Farseer Physics Engine

Revision History

FarseerPhysics 1.0.0.6

(Red items indicate possible breaking change. Sorry for the trouble.)

- **Changed PhysicsSimulator.Clear() to call "ConstructPhysicsSimulator(gravity)"**
  - Seemed like a better way to "reset" the engine.
- **Added QuadraticDrag option to Body**
  - Quadratic drag is useful for bodies that travel at high speeds. With quadratic drag, the drag applied to a body is proportional to its speed squared rather than just its speed like the normal linear drag.
  - Added **Body.IsQuadraticDragEnabled** property to enabled/disable quadratic drag. It is disable by default. There is a small performance hit when it's enabled.
  - Added **Body.QuadraticDragCoeficient**. This will only be used if quadratic drag is enabled.
  - Made changes to Body.ApplyDrag to apply quadratic drag if enabled.
- **Added PhysicsSimulator.Enabled property.**
  - Setting this to false will block the physics engine from updating. Essentially pauses the physics engine.
- **Added a check to Grid.Intersect to avoid a divide by zero situation.**

```
if (normal.X != 0 || normal.Y != 0)
  {
      Vector2.Normalize(ref normal, out normal);
      feature = new Feature(vector, normal, distance);
      return true;
  }
```

FarseerPhysics 1.0.0.5

(Red items indicate possible breaking change. Sorry for the trouble.)

- **Renamed delegate Body.OnChangedHandler to Body.UpdatedEventHandler**
  - May require name fix in your code.
- **Renamed Body.OnUpdated to Body.Updated**
  - May require name fix in your code.
  - This was done to conform to naming guidelines for .Net frameworks.
- **Renamed Event Body.OnDisposed to Body.Disposed**
  - See above for fix-up guide and reason.
- **Renamed Geom. CollisionHandlerDelegate to Geom.CollisionEventHandler**
  - May require name fix in your code.
  - Conforming to .Net framework naming guidlines
- **Renamed Geom.CollisionHandler to Geom.Collision**
  - May require name fix in your code.
  - Conforming to .Net framework naming guidelines
- **Renamed GeomList.GeomAddedRemovedDelegate to GeomList.ContentsChangedEventHandler**

    - May require name fix in your code.
    - Conforming to .Net framework naming guidelines
- **Renamed GeomList.GeomAddedHandler to GeomList.Added**
    - May require name fix in your code.
    - Conforming to .Net framework naming guidelines
- **Renamed GeomList.GeomRemovedHandler to GeomList.Removed**
    - May require name fix in your code.
    - Conforming to .Net framework naming guidelines
- **Made name changes similar to GeomList to BodyList, ControllerList, and JointList**

- **Modified FluidDragController**
    - Previously, the FluidDragController used an AABB to describe where the 'fluid' drag should take affect. In order to make the fluid drag controller more flexible and able to use 'shapes' other than an AABB, I changed it to use an 'IFluidContainer' interface instead. (see below)
    - Probably not many using this as it's new and undocumented
- **Added LinkedList, LinkedList.Node and Stack collection classes to new namespace Collections.Generic.**
    - These were added for the Silverlight version of the engine since they do not exist in the new Silvelight 2.0 beta.
    - The XNA version will still use the native .Net version of these collections.
    - Thanks to Bill Reiss for adding these classes and fixing up the SelectiveSweep and SweepAndPrune broad phase colliders to use them.
- **Added IFluidContainer interface**
    - This interface defines 2 methods
        - Intersect(AABB aabb) determines a sort of broad phase intersection
        - Contains(ref Vector2 vector) determines if a point is contained by the FluidContainer
    - Any class that implements these 2 methods can be used by the FluidDragController.
- **Added AABBFluidContainer**
    - Implements the IFluidContainer interface for AABBs.
- **Added WaveController**
    - Implements IFluidContainer
    - Controls the generation of 2D rolling waves
- **Added new 'Controllers' namespace**
    - I expect to develop more and more controllers and felt a namespace was in order.
    - Springs, the Base Controller class, and Controller Factory will remain in the Dynamics namespace. The Controllers namepace is more for higher level things like waves and water physics.
- **Added "Enabled" bit to controllers so they can be turned on and off**
- **Added the following methods to the Calculator class**
    - Note:These were provided by 'sickbattery' from the codeplex forums. I have not had a chance to test them so let me know if you find any issues with them
    - public static float VectorToRadians(Vector2 vector)
    - public static Vector2 RadiansToVector(float radians)
    - public static void RadiansToVector(float radians, ref Vector2 vector)
    - public static void RotateVector(ref Vector2 vector, float radians)
    - public static Vector2 LinearBezierCurve(Vector2 start, Vector2 end, float t)

    - public static Vector2 QuadraticBezierCurve(Vector2 start, Vector2 curve, Vector2 end, float t)
    - public static Vector2 QuadraticBezierCurve(Vector2 start, Vector2 curve, Vector2 end, float t, ref float radians)
    - public static Vector2 CubicBezierCurve2(Vector2 start, Vector2 startPoitsTo, Vector2 end, Vector2 endPointsTo, float t)
    - public static Vector2 CubicBezierCurve2(Vector2 start, Vector2 startPoitsTo, Vector2 end, Vector2 endPointsTo, float t, ref float radians)
    - public static Vector2 CubicBezierCurve2(Vector2 start, float startPointDirection, float startPointLength, Vector2 end, float endPointDirection, float endPointLength, float t, ref float radians)
    - public static Vector2 CubicBezierCurve(Vector2 start, Vector2 curve1, Vector2 curve2, Vector2 end, float t)
    - public static Vector2 CubicBezierCurve(Vector2 start, Vector2 curve1, Vector2 curve2, Vector2 end, float t, ref float radians)
    - public static Vector2 InterpolateNormal(Vector2 vector1, Vector2 vector2, float t)
    - public static void InterpolateNormal(Vector2 vector1, Vector2 vector2, float t, out Vector2 vector)
    - public static void InterpolateNormal(ref Vector2 vector1, Vector2 vector2, float t)
    - public static float InterpolateRotation(float radians1, float radians2, float t)
- **Added Body.ClearImpulse()**
    - This is similar to ClearForce() and ClearTorque()
    - When you create a pool of bodies this helps "Reset" the body when it is released to the pool.
- **Added Body.ResetDynamics()**
    - This method will reset position and motion properties of the body.
    - Useful when creating pools of re-usable bodies. It would be used when releasing bodies back into a pool.

FarseerPhysics 1.0.0.4

- Small change to Vertices.Translate(). Given a vector, this method was doing a subtraction for the translation where it should have been adding. If you were using this method you will need to negate the vector you were passing in to this method.

- Changed Vertices.CalculateMomentOfInertia() to Vertices.GetMomentOfInertia() to match naming convention used by other methods in the class. If using this method you will need to do a global find/replace.

- Changed Vertices.GetMomentOfInertia to center the verts on its centroid prior to calculating the moment of inertia. It made more sense to do this. The underlying vertice values are not changed permanently by this method only for the calculation.

- Added BodyFactory.CreatePolygonBody(..., Vertices vertices, ...) overload. This method takes a set of vertices and uses them to calculate the moment of inertia so the user doesn't need to do this themselves.
    - Example: the code below will create a body with it's MomentOfInertia automatically calculated to be that of the triangle defined by the vertices passed

in. In the past you would have had to call verts.CalculateMomentOfInertia, then set that explicitly on the Body after it was created.

```
verts = new Vertices();
verts.Add(ConvertUnits.ToSimUnits(200, 200));
verts.Add(ConvertUnits.ToSimUnits(100, 200));
verts.Add(ConvertUnits.ToSimUnits(100, 300));

testBody = BodyFactory.Instance.CreatePolygonBody(physicsSimulator, verts, 1);
```

- Changed GeomFactory.CreatePolygonGeom(..). This method will now automatically align the centroid (center of mass) of the geom with the associated body position unless an offset is provided in which case the geom's centroid will be offset from the body by the amount of the offset.
    - Previously this method caused a lot of confusion if the vertices were not defined relative to 0,0. With this change, verts can now be defined anywhere and this method will fix up the verts so they act as though they were created relative to 0,0.

- Changed the way Body rotation, totalRotations, and revolutions are tracked and clamped.
    - This change was suggested by 'yota' in the CodePlex Issue Tracker.
    - The rotation of a body is now clamped as follows: 0 <= rotation < 2Pi. (It use to be -2Pi < rotation < 2Pi)
    - In addition, the revolutions are more accuratly tracked if a large rotation value is passed in to the Body.Rotation property. Thanks again to yota for this fix.
    - Finally, Body.revolutions, which was a private internal variable in Body now has a property accesor.

- Fixed linear drag bug found by michael_brooks11 from the codeplex forums.
    - Linear drag was not being computed properly.
    - See this post for more info: https://www.codeplex.com/Thread/View.aspx?ProjectName=FarseerPhysics&ThreadId=21087
    - You may need to adjust existing 'LinearDrag' settings for your bodies due to this change.

----------ARCHIVED REVISION HISTORY----------

FarseerPhysics 1.0.0.3

-SMALL BREAKING CHANGE (eassy to fix up)
--Added a FrictionType property to PhysicsSimulator. You can now choose between FrictionType.Minimum and FrictionType.Average. The first will cause the fiction between 2 bodies to be the min of the two, the second will use the average. In previous versions only minimum was used. Now the engine defaults to use average. If you want the engine to work the way it has been, simply set this property to minimum.
-Added BioSlayers fix to SelectiveSweep.
-Changed the Arbiter to take a PhysicsSimulator rather than a bunch of fields. All the fields were in the PhysicsSimulator.

FarseerPhysics 1.0.0.2

-BREAKING CHANGES
--Changed Body.GetVelocityAtPoint to Body.GetVelocityAtLocalPoint. This was to differentiate if from the new method Body.GetVelocityAtWorldPoint

-Added Body.GetVelocityAtWorldPoint and Body.ApplyForceAtWorldPoint methods.
-Update Broadphase Collision As follows:
--Add IBroadPhaseCollider interface
--Created a BruteForceCollider that implements IBroadPhaseCollider.
--Added SweepAndPruneCollider that implements IBroadPhaseCollider (Thanks Andy Jones)
--Added SelectiveSweepCollider that implements IBroadPhaseCollider (Thanks BioSlayer)
--Made SelectiveSweepCollider the default. While I found the SweepAndPruneCollider faster in most cases, SelectiveSweep seems for forgiving in general situations.
--Added SetBroadPhaseCollider(IBroadPhaseCollider) to PhysicsSimulator.
--All the broad phase stuff is still somewhat in flux. It could change again in the future.

-Added a Demo8: Broad Phase Collision Stress Test.
-Renamed Calculator.PiX2 to Calculator.TwoPi to be consistent with XNS math library.
-Added FluidDragController to simulate rigid bodies in water or other liquids.
-Added a number of LineIntersection methods. Used to test intersection two lines or lines an Geoms. The methods can be found in the new CollisionHelper class. (Thanks to Jeremy Bell for the code.)

FarseerPhysics 1.0.0.1

BREAKING CHANGE:
-Geometry object name was changed to Geom. I know this might cause some temporay headaches but 'Geometry' conflicts with a Geometry object in Silverlight and long term I think this is the right decision.

-PhysicsSimulator
Added new SweepAndPrune BroadPhaseCollision option. This option can be toggled by setting PhysicsSimulator.BroadPhaseCollider.
Default is brute force until SweepAndPrune can be optimized.

-PhysicsSimulator.ProcessRemovedItems:
Added code to remove any arbiters associated with geometries that are being removed from the simulator.

-PhysicsSimulator.ProcessAddedItems:
Changed the order the ProcessXXX methods were called. Put ProcessAddedItems 1st to avoid confusion when add/remove is called without calling update in between.

-Mathematics.Calculator:
Added DegreesToRadians method.

-Collision.Vertices
Added CalculateMomentOfInertia() to make it easy to get the moment of inertia for a polygon
Fixed a nasty collision bug that would appear with semi-complex geometries.

-Global
Did some performance tuning by replacing some existing method calls with inline logic.

-RevoluteJoint
Changed default biasFactor to .2 from .8. For stability purposes.

-Body
Changed the ApplyImpulse logic to store impulses applied externally and have the PhysicsSimulator apply the summed impules all at once.
Joints and Arbiter now call internal method "ApplyImmediateImpulse"

FarseerPhysics 1.0.0.0

-First Release