Animation tutorial

Bullet Physic Engine

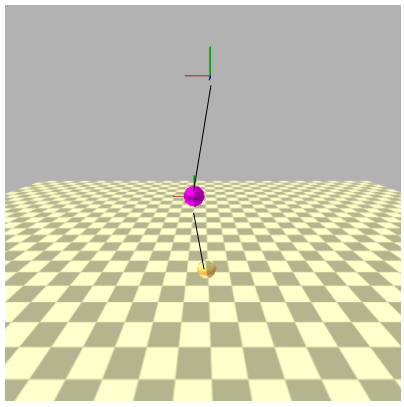
From a pendulum to flying trapeze

The aim of this tutorial is to discover the possibilities of the bullet engine. We will build a simple pendulum scene and we will progressively change it into a flying trapeze.

General advice:
The bullet website (http://bulletphysics.org) is your best friend. Use the API reference, the wiki and the demonstration files given with the Bullet SDK. For this tutorial the sources of the SoftDemo application will be a good help and inspiration.

A. Download the source code to start the tutorial (the teacher will tell you where download it) and build your own Bullet project with Code::Block or with any other solution if you want to.

B. Double pendulum with rigid constraints
This exercise must be done inside the function:
```
void PendulumApplication::buildDoublePendulumRigid()
```



1. Create a sphere with a radius of 1 and a mass of 1.
   Useful function:
   ```
   btRigidBody*
          DemoApplication::localCreateRigidBody(...)
   ```

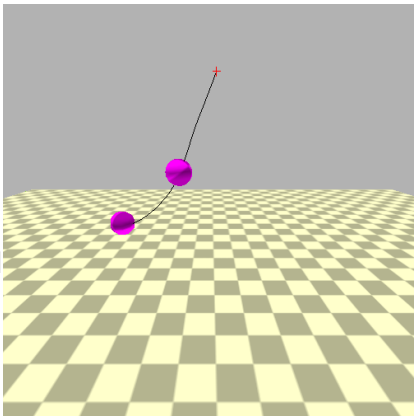
2. Create a position constraint on the sphere with the `btPoint2PointConstraint` object. Use the "C" keyboard shortcut to display the constraint rotation pivots. Apply an impulse to the sphere to make it move.
   Useful function:
   ```
   void btRigidBody::applyCentralImpulse(...)
   ```

3. Create a second sphere and a second constraint that link the first sphere with the second.

4. Set a breaking threshold to the constraints.
   Useful function:
   ```
   void btTypedConstraint::setBreakingImpulseThreshold(...)
   ```

5. Change the mass of the spheres and the length of the constraints to see how your pendulum reacts.

C. Double pendulum with soft constraints
In this exercise we will replace the constraints by ropes.
This exercise must be done inside the function:
```
void PendulumApplication::buildDoublePendulumSoft()
```



1. Create a sphere and a rope linked to this sphere.
   Useful functions:
   ```
   btSoftBody*   btSoftBodyHelpers::CreateRope(...)
   void btSoftBody::appendAnchor(...)
   ```

2. Create a second sphere and a second rope linked

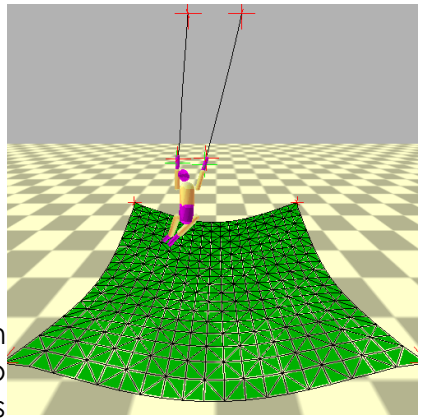
to the first sphere and this second sphere.
Useful functions:
```
btSoftBody*   btSoftBodyHelpers::CreateRope(...)
void btSoftBody::appendAnchor(...)
```

3. Try to find out an alternative solution to the `setBreakingImpulseThreshold()` function which doesn't exist for a rope.
Useful function:
```
bool*  btSoftBody::cutLink(int node0,int node1,btScalar position)
```

Advice: update the m_cameraTargetPosition attribute in the `void PendulumApplication::renderme()` *function in order to follow the action with the camera.*

D. Flying trapeze

This exercise must be done inside the function:
```
void PendulumApplication::buildFlyingTrapeze()
```



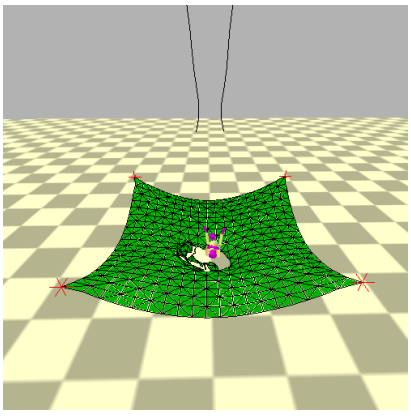
1. Create a rag-doll by using the `RagDoll` class.

2. Attach each lower arm of the rag-doll to a rope.
Useful function:
```
btRigidBody* RagDoll::getBodyPart(...)
```

3. Animate the rag-doll's legs with the keyboard in order to make the trapeze artist swing. One key to swings the legs forward and another key to swings the legs backward.
Useful function:
```
btRigidBody::applyTorqueImpulse(...)
```

4. Create a safety net under the ragdoll. The size of the patch, the resolution, the setting of the material and threshold of collision detection should be chosen wisely to obtain a credible result.
Useful functions:
```
btSoftBody* btSoftBodyHelpers::CreatePatch(...)
btSoftBody::appendMaterial() ,
btSoftBody::Material::generateBendingConstraints(...) ,
```

5. Find a way to make the trapeze artist release the rope when the 'x' key is pressed.



6. Let's imagine the worst case scenario. The trapeze artist fall down on the safety net but the safety net breaks on impact.
Useful functions classes and attributes:
```
void btSoftBody::refine()
class btSoftBody::ImplicitFn
tRContactArray btSoftBody::m_rcontacts
```

Mouse commands

Mouse and keyboard	Function
Click Left Mouse	Grasp the pointed object
Click Right Mouse	Throw a cube
Ctrl+Click Left Mouse	Rotate the camera
Ctrl+Click Right Mouse	Zoom camera
Ctrl+Click Middle Mouse	Translate the camera

Camera commands

Keyboard	Fist letter of	Function
l	Left	Rotate the camera to the left
r	Right	Rotate the camera to the right
f	Front	Rotate the camera up
b	Back	Rotate the camera down
z	Zoom In	Zoom in
x	-	Zoom Out
o	Orthogonal	Switch projection (perceptive/orthogonal)

Display Commands

Keyboard	Fist letter of	Function
g	-	Display/hide projected shadows
u	-	Display/hide textures
w	Wire	Wire display
h	Hide	Display/hide debugging information
a	AABB	Display/hide axis aligned bounding boxes
c	Collision	Display/hide collisions
C	Constraint	Display/hide constraint pivots
L	Limits	Display/hide constraint limits

Other commands

Touche clavier	Fist letter of	Function
+	Plus	Increase the speed of the launched cube
-	Less	Decrease the speed of the launched cube
i	Idle	Stop the simulation
s	Step	Compute one single step of simulation
space	-	Restart the application
q	Quit	Quit the application