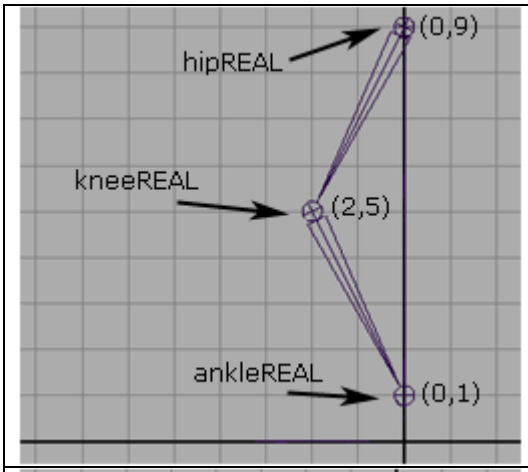
Elastic Foot using Nodes

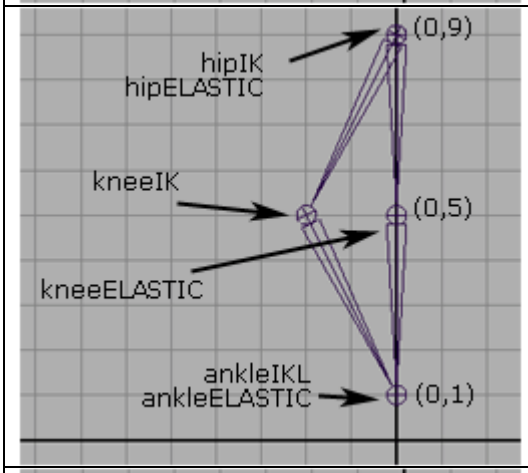
Cartoon animation looks great with some squash and stretch.
To be able to 'stretch' bones you need a set-up that can handle 'elasticity'.

This tutorial shows how to build an elastic foot using nodes. This version is a little bit more complex than the expression one, but it shows how to work with different nodes types in Maya. Knowing how to use nodes is a great way to expand your Maya knowledge in addition of making your scene run a lot faster**.**



Bones
Create the ***REAL*** bones using the grid.

hipREAL->kneeREAL->ankleREAL
in position (0,9 – 2,5 - 0,1) relative to the grid.



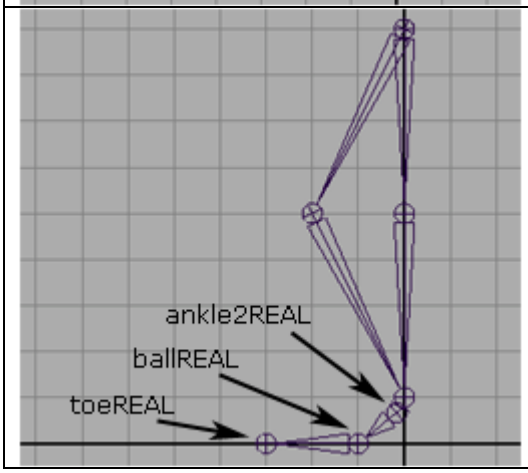
IK and ELASTIC Bones
Create ***IK*** bones by duplicating the ***REAL*** ones.

hipIK->kneeIK->ankleIK

Create the ***ELASTIC*** bones using the grid.

hipELASTIC->kneeELASTIC->ankleELASTIC
in position (0,9 – 0,5 – 0,1)

Create an ikRPsolver form hipIK to anleIK
Name the IK handle:
ikHandle

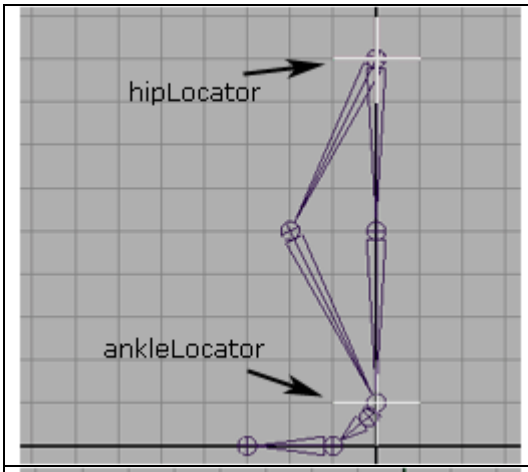


More Bones
Create the ***REAL*** foot

ankle2REAL->ballREAL->toeREAL

The ***ankle2REAL*** is used for skinning purposes so that the foot doesn't stretch when it's in elastic mode. That's why it should be close to the ***ankleREAL***.

Parent ***ankle2REAL*** to ***ankleREAL*** to connect the foot to the leg.

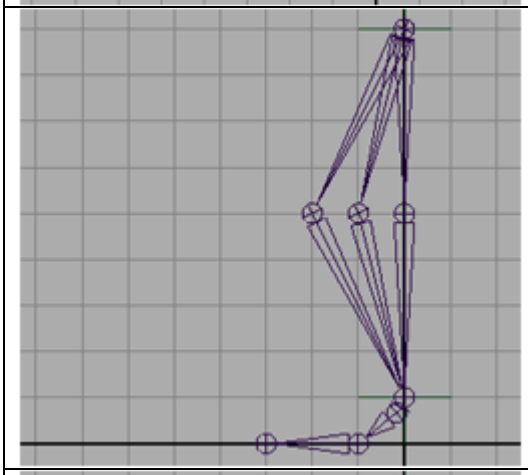


Locators
Create a locator and place it on the same position as the hips (0,9).

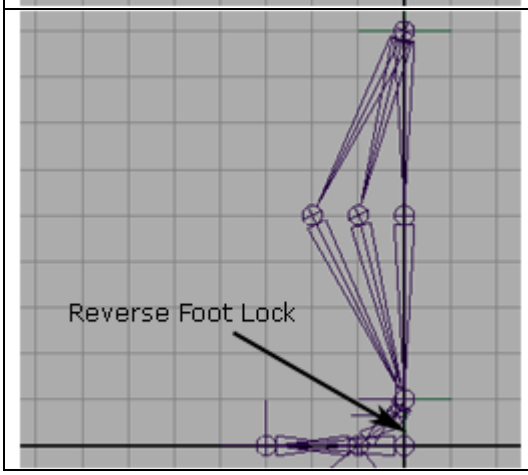
locatorHip

Create another locator and place it on the same position as the ankle (0,1).

locatorAnkle



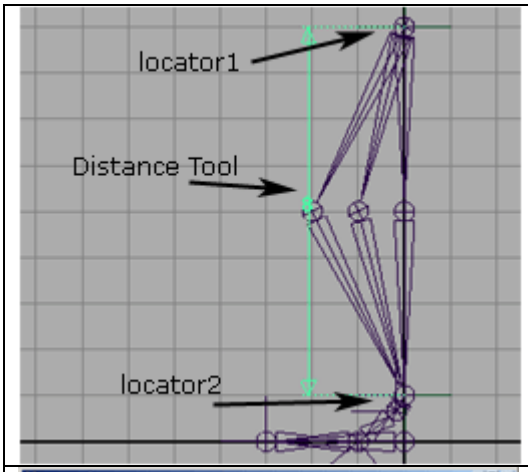
Constrains
Point constrain all the hips to ***locatorHip***.
Point constrain ***ankleREAL***, ***ankleELASTIC*** and ***ikHandle*** to ***locatorAnkle***.
Point constrain ***kneeREAL*** to both ***kneeIK*** and ***kneeELASTIC***.
Point constrain ***kneeELASTIC*** to both ***hipLocator*** and ***ankleLocator***. (this way the ***kneeELASTIC*** will always be half way between the locators).
Orient constrain ***hipREAL*** to ***hipIK*** and ***kneeREAL*** to ***kneeIK***.



Reverse Foot Lock

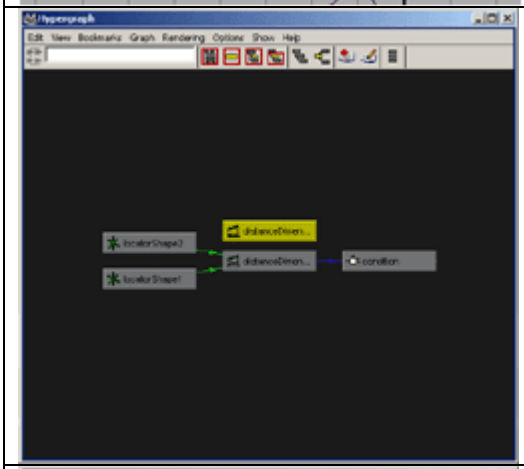
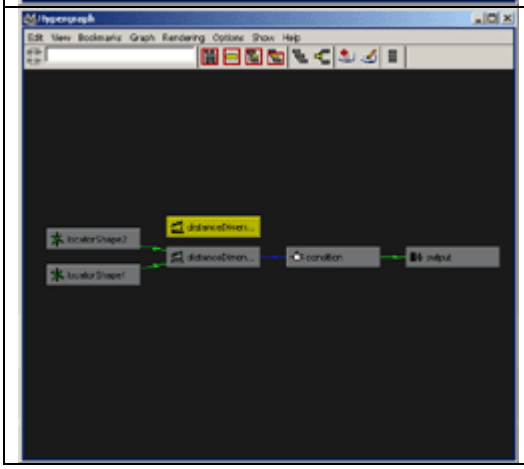
Create a reverse foot lock. (see Reverse Foot Lock tutorial)

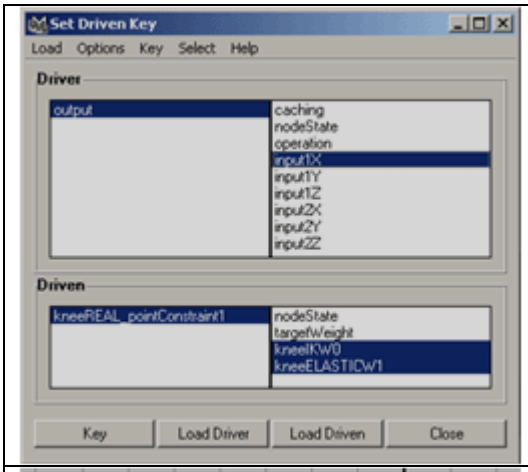
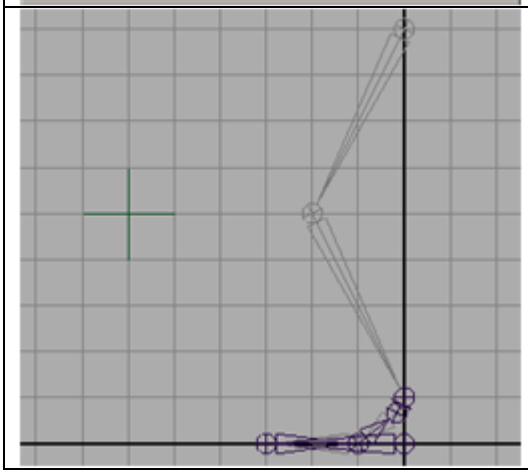
Compensate by creating an additional ***ankle2RF*** for the extra ankle joint.

<table>
<tr>
<td> 
</td>
<td>Measure Distance
Create a distance tool.

Point constrain locator1 (from the distance tool) to hipLocator and locator2 to ankleLocator.

Because we used the grid to create the bones, the distance tool show measure 8 units.
When the knee is straight, the distance will be 9 units long.

(If you didn’t use the grid, you must find out what the distance is when the leg is stretched)</td>
</tr>
<tr>
<td>  </td>
<td>Condition Node
Go into the hypergraph and select the distanceDimension1 node.
Click on “Up and Downstream Connections”.
Create a condition node by typing in the command line:
createNode condition –n “condition”;
Connect the ‘distance’ from distanceDimensionShape1 to the ‘firstTerm’ in condition.
Go into the attributes (Alt-A) for condition.
Change Second Term to 9, the operation to “Greater or Equal” and Color1 to 1 and Color2 to 0.</td>
</tr>
<tr>
<td>  </td>
<td>MultiplyDivide Node
Create a multiplyDivide node by typing in the command line:
createNode multiplyDivide –n “output”;
connect the outColor from condition to input1 in output.</td>
</tr>
</table>

<table>
<tr>
<td> 
</td>
<td>Set Driven Keys
Set output as the Driver and kneeREAL_poinConstraint1 as the Driven.
Select input1X as the Driver in output and key kneeIKW0=1 and kneeELASTICW1=0.
Translate the Reverse Foot Lock down (so that you stretch the joint)
This way the condition will output 1 in output.input1X.
Key kneeIKW0=0 and kneeELASTICW1=1.</td>
</tr>
<tr>
<td>  </td>
<td>Done
What’s left to do is just clean the hypergraph so that everything is nice and tiddy.

Now, grab the Reverse Foot Lock and move the foot around.</td>
</tr>
</table>