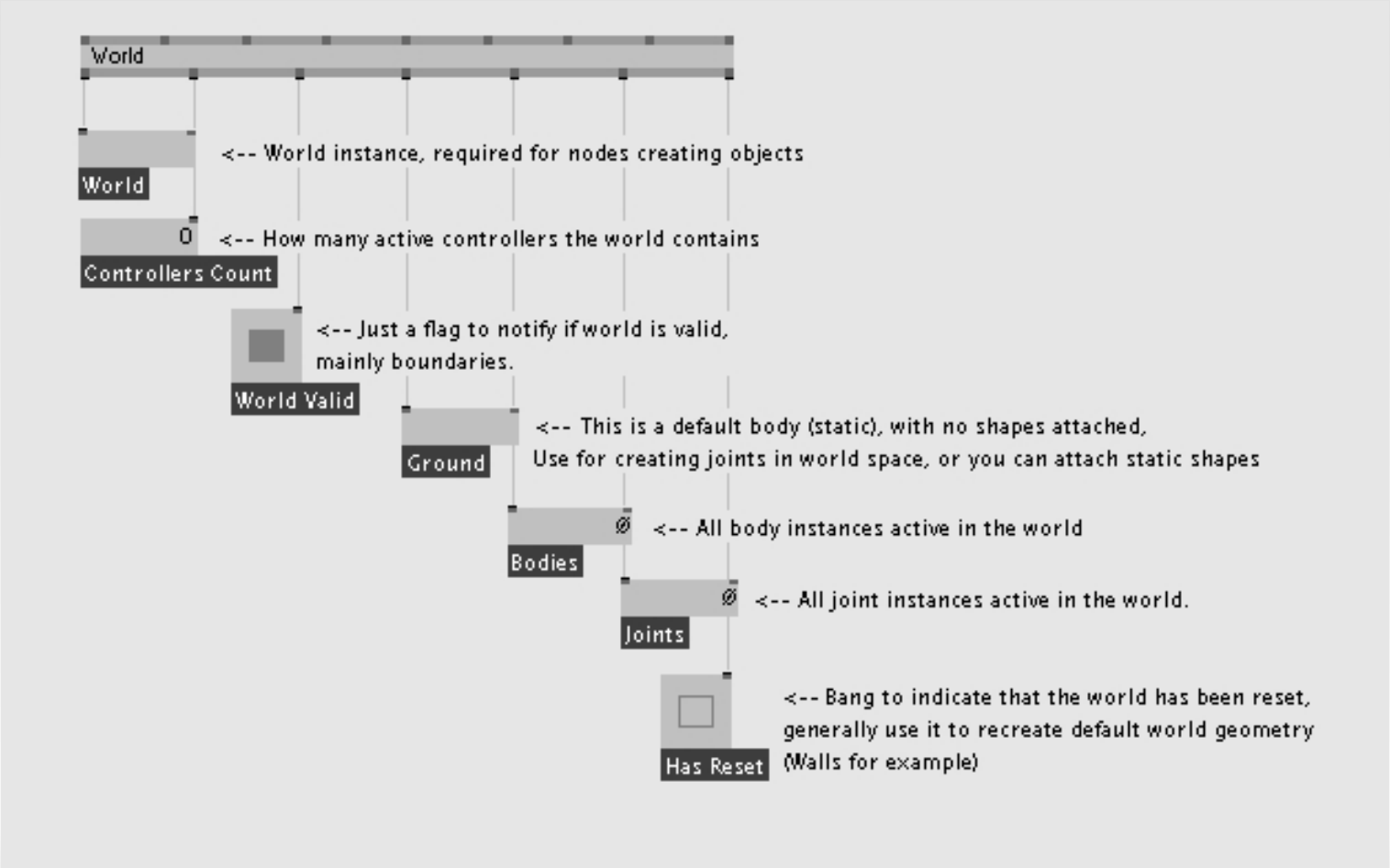
Physics based animations

World



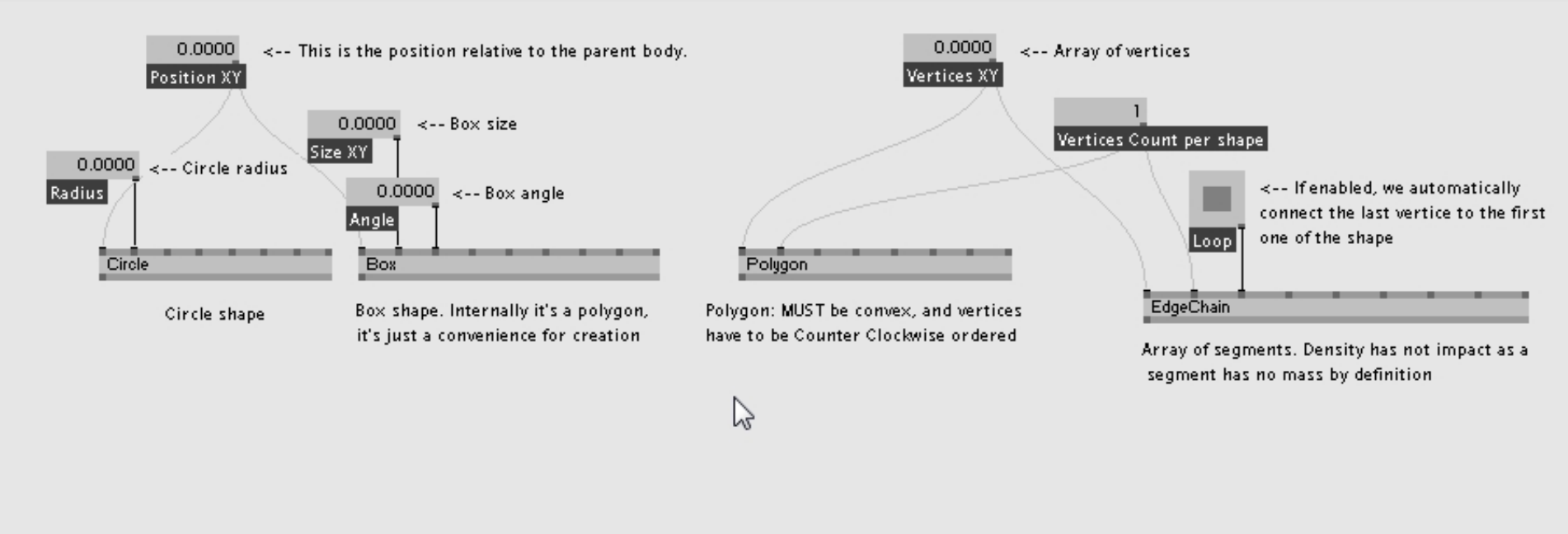
Physics based animations

World



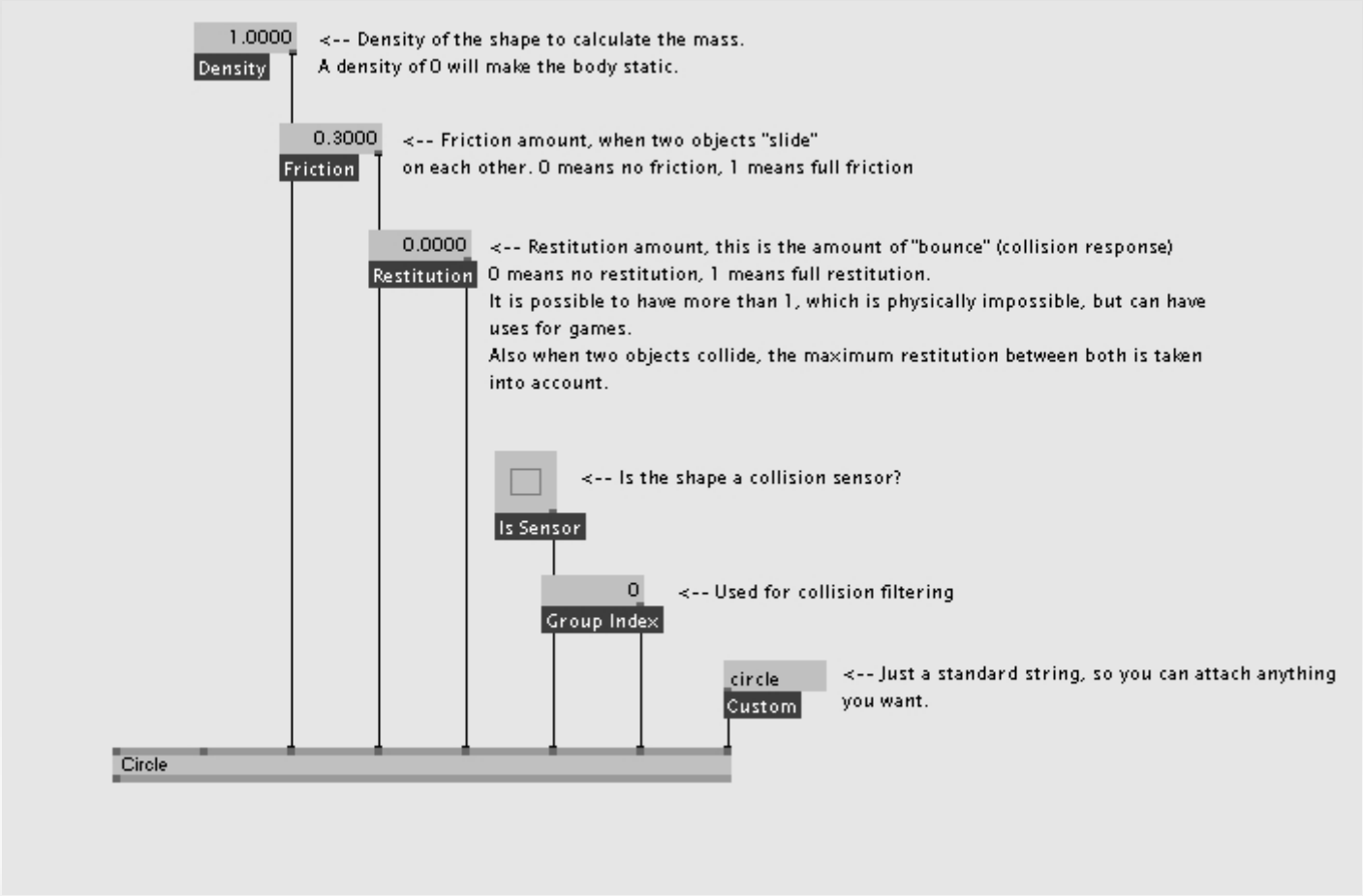
Physics based animations

Primitives



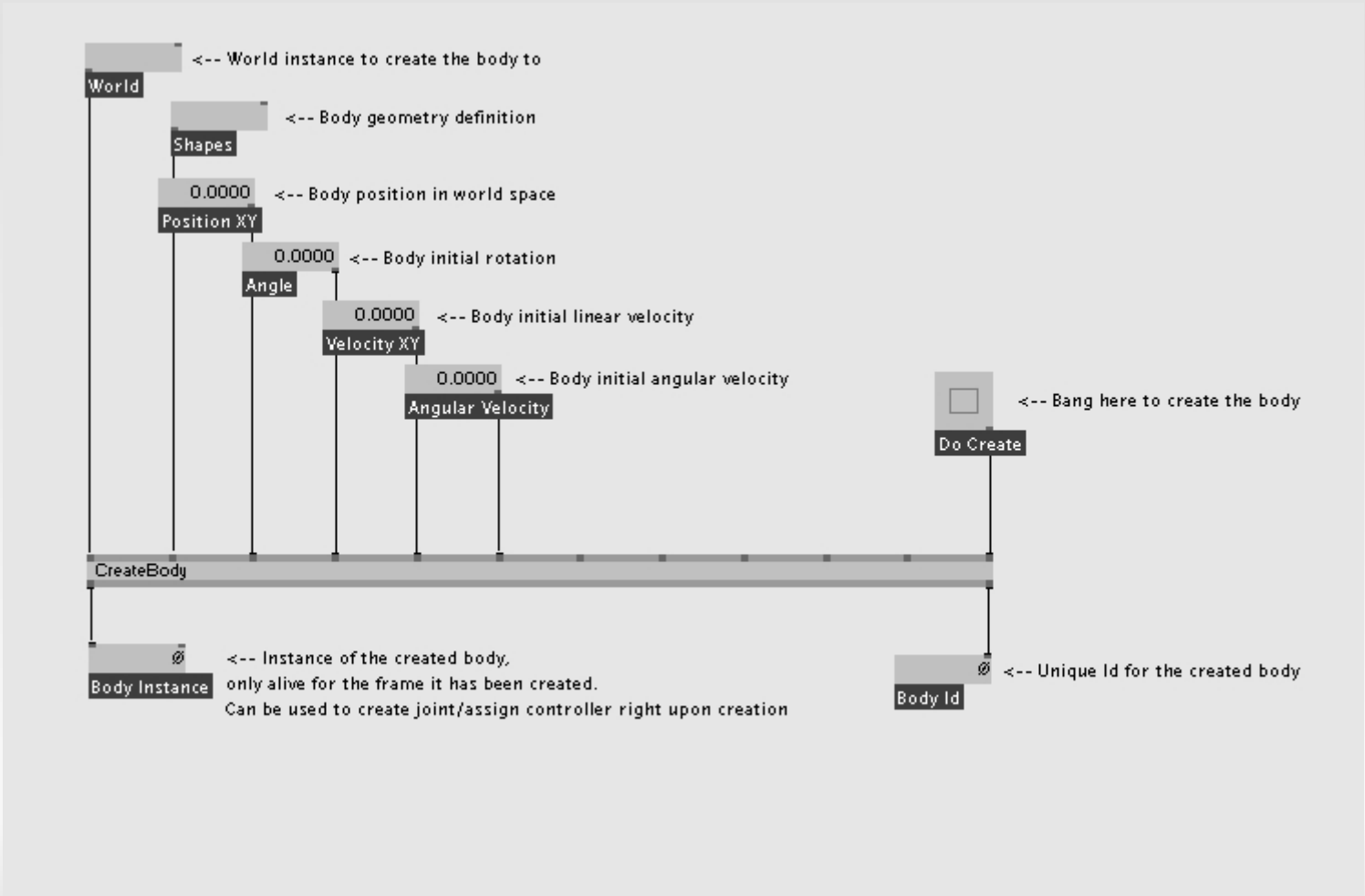
Physics based animations

Primitives



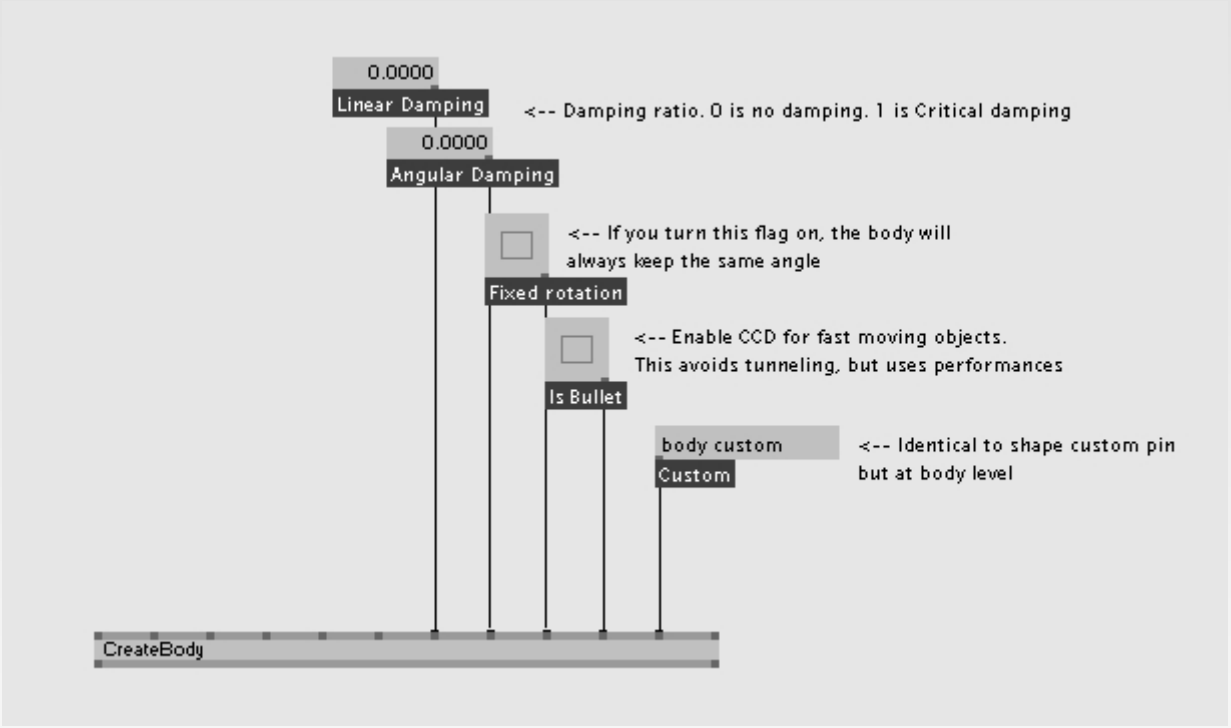
Physics based animations

Creating Bodies



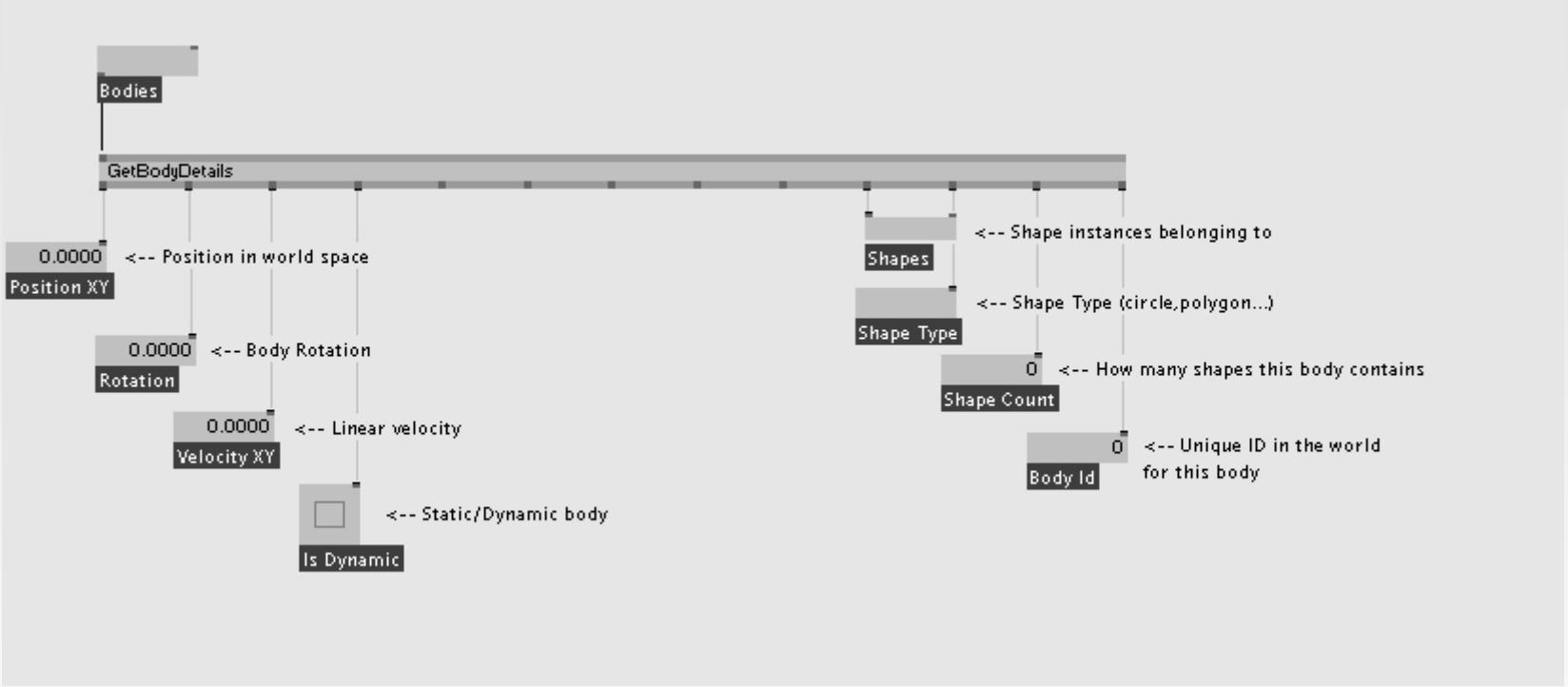
Physics based animations

Creating Bodies



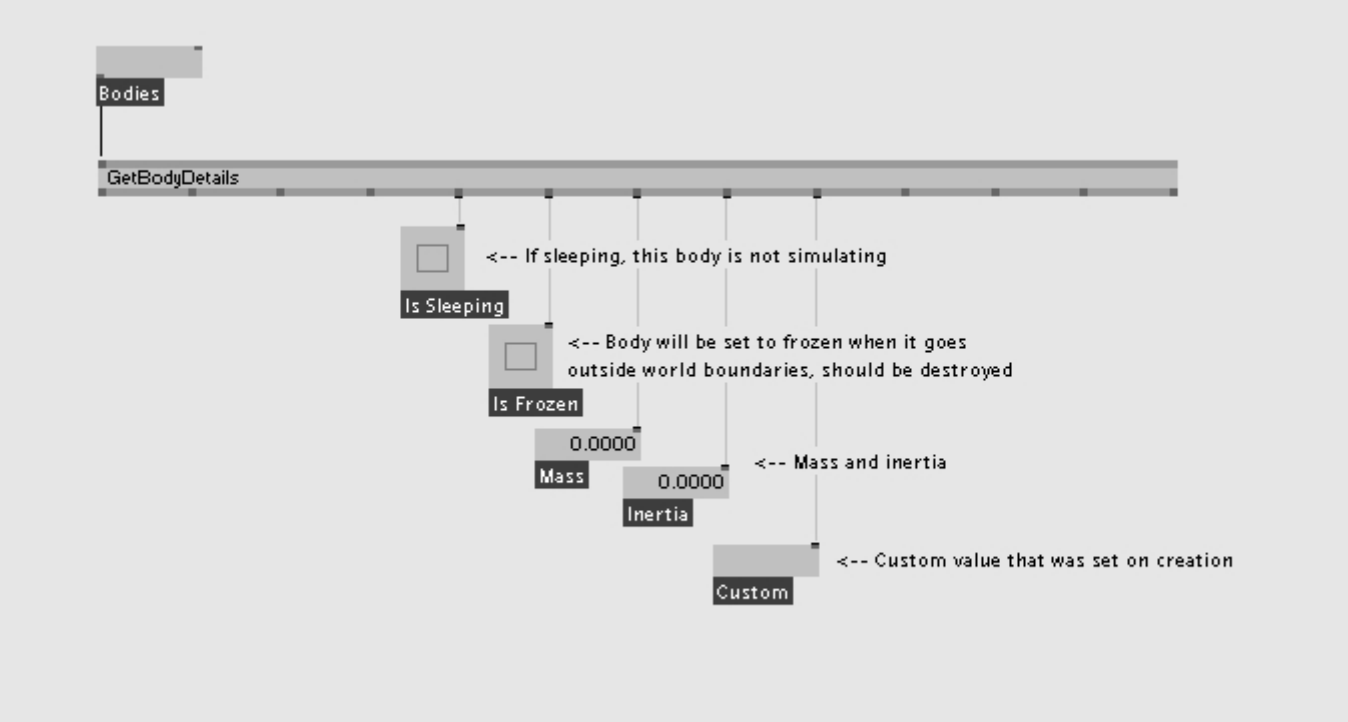
Physics based animations

Retrieve Body information



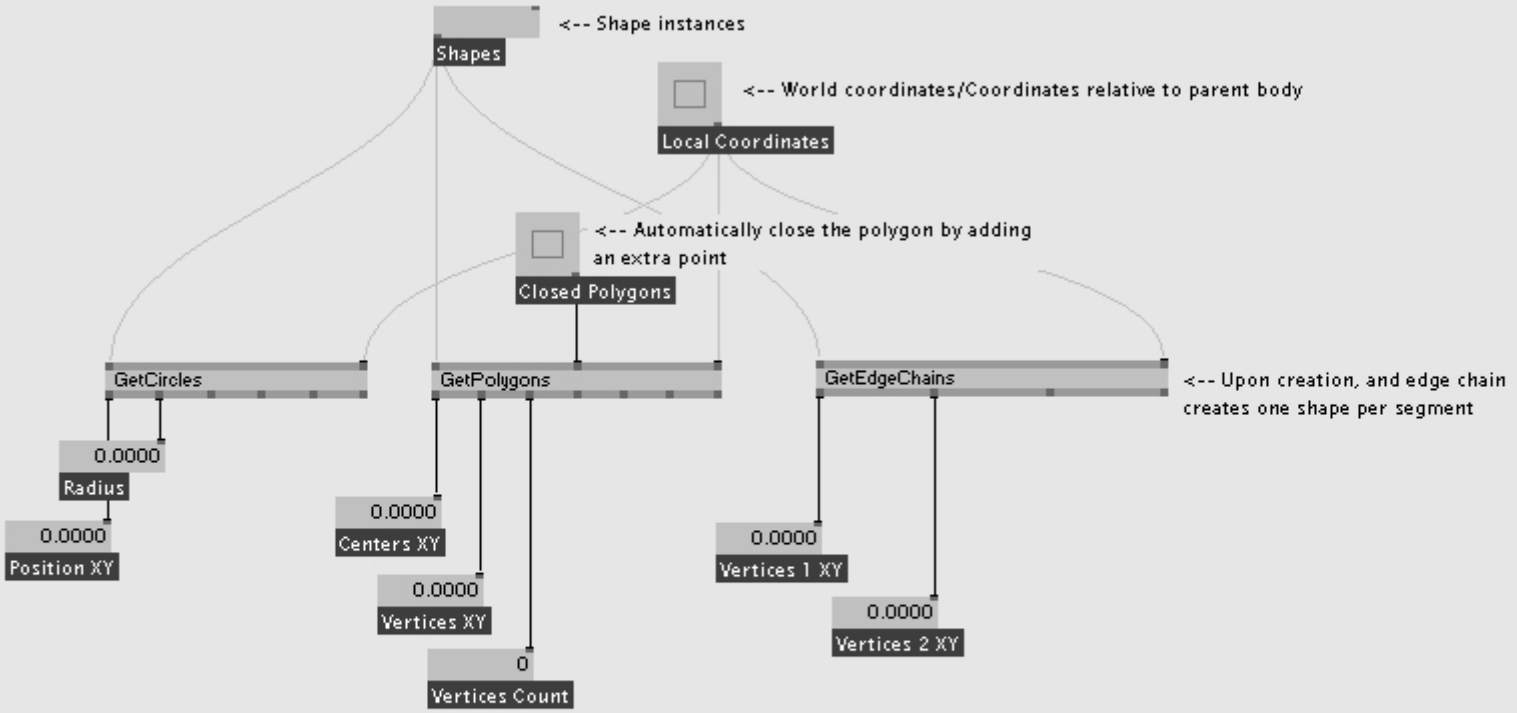
Physics based animations

Retrieve Body information



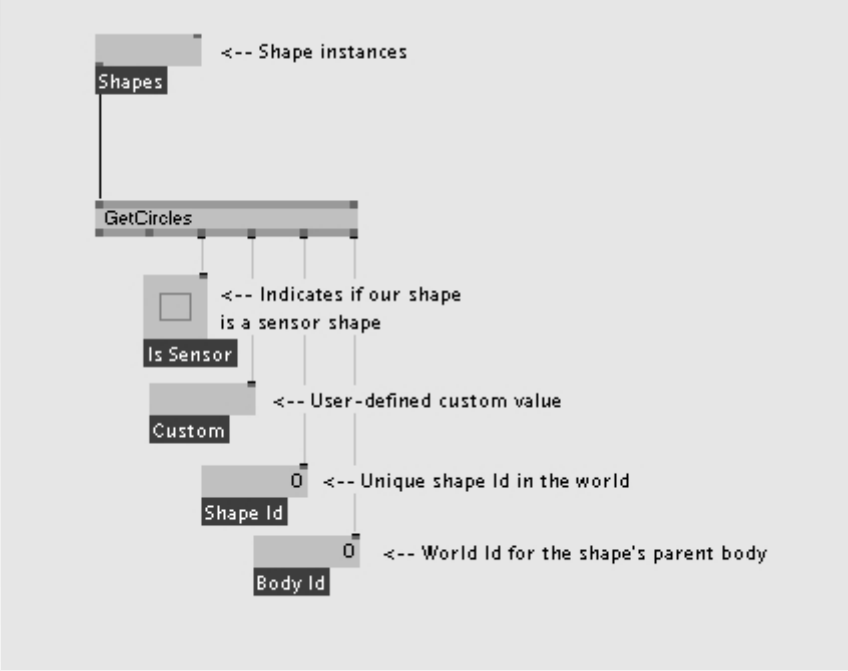
Physics based animations

Retrieve Shapes information



Physics based animations

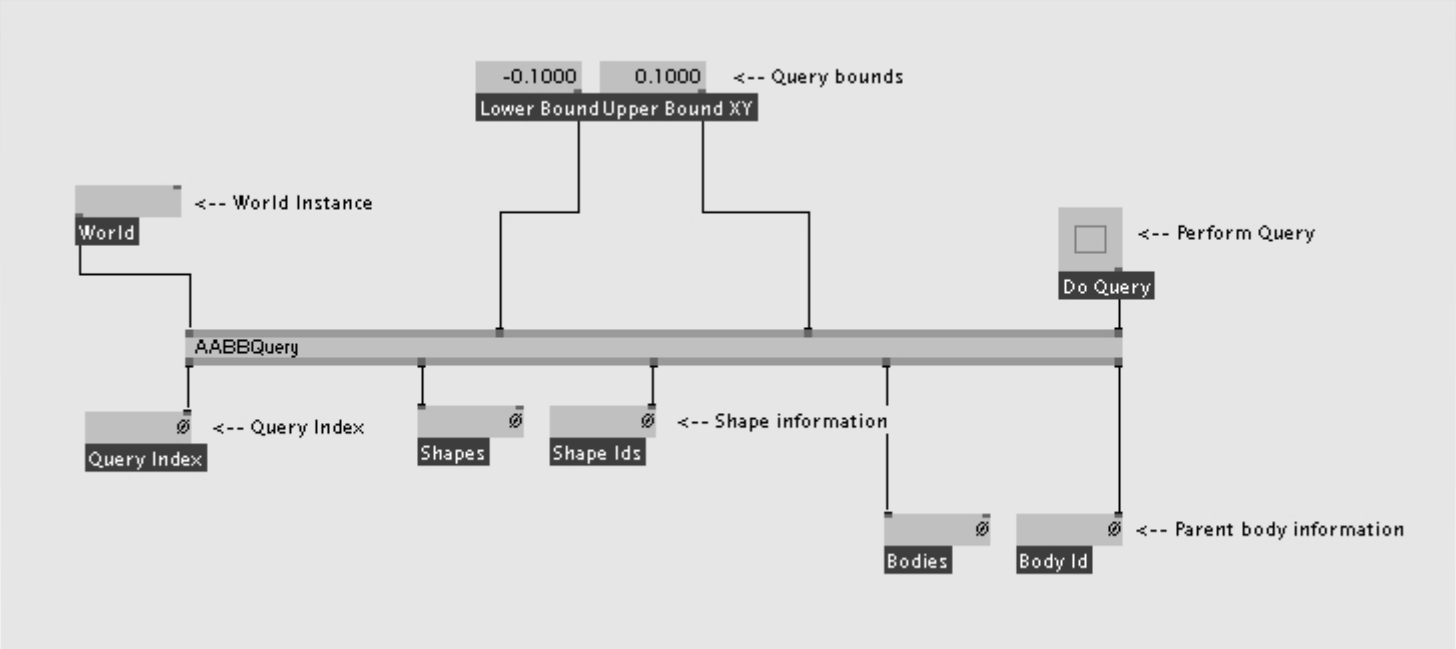
Retrieve Shapes information



-All the pins above are common to every single shape

Physics based animations

Find objects (AABBQuery)



-AABBQuery performs a query from the supplied ABB to shapes ABB. So positive result does not necessarily means that the shape overlaps.

-As AABBQuery performs AABB checks, it can be easily done in the Broadphase, hence it's a very fast way to find objects wich might intersect.

-Query is done against shapes, but as very often we want to interact with it's parent body, it is returned as a convenience

Physics based animations

Find objects (Ray Casts)

Physics based animations

Find objects (Test Points)

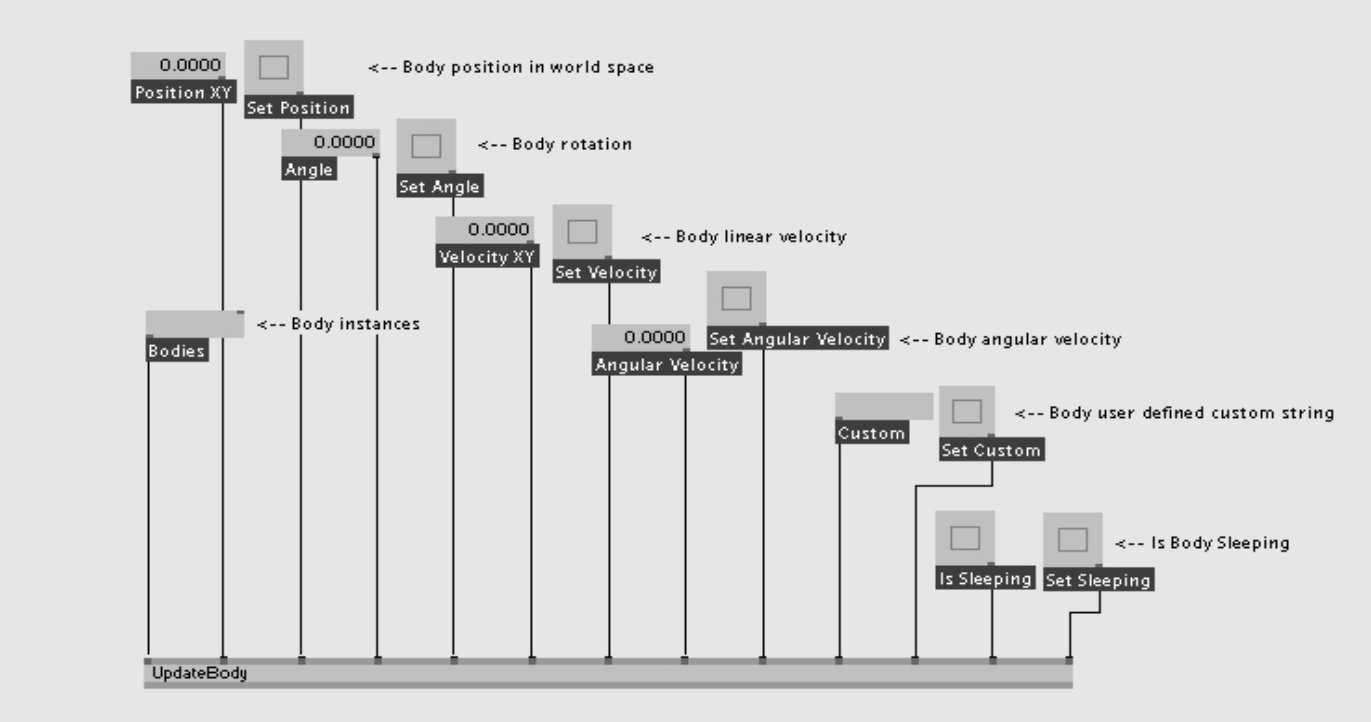
Physics based animations

Update Object properties

- UpdateBody
- DestroyBody
- UpdateShape
- ScaleShape

Physics based animations

Update Body properties



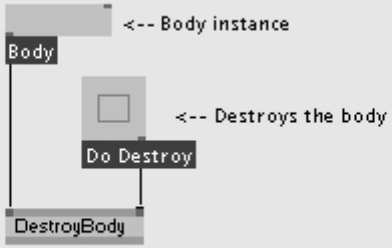
This node is fully spreadable, so you can set multiple bodiesat the same time upon conditions.

Putting a body to sleep will also Set all velocity components to 0.

Setting new position is generally not recommended, as it can break the simulation. More efficient methods for moving objects will be presented later.

Physics based animations

Destroy Body



This node is also fully spreadable, so you can set multiple bodies at the same time upon conditions.

Technical note:

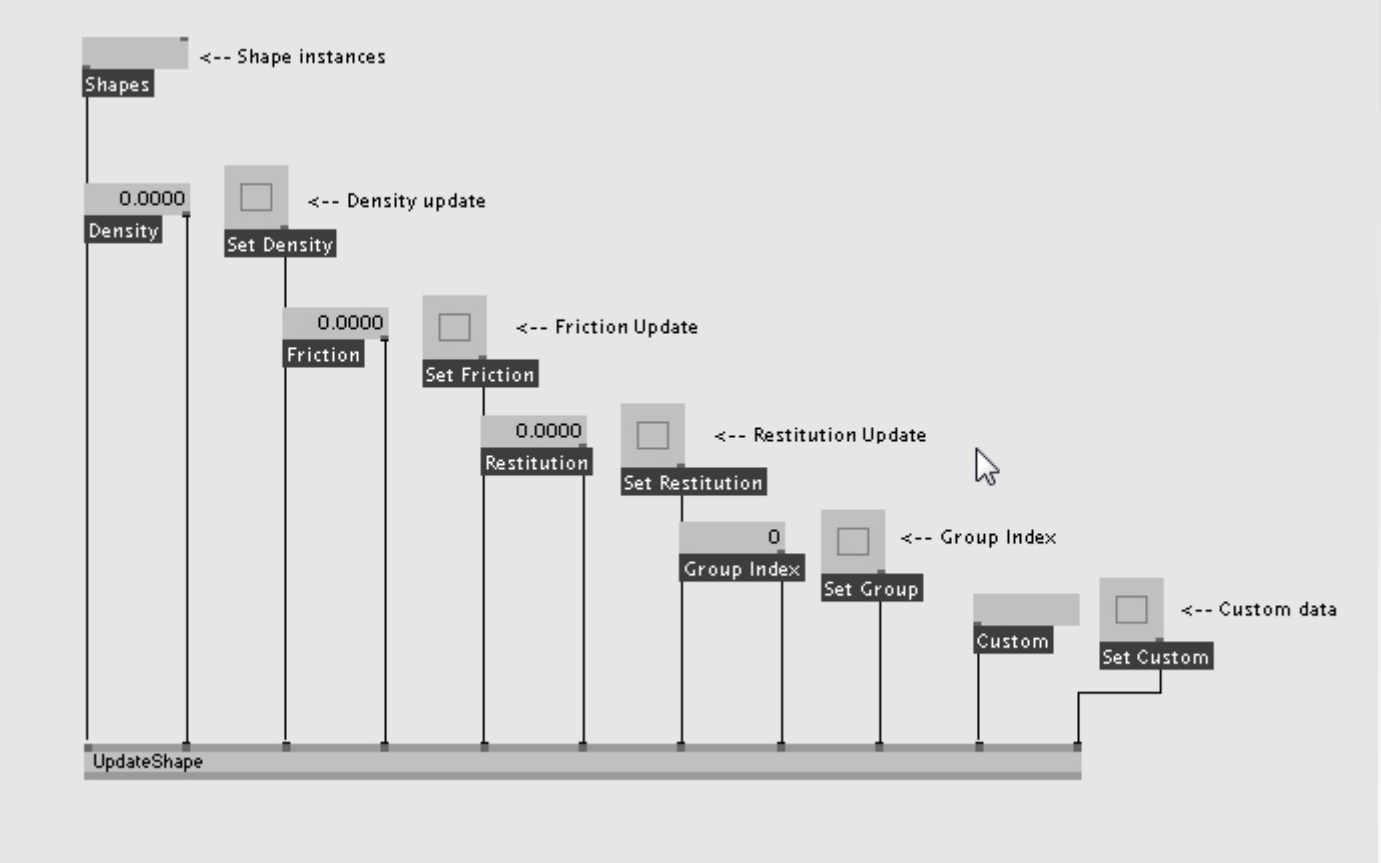
Internally and for stability purposes. A body is not immediately destroyed. Instead a flag tells it has to be destroyed instead.

The world node, before to execute time step, will check for all bodies marked for deletion and detroys them before the time step.

If we immediately destroy the body, we could have a case when we try to destory/update properties in the same frame, which would result in a crash.

Physics based animations

Update Shape Properties

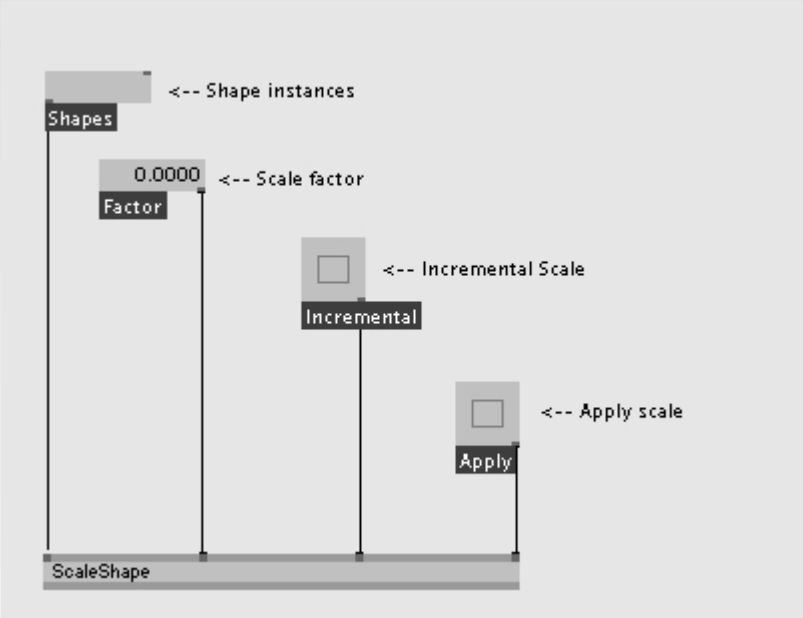


This node is also fully spreadable, so you can set multiple shapes at the same time upon conditions.

Setting a density to zero/non zero can switch the body from static/dynamic.

Physics based animations

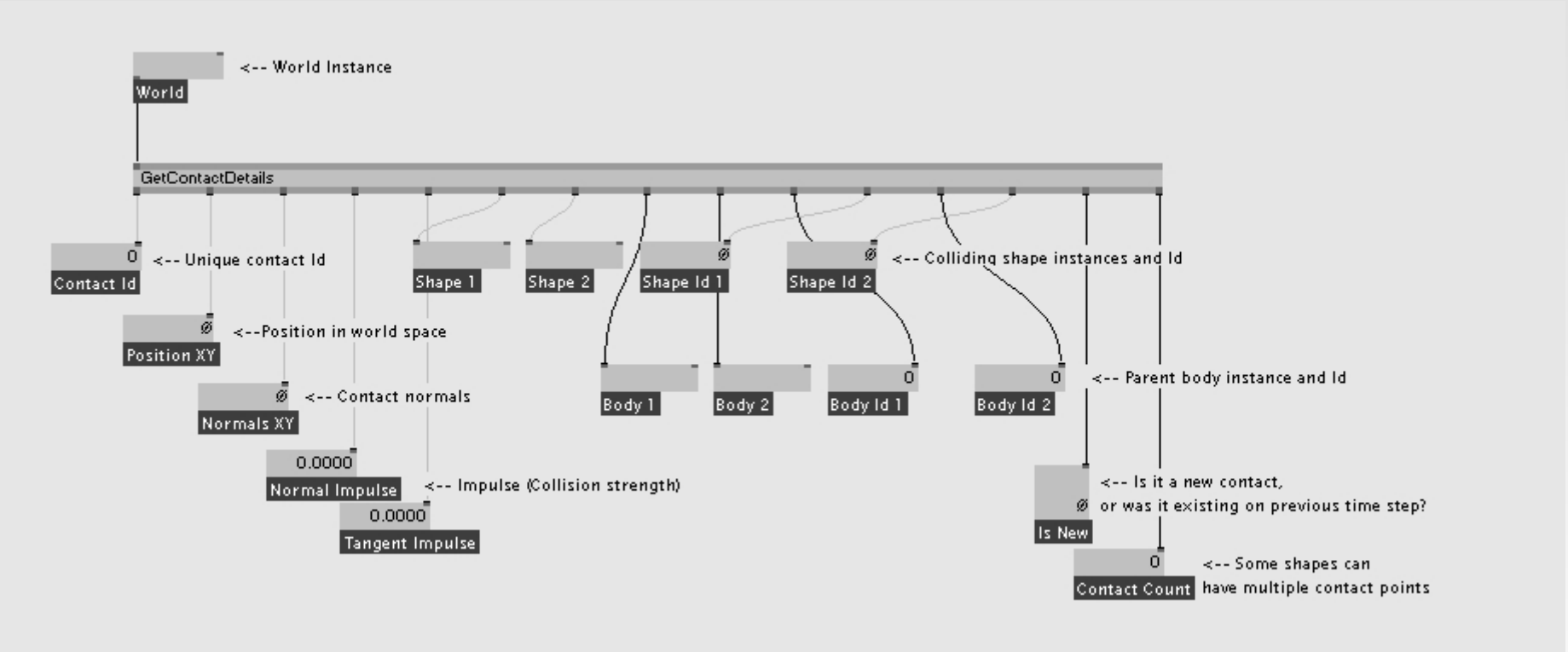
Scale Shape



- This node is also fully spreadable, so you can scale mutiple shapes.
- Incremental makes the new scale calculations as follows: if turned off, new scale = current size * scale factor. If turner on, new scale = current size + scale factor.
- Scale shape is a convenience. Eg: internally shape is destroyed/recreated while keeping same other properties.
- It is not possible to scale by dimension for boxes (unique interface for every shape type)

Physics based animations

Collisions – Retrieve Contact informations

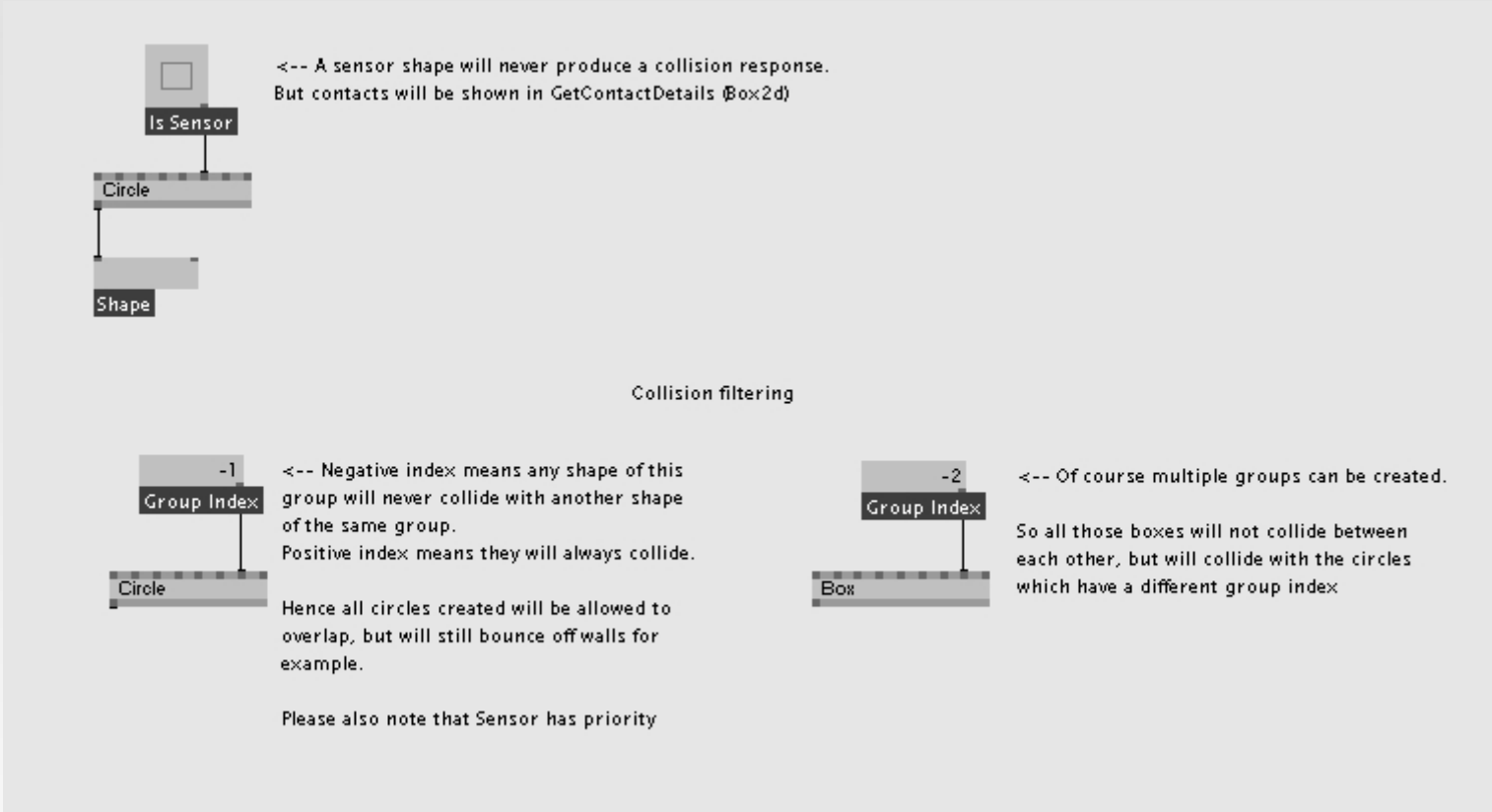


As for AABBQuery, only shape collide, but parent body is returned as convenience.

A contact count is provided as two shapes can have multiple contact points (two boxes with the same angle for example).

Physics based animations

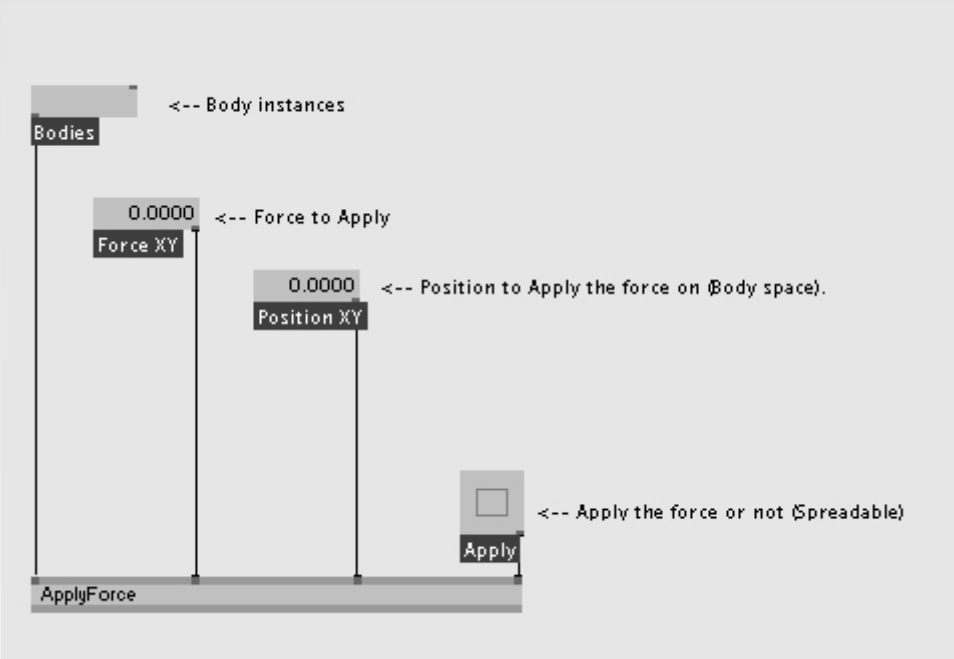
Collisions – Sensors and Filters



Common sensor uses: Proximity detection, Check if an object is in an area, Bullets in tank game (we dont wnat the object to bounce, just know when if reaches another players)

Physics based animations

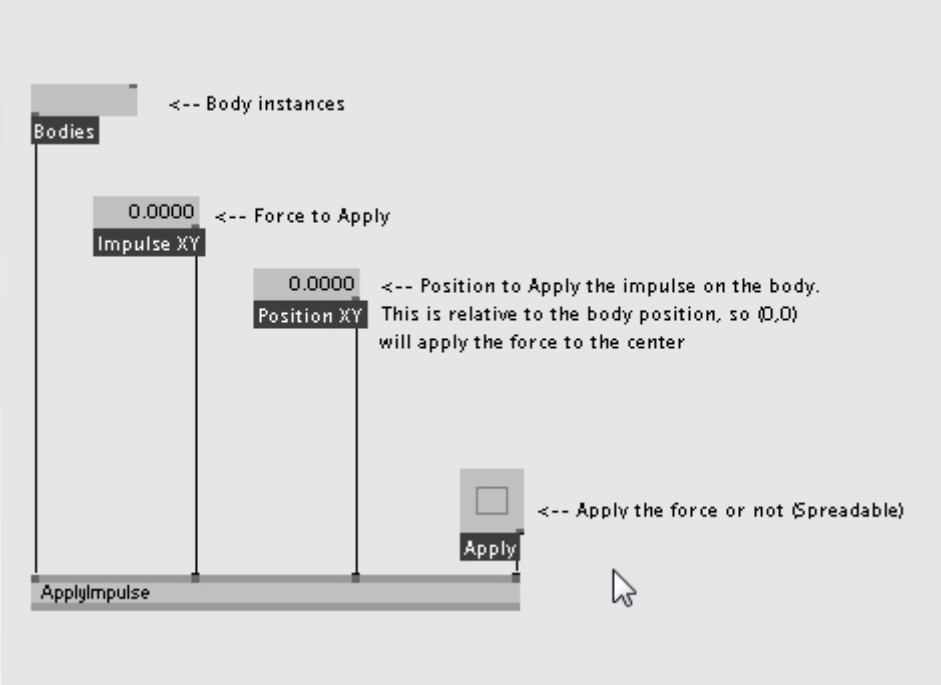
Body Interactions : Force



- Position XY is in Body space, so a position of (0,0) will apply the force to the center of the object.
- Applying a force non central to the object will also impact torque (force on object rotation)
- ApplyForce impacts on the body acceleration.

Physics based animations

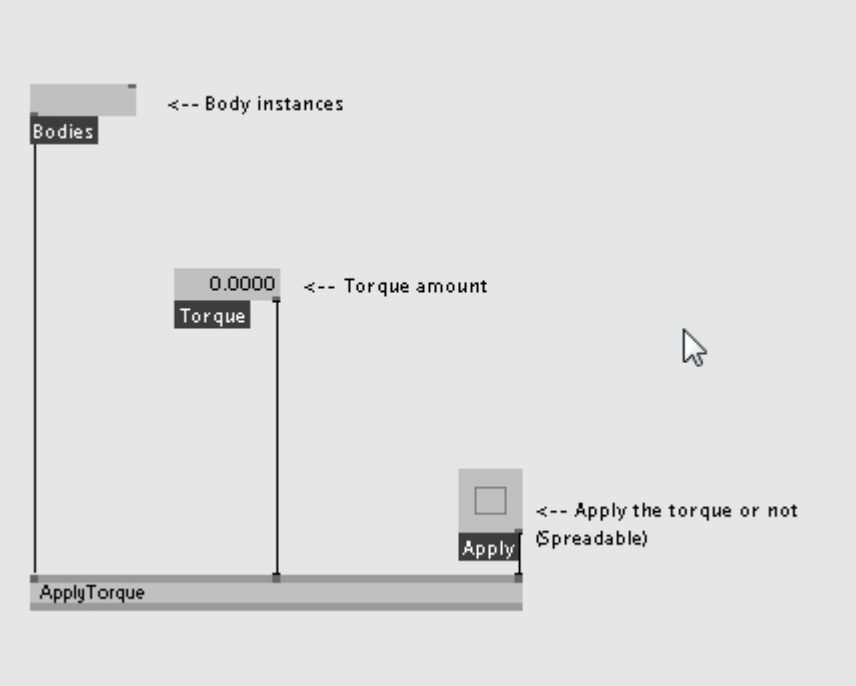
Body Interactions : Impulse



- Position XY is in Body space, so a position of (0,0) will apply the impulse to the center of the object.
- Applying an impulse non central to the object will also impact torque (force on object rotation)
- ApplyImpulse directly impacts on the body velocity, so very small values can have so drastic changes

Physics based animations

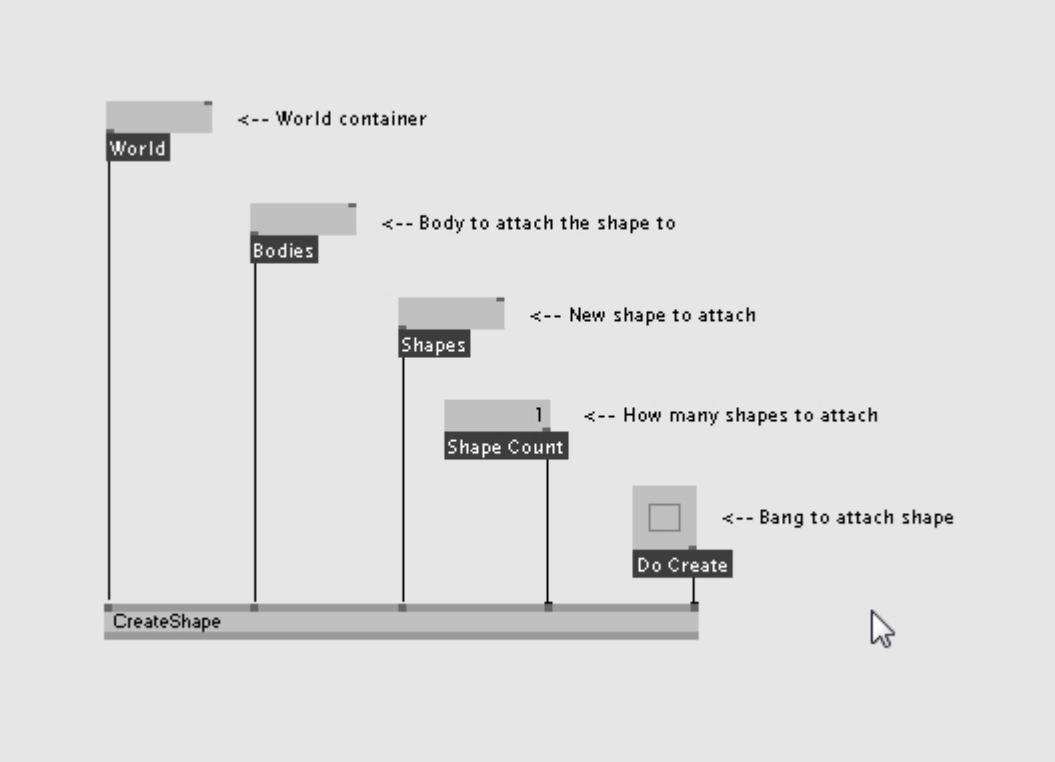
Body Interactions : Torque



- Torque impacts object rotation (not any linear vector parameters)
- Torque can be positive or negative of course.

Physics based animations

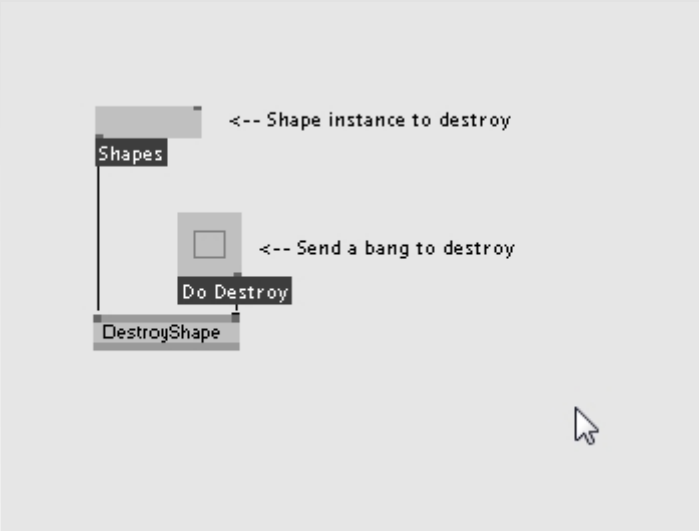
Compound Bodies : Create Shape



- Bodies can contain several shapes
- Collision between shapes belonging to the same body will always be disabled

Physics based animations

Compound Bodies : Destroy Shape



- Please note that when we destroy shapes, some bodies can end up with no shape at all. This need to be taken into account, but is up to the user.

Physics based animations

Joints

Joints allow you to create constraints between bodies.

Generally they are used to limit the degree of freedom between objects.

For example, think of two objects attached by a rope.

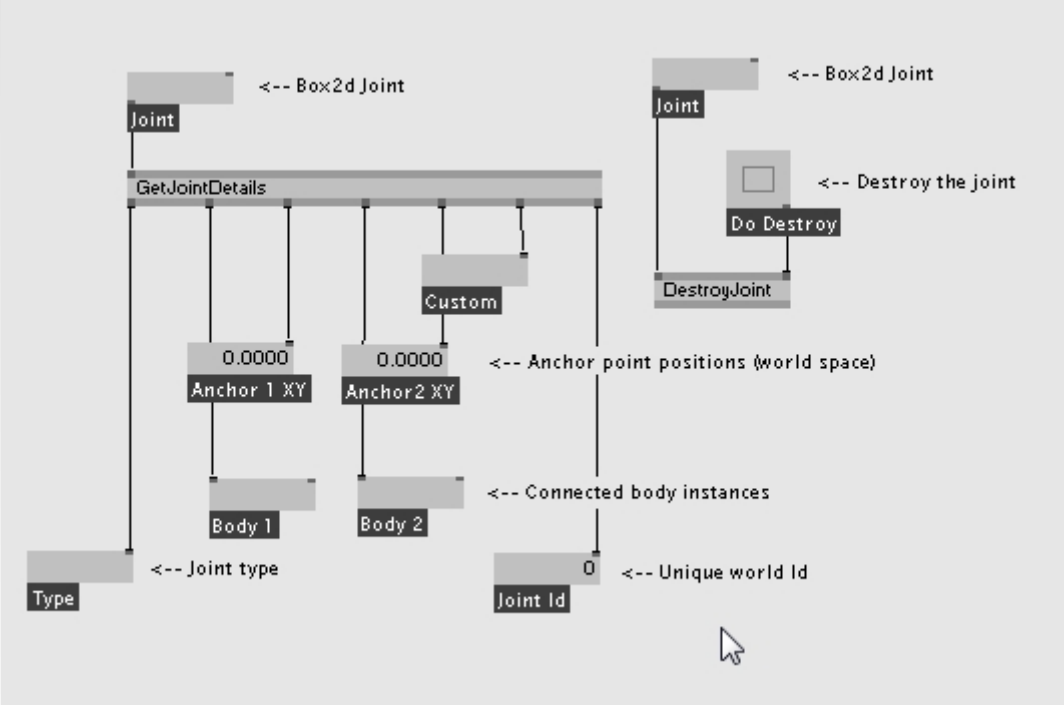
Joints can provides different features:

- Limits: This allows you to control the degree of freedom
- Motors: You can 2drive the joint at a specified speed
- Connection point from a joint to an object is called an Anchor
- You can either connect a body to another one. For some joints you might only want to connect it to an arbitrary point in space. In this case we can use the ground output from the world node.
- Most joints also provides you a flag indicating if connected bodies should collide with each other.

Physics based animations

Joints

Before to see every joints in details, here are the two nodes allowing to interact with existing ones:



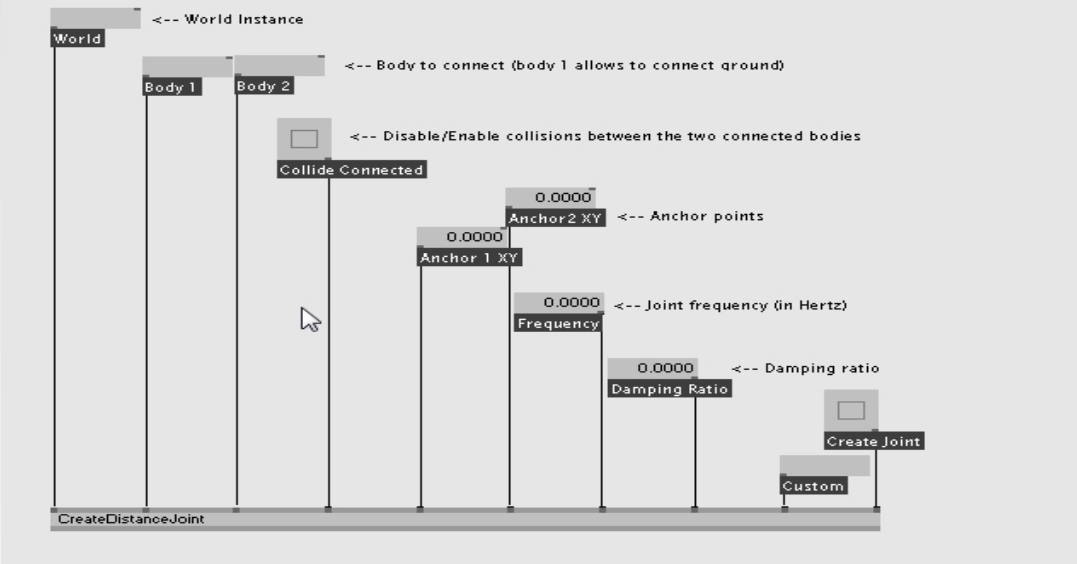
-Destroy joint works exactly the same way as destory body.

-If a body is destroyed, all joints attached to it are also destroyed.

Physics based animations

Joints : Distance

This creates a constraint so objects haver to stay at a fixed distance to each other.



Frequency and Damping ratio are used to ”loosen” the constraint, and make it behave more like a spring.

Frequency is measured in Hertz (oscillator)

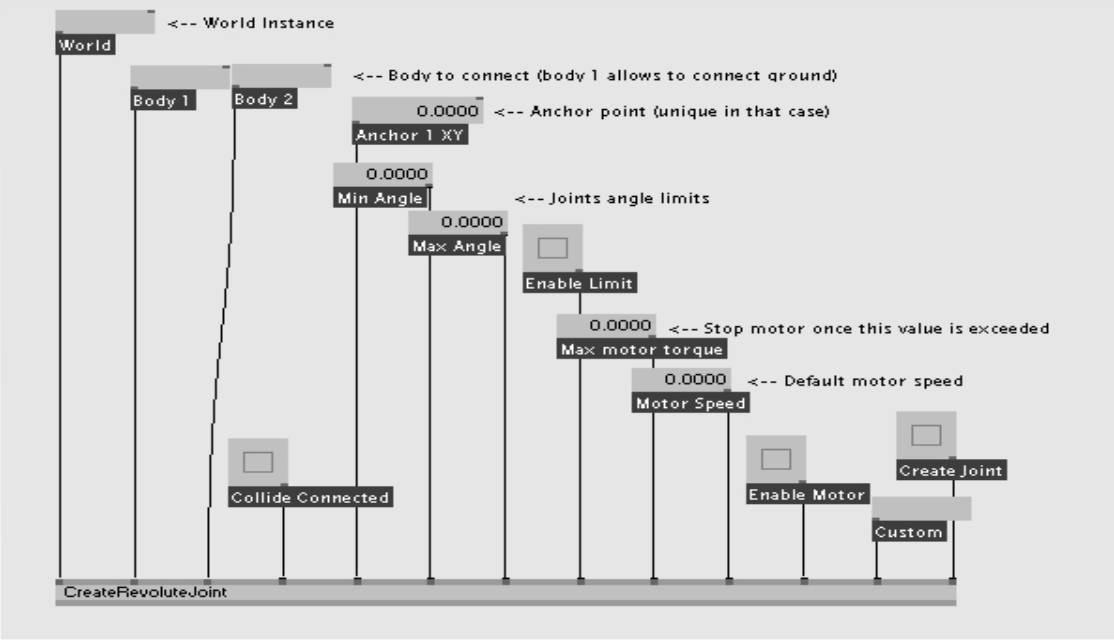
Damping ratio says how ”thick” is the spring (0 means fully rigid, 1 fully loose)

Distance joint allows you to attach an object to the ground

Physics based animations

Joints : Revolute

This creates a common anchor point, limiting objects rotation. This is like a ”pin”



-As we pin both objects, only one anchor is necessary.

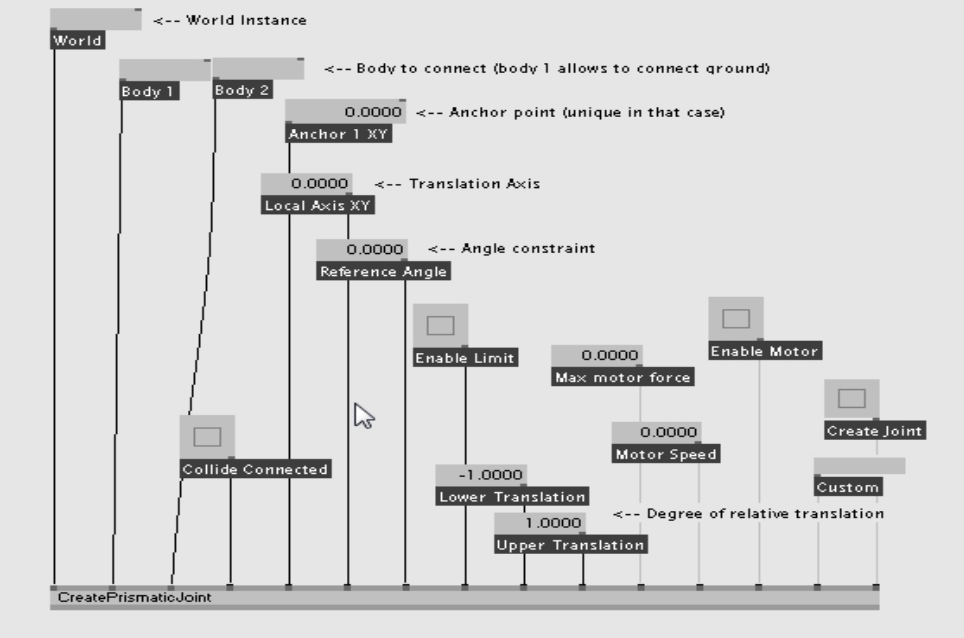
-We can specifiy rotation limits, so this is useful to create objects like ragdolls.

-Motor can be used for this type of joints

Physics based animations

Joints : Prismatic

This works like revolute, but creates a distance limit, and also prevents any relative rotation



-As we pin both objects, only one anchor is necessary.

-We can specifiy translation limit (relative), so make sure lower is negative and upper is positive.

-Motor can be used for this type of joints

Physics based animations

Objects Controllers

-A controller is an instance to seamlessly process a number of bodies in a certain way.

We can assign/unassign bodies to controllers at will, bodies can be attached to multiple controllers at once.

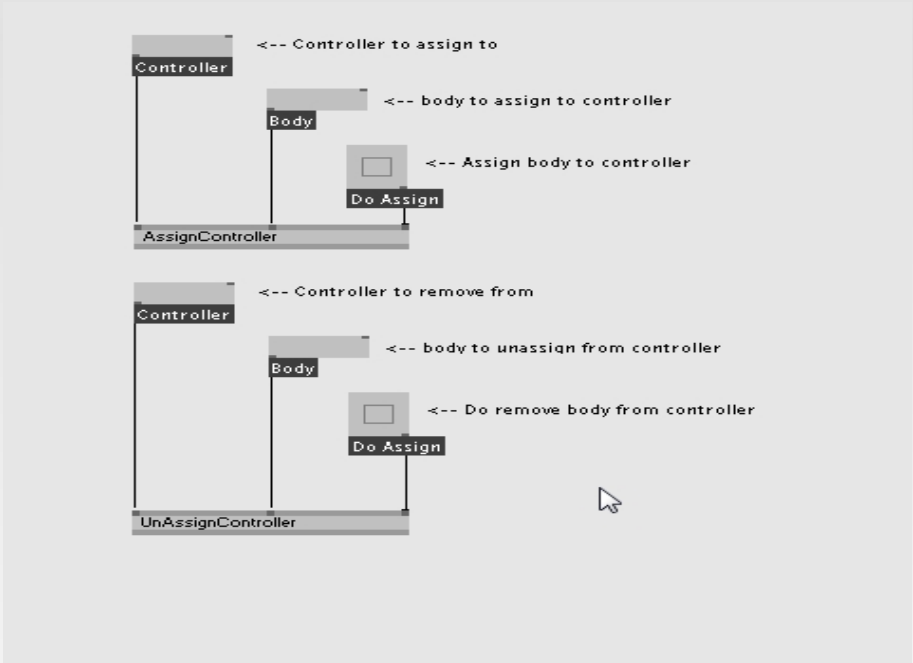
It's an easy way to manage a lot of objects at once.

-Every controller node also contains a ”Clear” pin, to remove all bodies that this controller contains at once, provided as a convenience.

-Every controller parameter can be of course changed in real time

Physics based animations

Objects Controllers

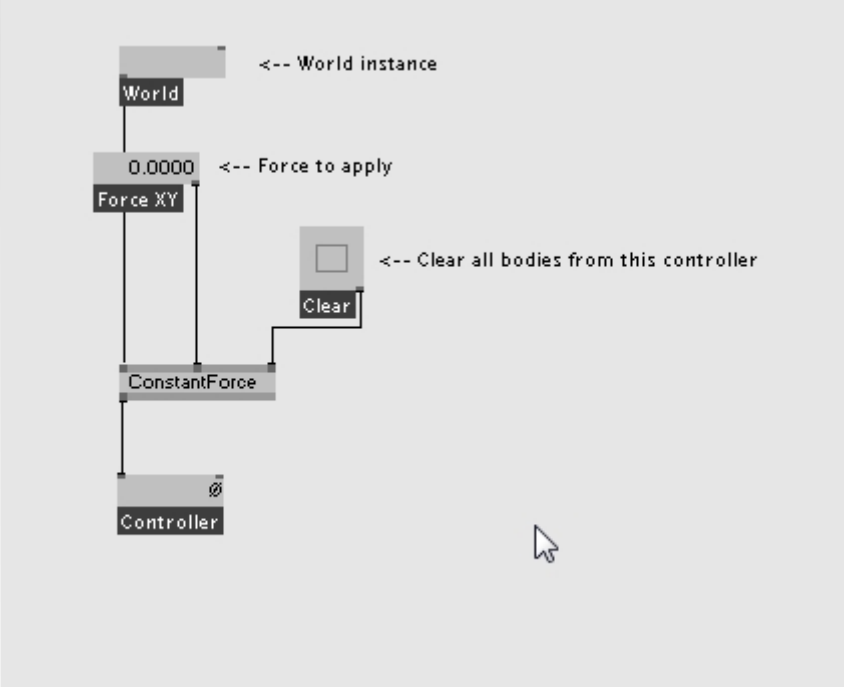


-Assigning/Unassigning bodies to controllers it pretty straightforward, just using the two node (please note they are fully spreadable).

-The engine will take care that no multiple assignments have been done, and will take care of body destruction.

Physics based animations

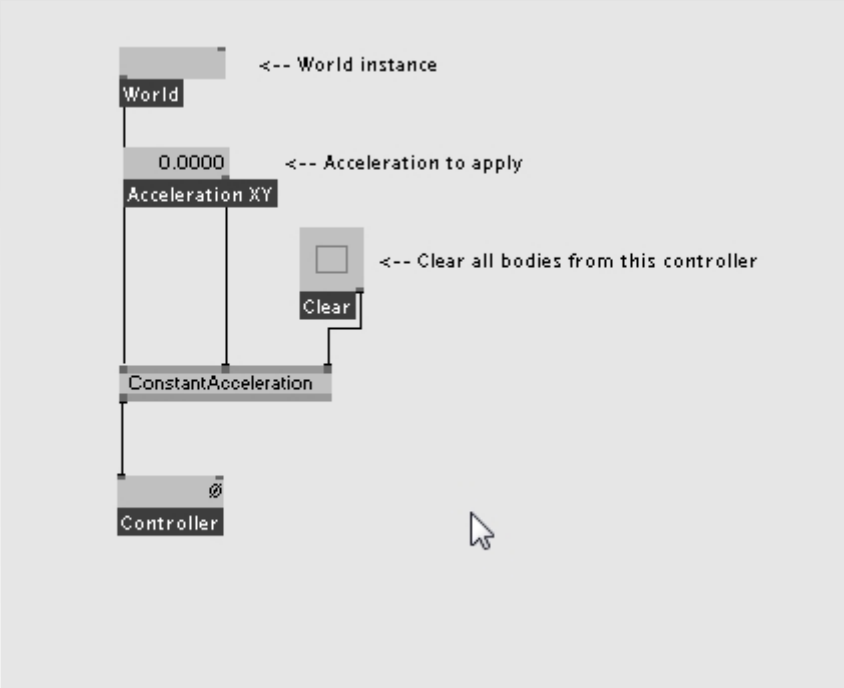
Objects Controllers : Constant force



Constant force applies a force to each body it contains. It's equivalent to using ApplyForce every frame to a set of bodies.

Physics based animations

Objects Controllers : Constant acceleration

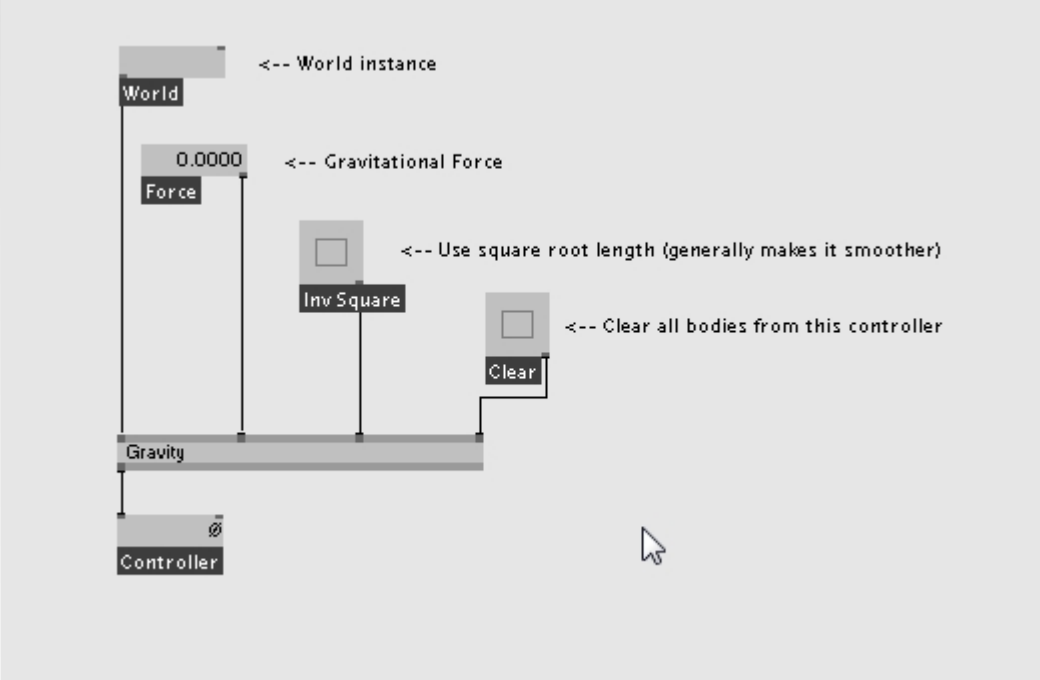


Constant acceleration is modyfying object acceleration. This is how gravity works eg:

Body.Velocity = Body.Velocity + DeltaTime * A

Physics based animations

Objects Controllers : Gravity



Unlike ConstantAcceleration, this controller does not apply a uniform graivty.

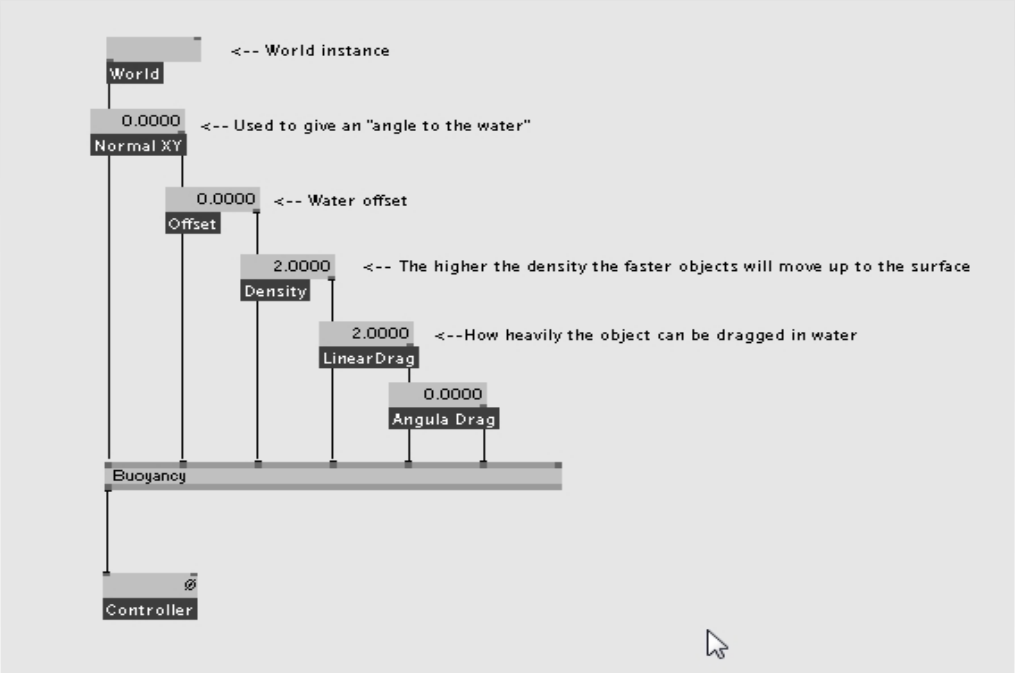
Each body contained in this controller will exert a gravitational force against each other.

Gravitational force depends on objects distance between ech other and their mass.

Force can be positive (attractor) or negative (repulsor)

Physics based animations

Objects Controllers : Buoyancy

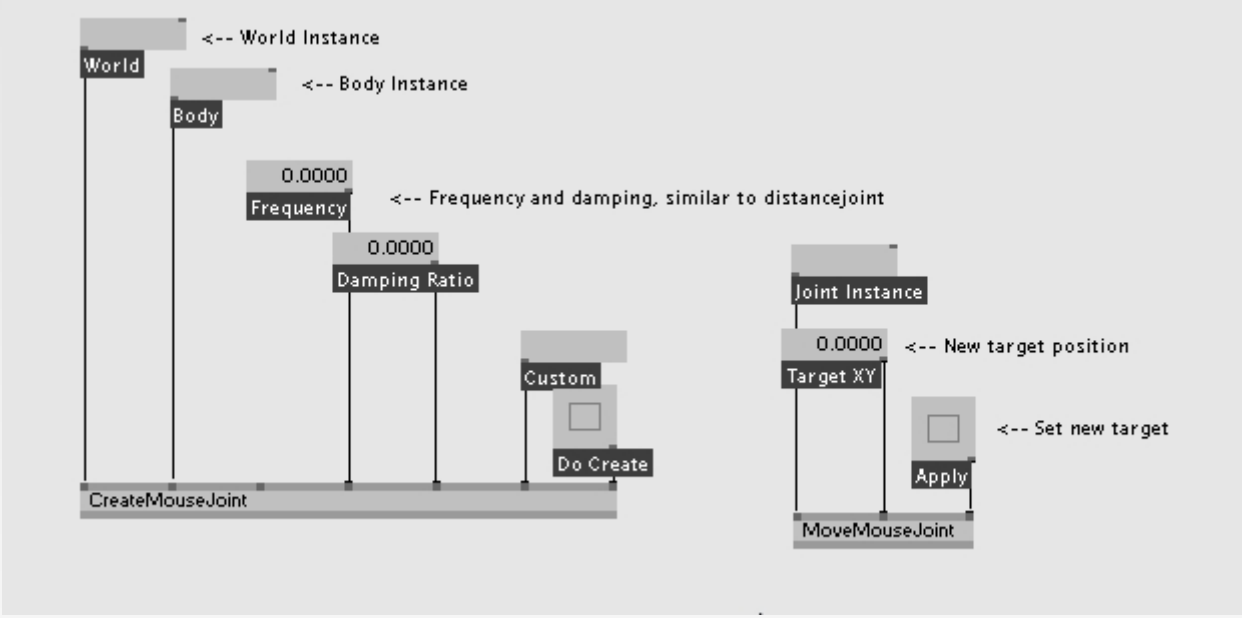


This simulates objects moving in water. See usage example

Physics based animations

More on Joints : Mouse Joints

Mouse joint is a simple facilty which was used to facilitate debugging, but turned out into a nice feature. We can assign a position and move an object to that position. Internally this is like a ”moveable” distance joint

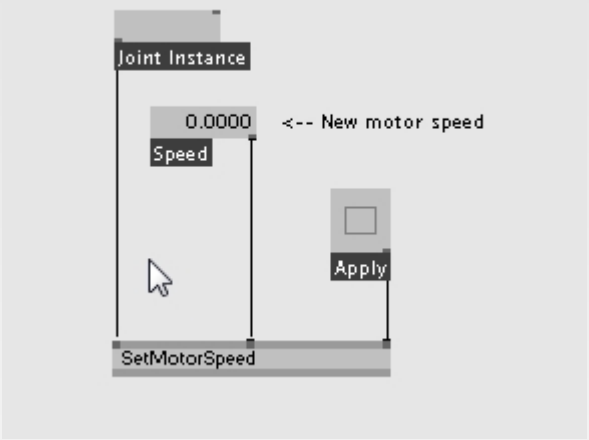


As a specific case, this joint only has one body.

Physics based animations

More on Joints : Motors

As some joints allow motor, we can also update the motor speed.



Engine takes care if a joint doesnt support motor, but it is recommended to do some filtering beforehand.

Physics based animations

Pratical box2d cases

Physics based animations

Retrieve objects (Advanced)

Retrieving objects can be done applying different techniques:

-We can use the body unique Id: in this case, we simply need to use GetSlice (Node)

-One more advanced technique comes by using the Custom pin. We can assign an abritrary string to an object, then use Sift (String) + GetSlice (Node) to filter object instances.

-If we need to store more than a single information inside the custom pin, we can use either semicolon separated strings, or xml data (make sure all objects create an xml with the same structure to simplify)

Physics based animations

Texturing/Coloring

Coloring/Texturing objects can be done multiple ways. Both share the same concepts so any technique applies to both.

Case 1: Non deterministic

In that case we only need the object not to change color, it will remain the same all it lifetime, and we do not mind which color.

Simple way is to create a simple static color pool (Random color spread for example), then use the body/shape id with getslice color.

As the object id is constant trough all the object lifetime, the object will never change color (unless we change the color pool)

Physics based animations

Texturing/Coloring

Case 2: Deterministic

In that case we only need the object not to change color, it will remain the same all it lifetime, but we want to specify the color upon creation.

-Technique 1: Store color as Hex in custom pin. This works and is very easy, but this will not apply to texturing. It also of course allows to change color using UpdateBody

-Technique 2: Create a color pool as previously, and store the color slice index in the custom pin. This will also apply easily to texturing.

-Technique 3: Create a color pool using a pair structure (code/color). Code is just a string to represent the color. We can then use Sift (String) and GetSlice (Color).

Physics based animations

Device/User interactions

Box2d offers a lot of features to interact with an existing world. As there is a lot of different devices, doing a generic system is almost impossible.

Altough some devices are more adapted for some interactions than others.

We will speak about device types and discuss different ways of integrations.

Physics based animations

Device/User interactions : MIDI

They generally offer a standard interface (notes/controls), so building a midi system is generally quite straightforward.

Some implementations cases:

- Create objects using midi notes (Keyboards/Pads).
- Update gravity/controller data (Any 2d vector works pretty well with any device which has an x/y slider), we can also easily apply factors using sliders/knobs. Enabling/disabling controllers can also easily be achieved using toggles.

Physics based animations

Device/User interactions : Accelerometers

Some examples include:

- Wii remotes
- Iphones

Some general good uses for them:

- controllers (update forces)
- Applyforce (so we can interact with single objects)

Physics based animations

Device/User interactions : Multi Touch

Multi touch integration with box2d is a vast subject. Some different possible interactions are:

- Object Creation/Destruction
- Move objects (Mouse joints/Applyforce works pretty well)

The main difficulty with multitouch is we often need to track if an object is assigned to a touch.

This can be easily achieved by storing touch id in the custom object pin.

Simple ways of processing hit tests can be done using AABBQuery, HitTest (polygon/quad), then processing the data.

We also need some modules to tell us when a touch appears/disappear.

Physics based animations

Build complex models

Once we get past the simple cool examples, we then often want to create more complex models.

This requires more efforts and forward design thinking to keep patching as clean as possible.

Some general considerations:

- Know your tools: The key to build a model is to know what feature to use in what case. Some joints type work better than other in certain cases for examples. So it is important to test as many node/parameter as possible to be able to know which one to use
- Build a framework: Generally a model will have 3 main parts, which are initial model creation, interactions, display. Having those in separate subpatches helps a lot once the model starts to be quite big.
- Try to keep interactions as device independent as possible: Having some hardcoded midi in the interaction patch is not a good idea. So having subpatches with pins for every interactions will make things easier.
- Build a lot of generic modules: modules like getbodybycustom are fairly general purpose and helps a lot.
- For evolving worlds/games, nodes like automata will simplify your life greatly.

Physics based animations

Mapping

Box2d can easily adapt to projection mapping. Techniques to build a model depends greatly on the complexity of the shape.

For simple models, boxes/circles/polygons will offer nice features. Building a simple ”drawing tool” will make you life slightly easier and save your time (common storage can be xml/sqllite databases)

For more complex models edge chains will come to mind. Image processing will often be needed in that case. Please also note that edges are expensive so you will want to simplify your polygons (to keep a good performance/accuracy balance).

Physics based animations

Boygrouping

Building box2d models in a boygroup environment is quite straightforward.

Generally a good pratice is to separate the server/client engine that way:

- Server: World, object creation, interactions
- Client: Keep getbodydetails in the server, then send coordinates to clients, and process display

This offers nice performances as the server will take care of all the simulation.

We can do custom filtering in the clients to avoid drawing objects which doesn't belong to the view.

Physics based animations

Box2d Future

Development of box2d is still ongoing, and some new features are already in development.

- No more world/shape limit
- Improved AABB/Raycasts
- Enable/Disable bodies on the fly
- New joints (Friction/Weld/Rope)
- Edge chain revamp (bi directional by default)
- Compound shape system revamp (ability to group shapes before to add them to a body)
- Fully integrated with new dynamic plugins, so you can access object properties, apply interactions in your own code. (You can write your own controller for example, create some complex collision filtering logic...). That also means that it will be beta24 required.
- Simplified retrieval system. More custom nodes to simplify patching, major changes in shapes retrieval.
- Features request welcomed :)