



Useful Tools for Making Video Games

Part III

An overview of





Features

- easy to use C-API
- available for windows, osx, linux
- plethora of convex collision shapes
- compound collision shapes
- continuous collision mode
- hinge, ball, slider, corkscrew,... and custom joints
- special vehicle container
- special [ragdoll](#) container



Startup sequence (simplified)

```
// create the newton world
NewtonWorld* nWorld = NewtonCreate(...);

// Create the collision shape
NewtonCollision* collision = NewtonCreateBox (...);

// Create the rigid body
NewtonBody* rigidBodyBox = NewtonCreateBody (nWorld, collision);

// set the body mass and inertia
NewtonBodySetMassMatrix (rigidBodyBox, ...);

// Set the transformation matrix
NewtonBodySetMatrix (rigidBodyBox, ...);

// Animate the body eg. By setting angular velocity
NewtonBodySetOmega (rigidBodyBox, ...);

// in game loop update timer, get new transformation matrix
NewtonUpdate (nWorld, timeStep);
NewtonBodyGetMatrix(rigidBodyBox, ...); // use this matrix to render a box
```



Using Callbacks

- At each update apply force and update transformation of rendering primitive

```
// at body creation
```

```
NewtonBodySetTransformCallback (boxBody, transCallback);
```

```
NewtonBodySetForceAndTorqueCallback (boxBody, forceCallback);
```

Where

```
void transCallback (const NewtonBody* body, const dFloat* matrix) {...}
```

```
void forceCallback(const NewtonBody* body) {...}
```



Materials

- Assign physical properties to Newton bodies such as friction, softness, elasticity etc.
- You need to set the material interactions for every combination of any two materials in your scene!

```
// create material
```

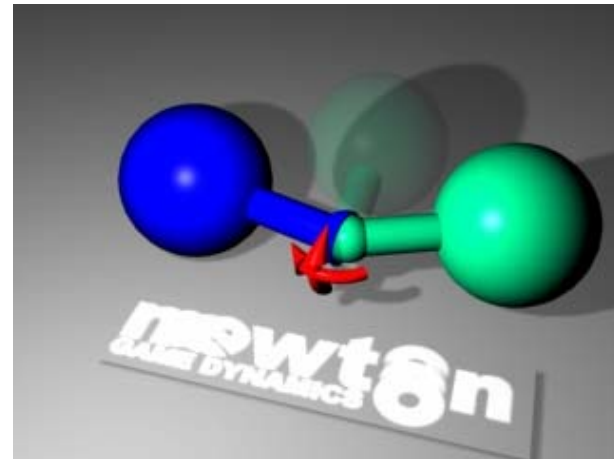
```
Int materialId = NewtonMaterialCreateGroupID(nWorld);  
NewtonMaterialSetDefaultSoftness (nWorld, defaultID, ...);  
NewtonMaterialSetDefaultElasticity (nWorld, defaultID, ...);  
NewtonMaterialSetDefaultCollidable (nWorld, defaultID, ...);  
NewtonMaterialSetDefaultFriction (nWorld, defaultID, ...);  
NewtonMaterialSetCollisionCallback (nWorld, defaultID, ...);
```

```
// set material to body
```

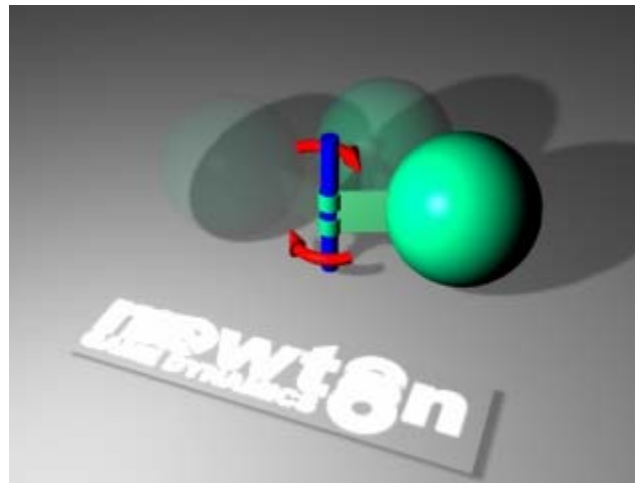
```
NewtonBodySetMaterialGroupID (boxBody, materialId);
```

Joint and Constraints

- Ball and Socket

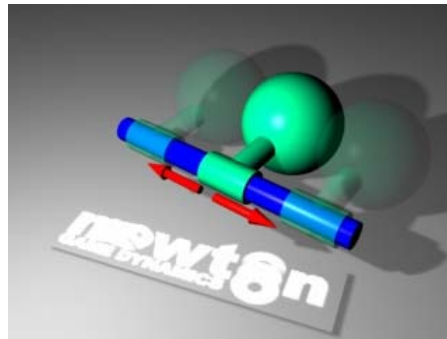


- Hinge

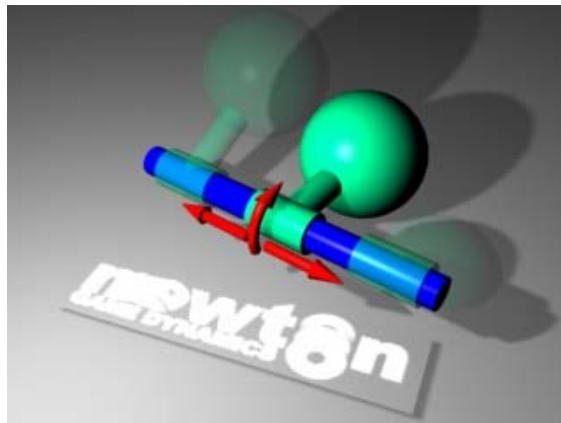


Joints and Constraints

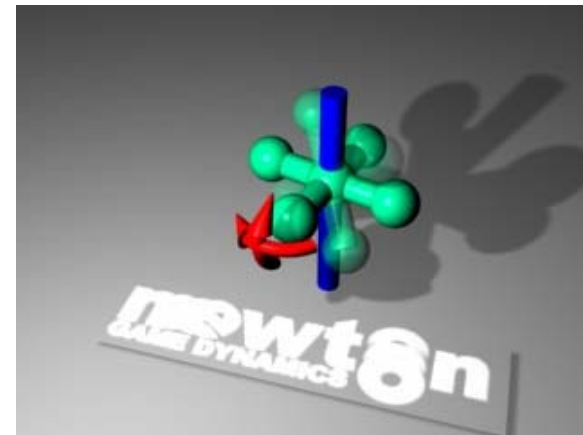
Slider



- Corkscrew



Universal



Joints and Constraints

NewtonJoint* joint =

```
NewtonConstraintCreate[Ball/Hinge/Slider/Corkscrew/Universal] (...,  
body1, body2);
```

```
Newton[Ball/Hinge/Slider/Universal]SetUserCallback (joint, callbackFunc);
```

- Also need to set limits, pivot, stiffness, friction depending on the type of joint
- for details look at `samples\tutorial_05_UsingJoints`
- Don't forget to check out `NewtonRaceCar` (`samples\tutorial_09_SimpleVehicle`)





Rendering Engine Integration

- Already done!
 - [OgreNewt](#)
 - [IrrNewt](#)
- Takes care of most of setting up, updating, allocation/deallocation and other details



References

- <http://www.newtondynamics.com/>
- <http://www.newtondynamics.com/forum>
- <http://aresfps.sourceforge.net/irrnwtt.htm>
- <http://walaber.com/index.php?action=showitem&id=9>