An Approach to Content Creation for Trainz

© Paul Hobbs and
Peter Pardoe-Matthews

Part 8

Animating Valve Gear Using Bones

(Updates and sample files available from http://www.44090digitalmodels.co.uk)

Index

Foreword.

Peter Pardoe-Matthews was kind enough to supply some notes on how he goes about animating outside valve gear on his models and this document is an attempt to create a layman's guide on how to do it. This method is something I hadn't tried before so we'll all (apart from Peter) be learning something new.

The original notes were a bit technical and this document will hopefully be easier to follow.

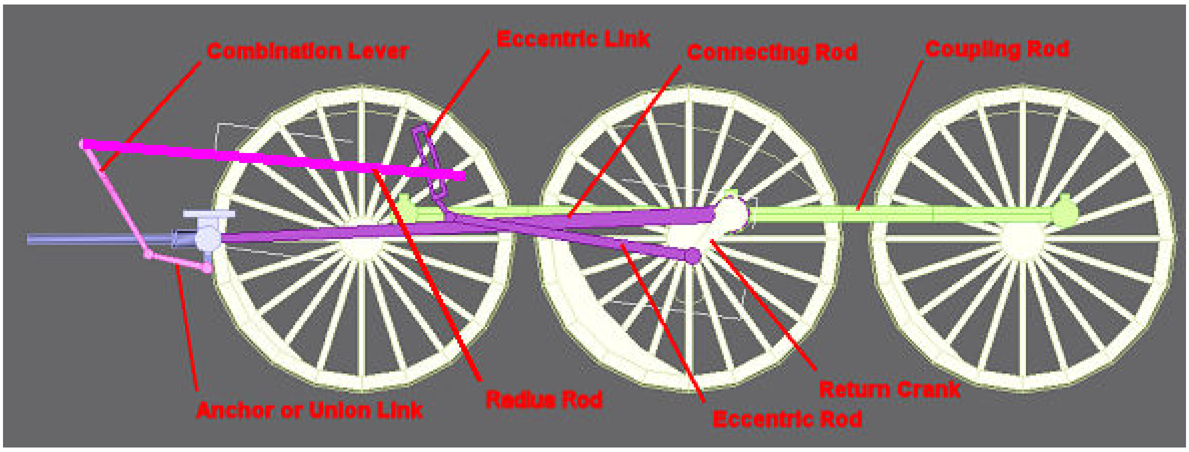
A bone is effectively a set of levers that are joined together. GMax calls this an ‚IK LIMB Bone'. The IK LIMB Bone is controlled by a thing called an IK Solver – this controls the positions of the Bones relative to each other. That's enough of the technical stuff – there's a lot more on the theory in the GMax help files which I looked at and it scared the hell out of me and I do this sort of thing for a living.

An ‚IK LIMB Bone' consists of two or three elements which is enough for all we have to do. Naming of bones is important, 'b.r.bone name' works well and ensures that the bones are not confused with dummies. As you will see in the animation, mesh objects can be linked to bones and bones can be linked to dummies.

Your bone making work will be much easier if you place helper points at each link point, this ensures greater accuracy of bone positioning.

Overview of Walschaerts Valve Gear

Before we start to animate the valve gear we'll need to get the naming of the various parts straight. We'll be assuming Walschaerts valve gear which was much the most common type although the technique can be used for any valve gear.



By the way, when the loco is in forward gear the joint between the Radius Rod and the Eccentric Link is below the pivot point of the Eccentric Link, and when in reverse gear above it. In real models (and most virtual ones!) a common dodge is to model the gear in 'neutral', i.e. the Radius Rod pivot is coincident with the Eccentric Link pivot. This means that the Radius Rod doesn't move which is a pity.

Bones must be created in the following order for one side only. The example assumes right view, being left side of loco looking forward

The following is a listing of the important locations we'll be using when doing the animation.

.AL	Expansion Link rotation point
.BL	Eccentric Rod to Expansion Link pivot point
.CL	Eccentric Rod to Return Crank pivot point
.DL	Radius rod to Expansion Link connection point
.EL	Radius rod to Combination Lever pivot point
.FL	Combination Lever to Union Link pivot point
.GL	Union Link to Crosshead pivot point
.HL	Point on centre line of piston rod
.IL	Crosshead to connecting rod pivot point
.JL	Connecting Rod to crank pin on wheel

Creating a Bone System

Make sure that the number of animation frames is set to 30 as described in Part 5 and practice bone making process before starting on the main model. After selecting “Animation -> Create Bones” for the first time in a session, select “Assign to Children” in side menu, and change both bone dimensions to 10 inches or metric equivalent.

Bone making process is easy. Place cursor, left click, release, left click, drag, release, left click, drag, release, right click. Practice this until it is second nature.

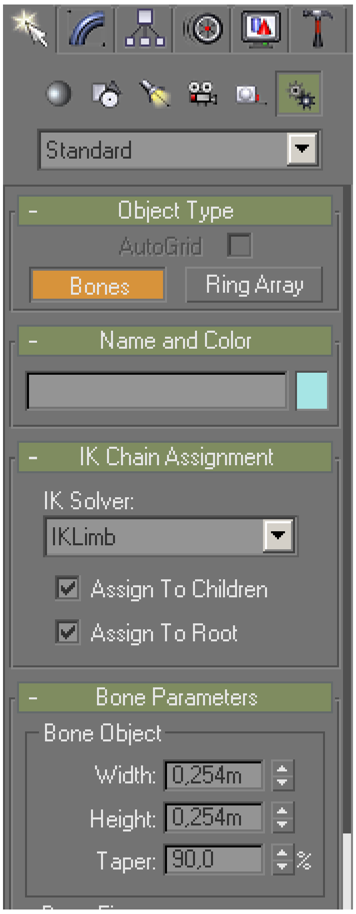
On to creating the first bone system for our model, hide all the rods and valve gear on the right side to avoid confusion.

You can download the bogey here to try out this tutorial.

Creating the Helper Points

The first step is to create helper points at the positions shown in the table for the left side of the valve gear. I then move the points in X by 2 metres so that they are away from the actual rods

The naming ‚.AL’, .BL’ is to ensure that they appear at the top of the ‚Select Object’ lit when you press ‚H’.



AutoGrid □

IK Solver: IKLimb

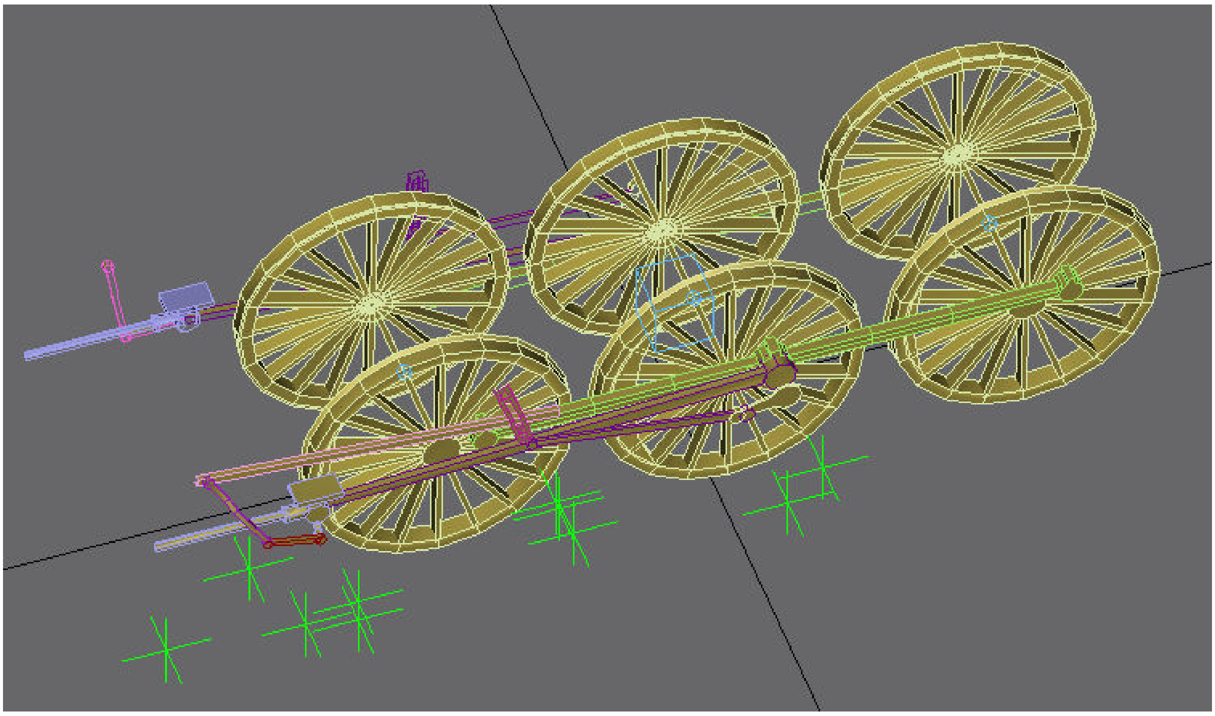
☑ Assign To Children

☑ Assign To Root

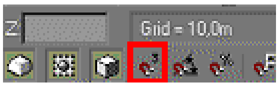
Width: 0,254m

Height: 0,254m

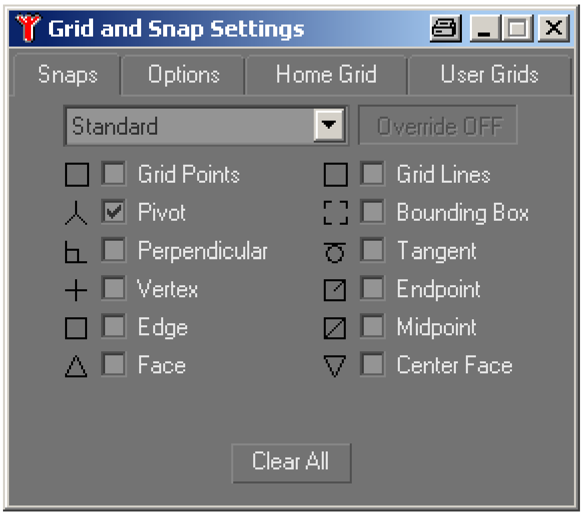
Taper: 90,0 %



Animating the Eccentric Link and Eccentric Rod



Next, and (very important!) click on the ‚2D snap toggle’ and set to snap to ’Pivot’ and nothing else. Make sure that the icon is selected so that the bones snap to the helper points. **When you’ve finished creating the bones don’t forget to turn the 2D snap icon off again.**



Grid and Snap Settings

Snaps | Options | Home Grid | User Grids

Standard | Override OFF

□ Grid Points
☑ Pivot
□ Perpendicular
□ Vertex
□ Edge
□ Face
□ Grid Lines
□ Bounding Box
□ Tangent
□ Endpoint
□ Midpoint
□ Center Face

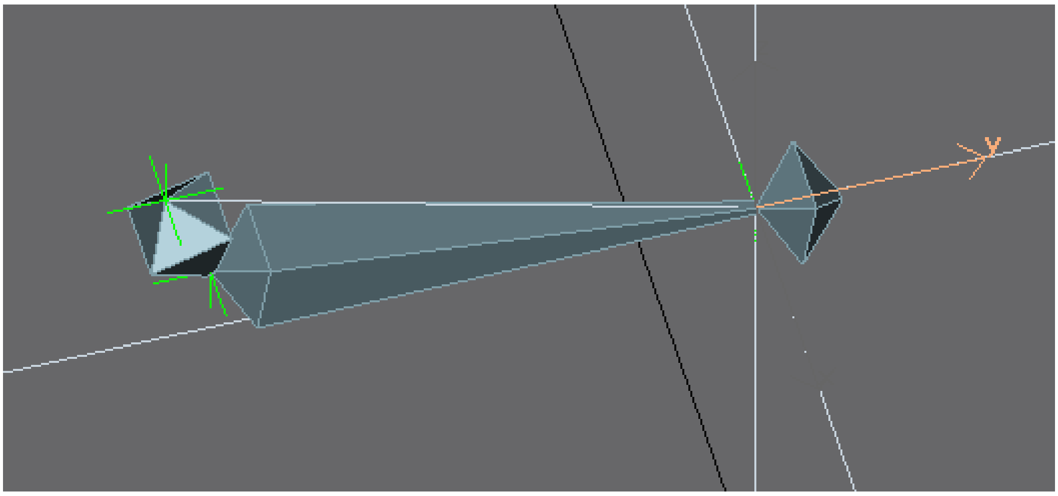
Clear All

The keyboard shortcut to toggle whether snapping is enabled or not is ‘S’.

It helps to hide everything but the points you need to avoid confusion.

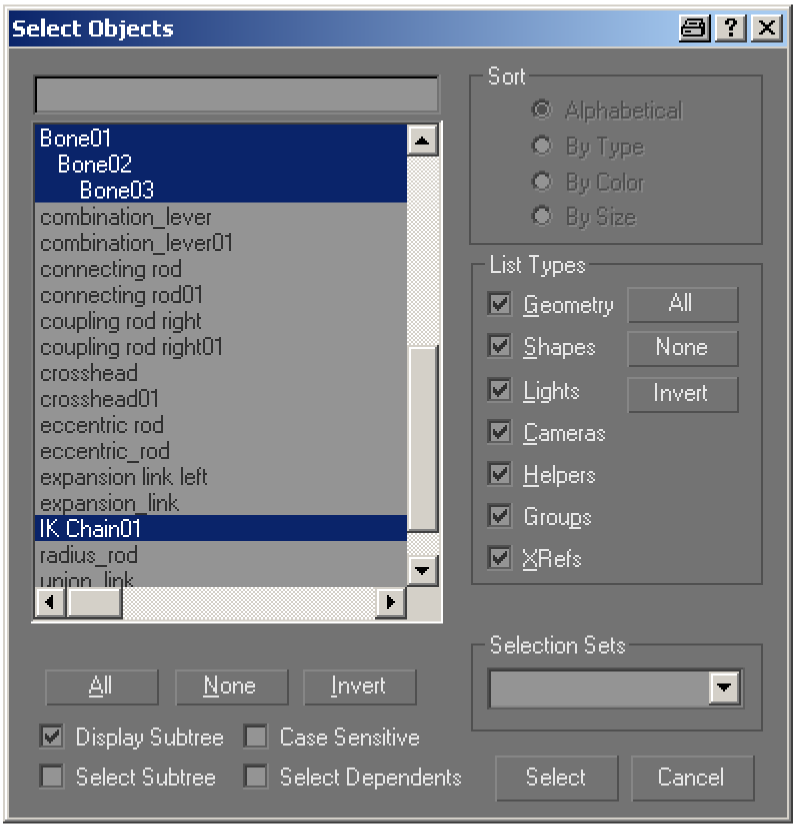
Make the first click at point .AL, release, left click drag to point .BL, release, left click drag to point .CL, release the left button and then right click.

The result should look like the picture on the next page. The cross at the end of the bone system is the IK Solver which is automatically created when the bone system is complete.



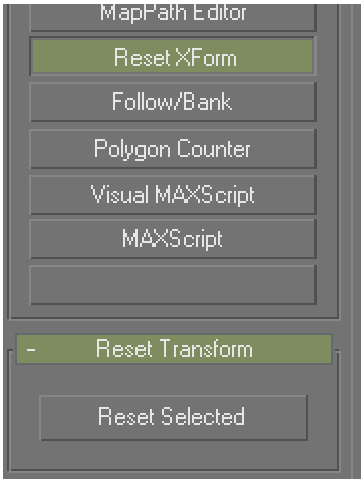
The newly created objects are shown highlighted here. ‚IKChain01' is the object that controls the movement of the bones relative to each other.

As we didn't create the bones in top view though, they need a ‚Reset Xform' to align them to world coordinates before we link anything to them. In fact ‚Reset Xform' on linked objects is a bad idea anyway – usually the objects jump away from the original position.



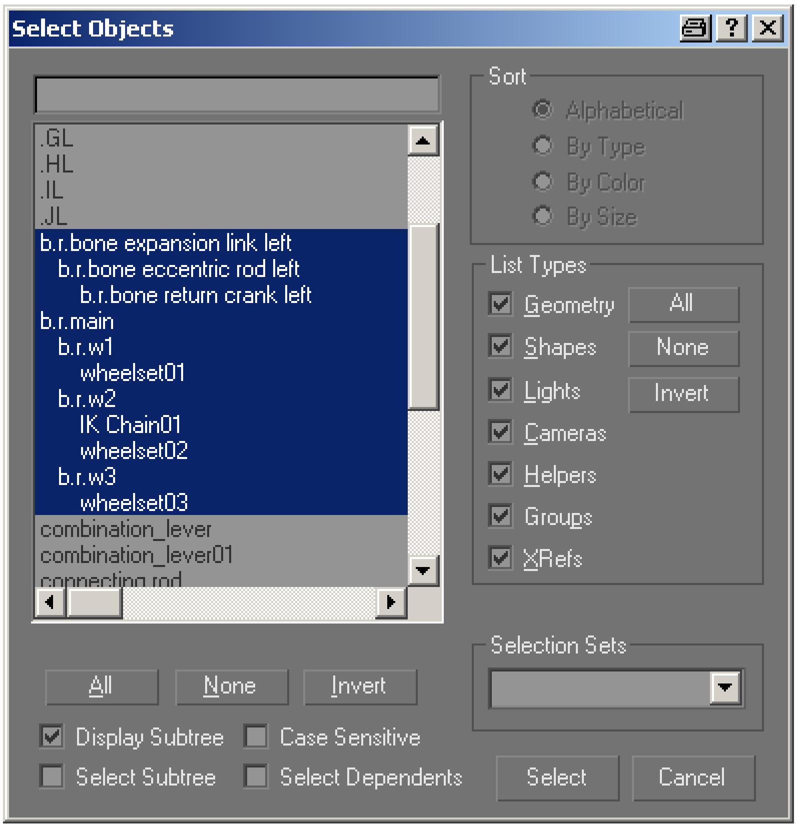
We have to unlink the bones, perform a ‚Reset Xform' on each of them and then relink them as they were before.

As an aside, creating the bones in top view doesn't help anyway, you still have to do a 'Reset Xform' on them to get them to align properly. In this respect bones seem to differ from other objects in GMax.



Rename ‚Bone01’ to ‚b.r.bone expansion link left’, ‚Bone02’ to ‚b.r.bone eccentric rod left’ and ‚Bone03’ to ‚b.r.bone return crank left’. **The ‚b.r.’ part of the name is important! Forget to rename them and the bones will not be exported.**

Animate the centre driver using an expression controller as detailed in Part 5, link the centre driver to it and then link ‚IK Chain01’ to the wheel dummy. The tree should look like this:



Select Objects

```
.GL
.HL
.IL
.JL
b.r.bone expansion link left
  b.r.bone eccentric rod left
    b.r.bone return crank left
b.r.main
  b.r.w1
    wheelset01
  b.r.w2
    IK Chain01
    wheelset02
  b.r.w3
    wheelset03
combination_lever
combination_lever01
connecting rod
```

Sort: ● Alphabetical ○ By Type ○ By Color ○ By Size

List Types: ☑ Geometry ☑ Shapes ☑ Lights ☑ Cameras ☑ Helpers ☑ Groups ☑ XRefs

☑ Display Subtree □ Case Sensitive □ Select Subtree □ Select Dependents

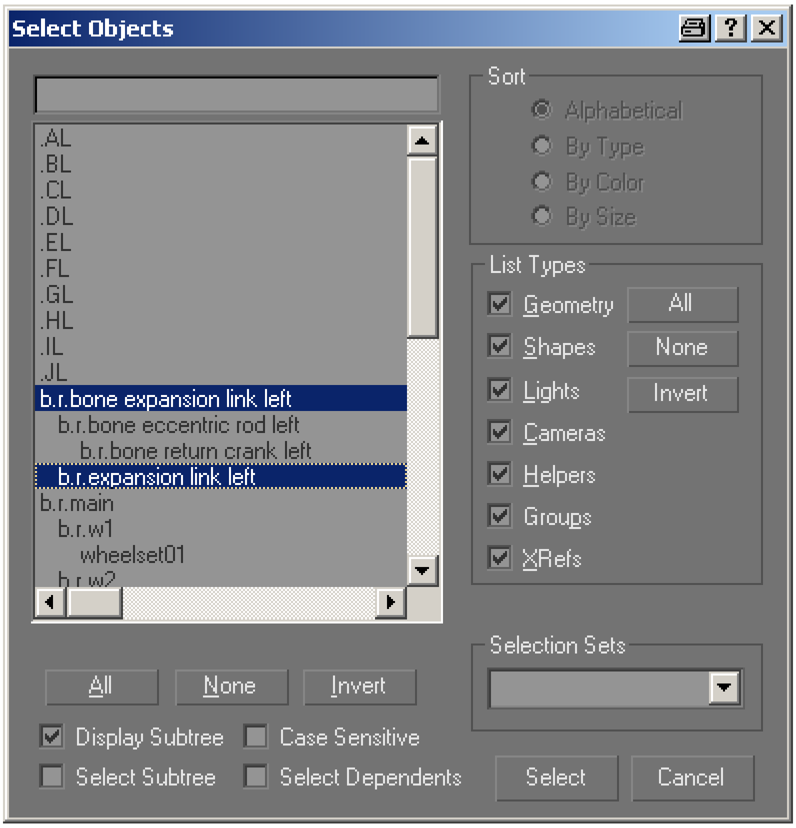
If you now play the animation you’ll see the bone system moving as if it were a set of linked levers.

The cunning part is that you can link the eccentric rod and the eccentric link to this bone system and it will move with it. Much easier than animating using dummies at each rod joint.

Animating the Radius Rod

Next create a dummy called ‚b.r.expansion link left’ at point .DL. Create the dummy itself in top view and then align it to point ‚.DL’. Link the dummy to ‚ b.r.bone expansion link left’.

.



Select Objects

```
.AL
.BL
.CL
.DL
.EL
.FL
.GL
.HL
.IL
.JL
b.r.bone expansion link left
  b.r.bone eccentric rod left
    b.r.bone return crank left
  b.r.expansion link left
b.r.main
  b.r.w1
    wheelset01
  b.r.w2
```

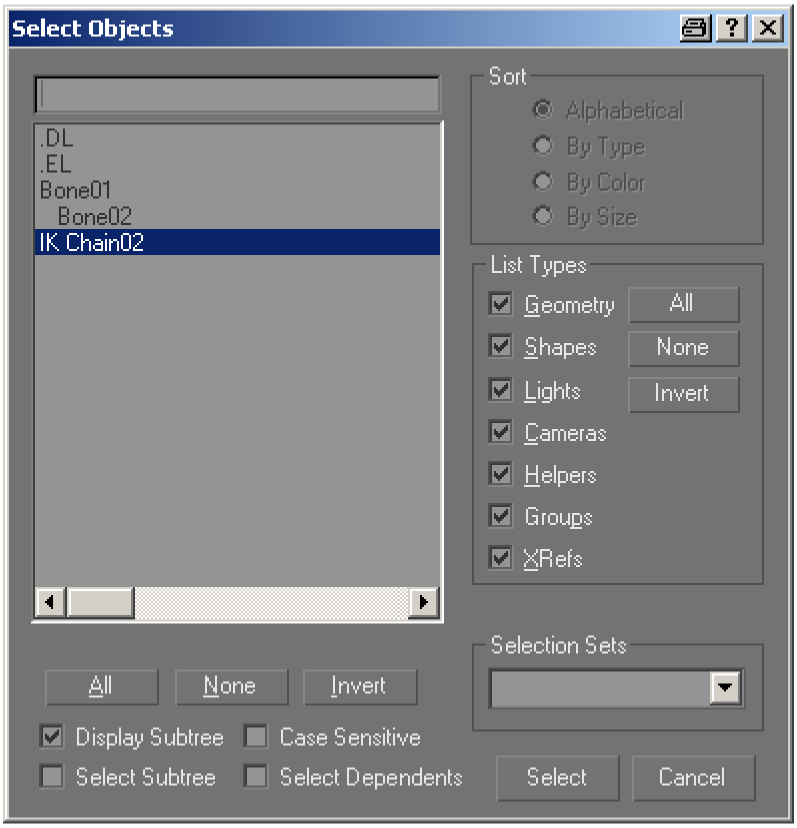
Sort: ● Alphabetical ○ By Type ○ By Color ○ By Size

List Types: ☑ Geometry ☑ Shapes ☑ Lights ☑ Cameras ☑ Helpers ☑ Groups ☑ XRefs

☑ Display Subtree □ Case Sensitive □ Select Subtree □ Select Dependents

Now create a bone from point ’.DL’ to ‚EL’.

Unlink ‚Bone02’, do a ‚Reset XForm’ on both ‘Bone01’ and ‘Bone02’ and then relink it as shown here.



```
.DL
.EL
Bone01
  Bone02
IK Chain02
```

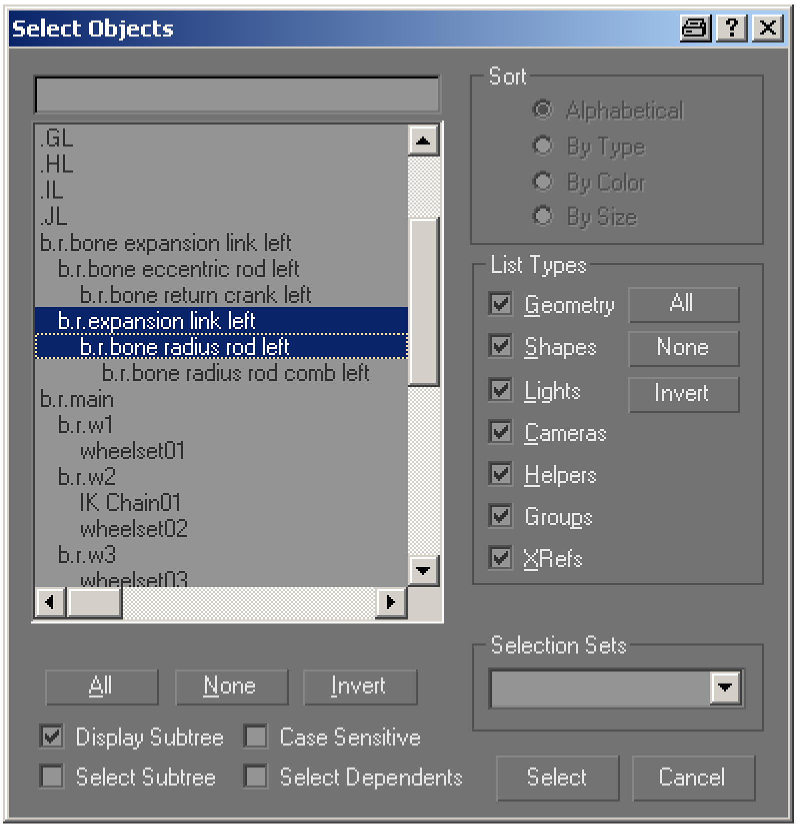
Sort: ● Alphabetical ○ By Type ○ By Color ○ By Size

List Types: ☑ Geometry ☑ Shapes ☑ Lights ☑ Cameras ☑ Helpers ☑ Groups ☑ XRefs

☑ Display Subtree □ Case Sensitive □ Select Subtree □ Select Dependents

Rename ‚Bone02’ to ‚ b.r.bone radius rod comb left’ and ‚Bone01’ to ‚ b.r.bone radius rod left’.

Link ‚ b.r.bone radius rod left’ to ‚b.r.expansion link left’.



```
.GL
.HL
.IL
.JL
b.r.bone expansion link left
  b.r.bone eccentric rod left
    b.r.bone return crank left
  b.r.expansion link left
    b.r.bone radius rod left
      b.r.bone radius rod comb left
b.r.main
  b.r.w1
    wheelset01
  b.r.w2
    IK Chain01
    wheelset02
  b.r.w3
    wheelset03
```

Sort: ● Alphabetical ○ By Type ○ By Color ○ By Size

List Types: ☑ Geometry ☑ Shapes ☑ Lights ☑ Cameras ☑ Helpers ☑ Groups ☑ XRefs

☑ Display Subtree □ Case Sensitive □ Select Subtree □ Select Dependents

Animating the Combination Lever and Union Link

In a similar manner create a bone system between points ‚.EL', ‚.FL' and ‚.GL'.

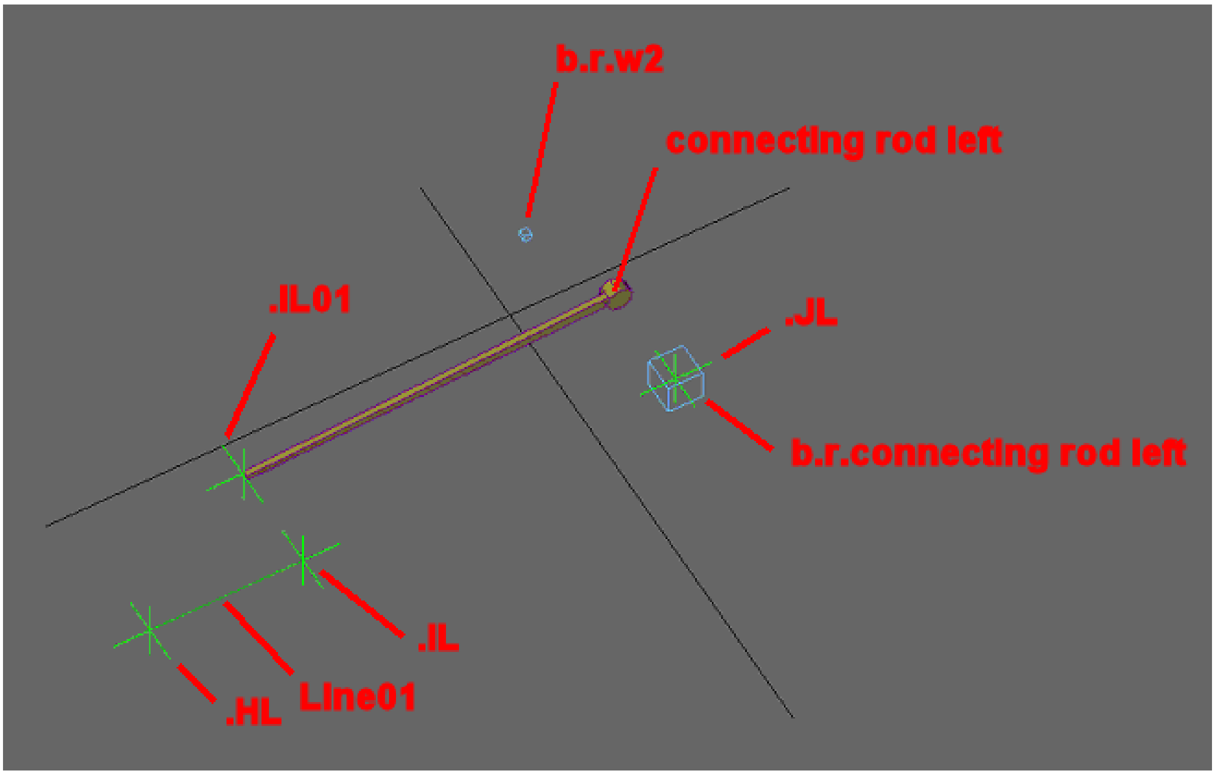
Do a ‚Reset Xform' on the new bones and link ‚Bone01' to ‚b.r.bone radius rod comb left.'

Rename ‚Bone01' to ‚ b.r.bone combination lever left', ‚Bone02' to ‚ b.r.bone union link left' and ‚Bone03' to ‚ b.r.bone union link x head left'. Create a new dummy in top view called ‚b.r.piston rod left' and align this to point ‚.IL'. Link ‚IK Chain03' to ‚b.r.piston left' and ‚b.r.piston left' to ‚b.r.connecting rod left'. Link ‚ b.r.bone combination lever left' to ‚ b.r.bone radius rod comb left'.

Animating the Connecting Rod and Crosshead the Hard Way

Now we have to animate the connecting rod and crosshead. You are probably wondering why I'm describing the hard way if there is an easier way and the answer is that in the past I've had problems with the easy method which now appear to be solved. The 'hard way' might still be useful in some circumstances though some I'm going to leave it in and suggest that in this case you skip ahead to the easy way and come back here if it doesn't work out.

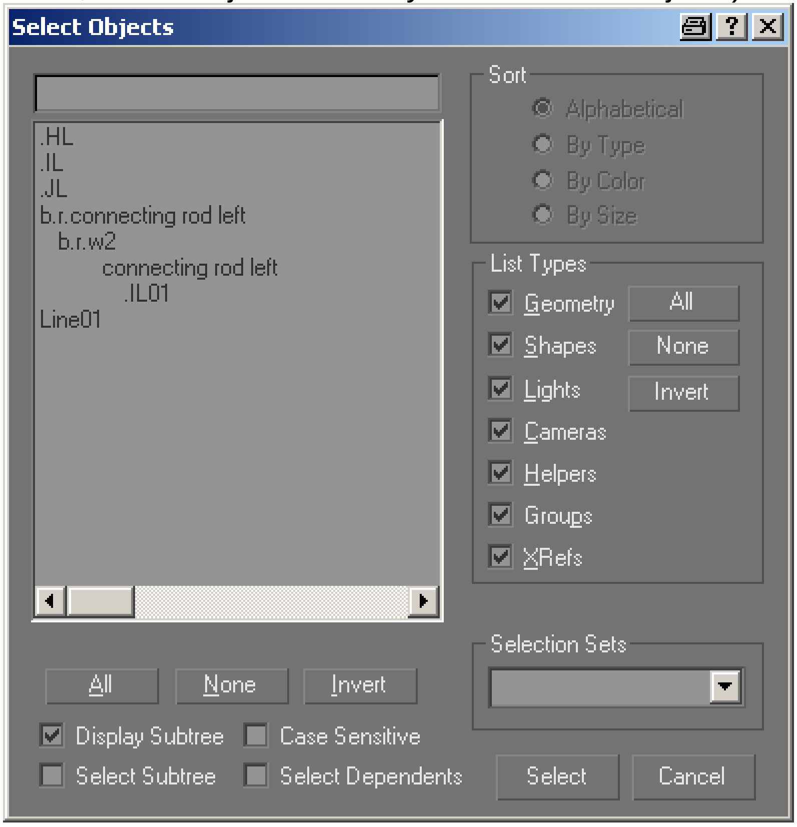
With this method it is necessary to create a dummy for the connecting rod and the crosshead and animating each object for each of the 30 frames. I used to do every rod like this(!)



It's not as bad as it sounds actually. Create dummy called ‚b.r.connecting rod left' in top view and align it in X,Y and Z to point ‚.JL'. Create a line ‚Line01' between ‚.HL' and ‚.IL' and copy ‚.JL' as ‚.JL01' . Link ‚b.r.connnecting rod left' to ‚b.r.w2', ‚connecting rod left' to ‚b.r.connecting rod left' and ‚.IL01' to ‚connecting rod left'.

The tree should look like this (note the ‚Select Objects box only shows visible objects):

Select ‚b.r.connecting rod left' – **Important! Always animate the dummy and never the object!**



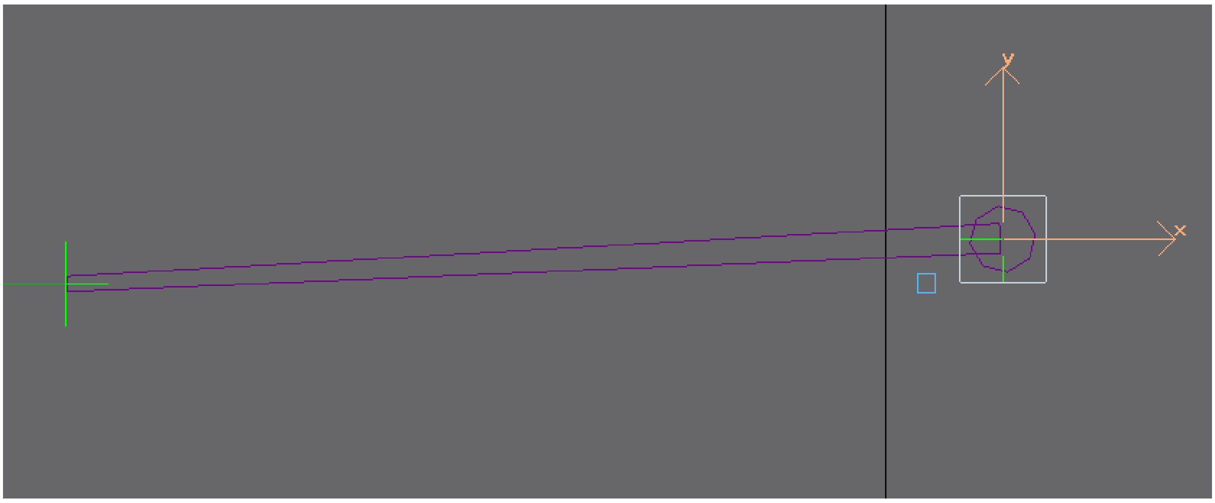
```
.HL
.IL
.JL
b.r.connecting rod left
   b.r.w2
         connecting rod left
            .IL01
Line01
```

Sort: ● Alphabetical ○ By Type ○ By Color ○ By Size

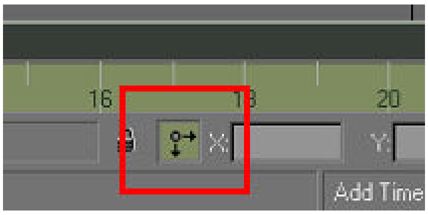
List Types: ☑ Geometry ☑ Shapes ☑ Lights ☑ Cameras ☑ Helpers ☑ Groups ☑ XRefs

☑ Display Subtree □ Case Sensitive □ Select Subtree □ Select Dependents

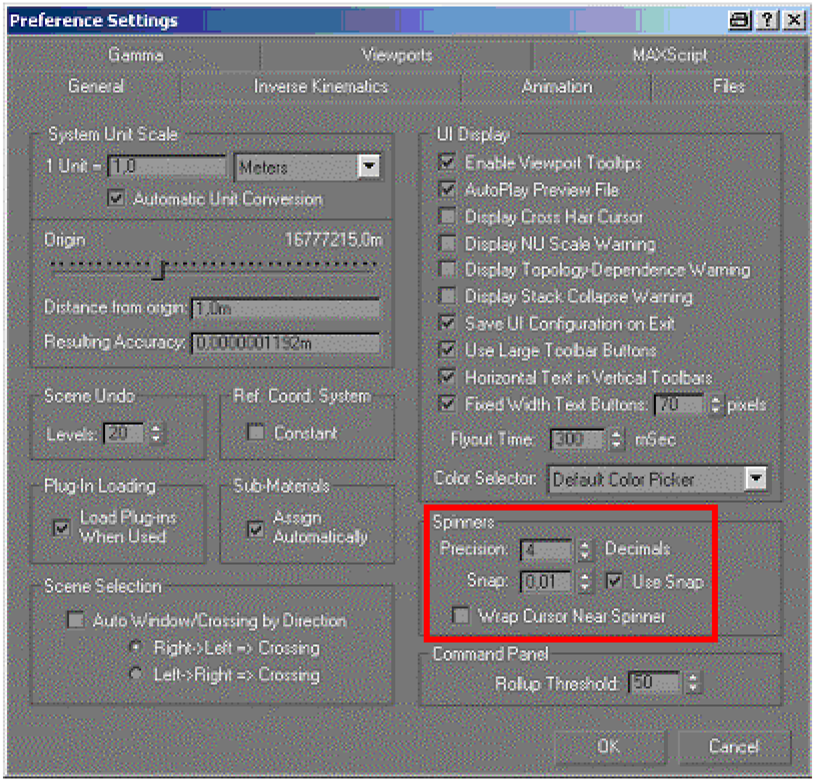
Switch to the right view (press ‚R')



Select the icon shown to enable relative movements of the objects.



Under ‚Customise -> Preferences’ change the ‚Precision’ to 4, ‚Snap’ to 0,01’ and select ‚Use Snap’. This enables finer control over the spinners for moving/rotating objects.



Preference Settings

Gamma | Viewports | MAXScript | General | Inverse Kinematics | Animation | Files

System Unit Scale
1 Unit = 1,0 Meters
☑ Automatic Unit Conversion
Origin 16777215,0m
Distance from origin: 1,0m
Resulting Accuracy: 0,0000001192m

Scene Undo
Levels: 20

Ref. Coord. System
☐ Constant

Plug-In Loading
☑ Load Plug-ins When Used

Sub-Materials
☑ Assign Automatically

Scene Selection
☐ Auto Window/Crossing by Direction
○ Right->Left => Crossing
○ Left->Right => Crossing

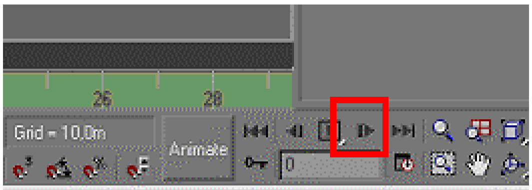
UI Display
☑ Enable Viewport Tooltips
☑ AutoPlay Preview File
☐ Display Cross Hair Cursor
☐ Display NU Scale Warning
☐ Display Topology-Dependence Warning
☐ Display Stack Collapse Warning
☑ Save UI Configuration on Exit
☑ Use Large Toolbar Buttons
☑ Horizontal Text in Vertical Toolbars
☑ Fixed Width Text Buttons: 70 pixels
Flyout Time: 300 mSec
Color Selector: Default Color Picker

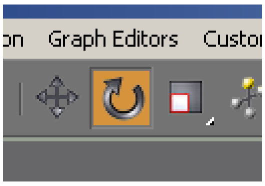
Spinners
Precision: 4 Decimals
Snap: 0,01 ☑ Use Snap
☐ Wrap Cursor Near Spinner

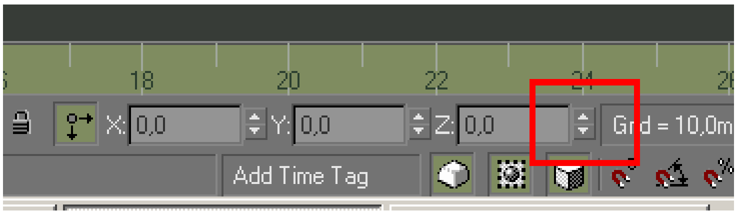
Command Panel
Rollup Threshold: 50

OK Cancel

Click on the icon shown to move to frame 1 of the animation and then select the ‚Animate’ button. Select the ‚Rotate’ tool in the top icon bar and using the spinners next to the Z coordinate field at the bottom of the screen adjust the position of the connecting rod so that it lies on ‚Line01’ again.



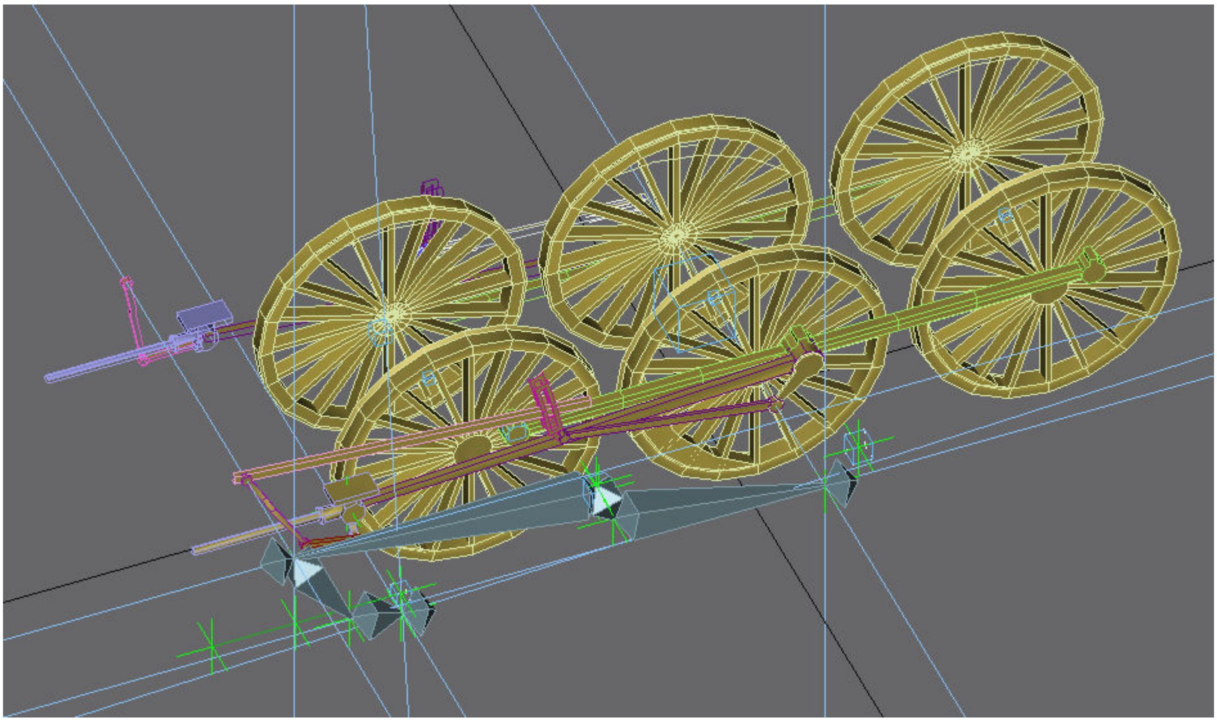




Select the ‚Animate’ button again and advance to the next frame, select ‚Animate’ and so on until you have adjusted the connecting rod position for each of the frames.

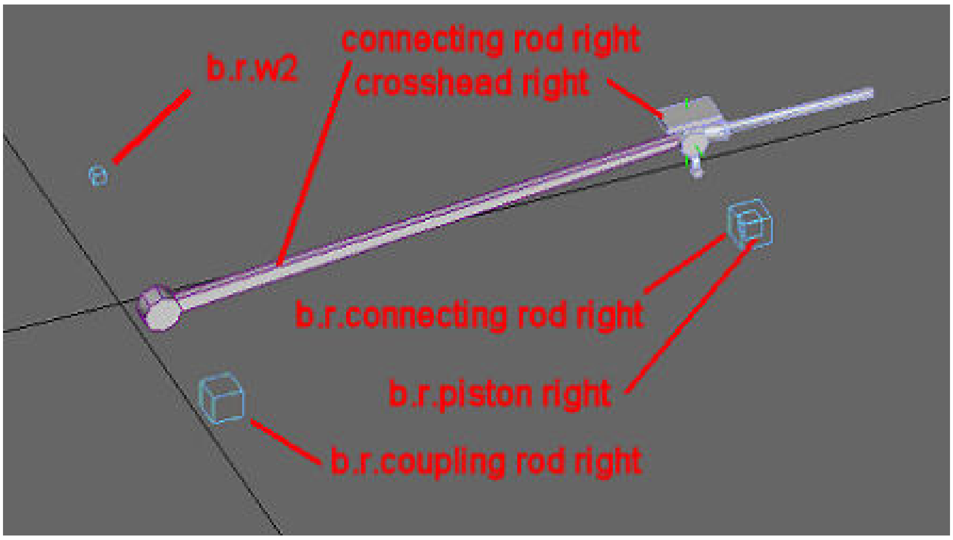
Similarly ‚b.r.piston left has to be rotated for each of the animation frames to get the animation correct.

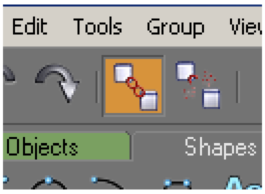
We are now ready to connect the rod objects to the bones and dummies to check the animation works, but before attaching anything make sure that the rods are aligned to world coordinates using ‚Reset Xform’ or using the box trick, i.e. in top view create a box and convert it to editable mesh. Then attach the rod object to the box and finally delete the box element.



Animating the Connecting Rod and Crosshead the Easy Way

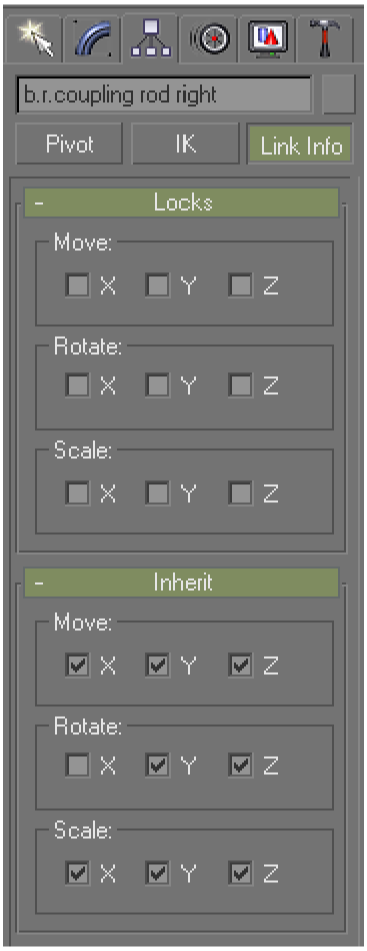
Note that we are now on the right side of the loco as I did the left side the hard way.





The first step is to link 'b.r.coupling rod right' to 'b.r.w2'. Hide everything you are not working on and use the 'Link' tool to create links between the objects by dragging. Create 'b.r.coupling rod right' in top view and centred on the crank pin of the driving wheel.

In the ‘Hierachy’ tab set the ‘Inherit’ check boxes for ‘b.r.coupling rod left’ as shown.



b.r.coupling rod right

- Locks

Move: ☐ X ☐ Y ☐ Z

Rotate: ☐ X ☐ Y ☐ Z

Scale: ☐ X ☐ Y ☐ Z

- Inherit

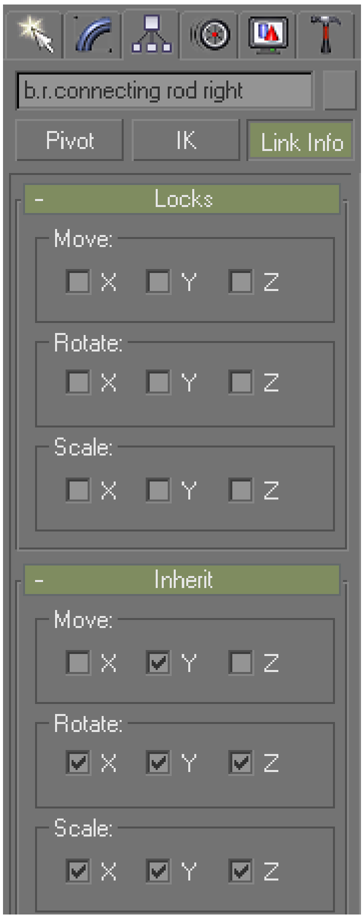
Move: ☑ X ☑ Y ☑ Z

Rotate: ☐ X ☑ Y ☑ Z

Scale: ☑ X ☑ Y ☑ Z

Make sure that ‘b.r.connecting rod right’ has the same X value as that of ‘b.r.coupling rod right .

Link ‘b.r.connecting rod right’ to ‘b.r.coupling rod right’. In the ‘Hierachy’ tab set the ‘Inherit’ and ‘Move’ check boxes for ‘b.r.connecting rod right’ as shown.



b.r.connecting rod right

- Locks

Move: ☐ X ☐ Y ☐ Z

Rotate: ☐ X ☐ Y ☐ Z

Scale: ☐ X ☐ Y ☐ Z

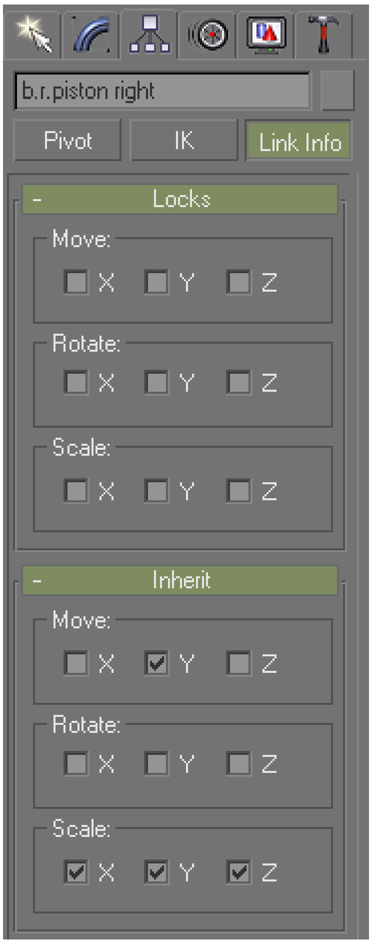
- Inherit

Move: ☐ X ☑ Y ☐ Z

Rotate: ☑ X ☑ Y ☑ Z

Scale: ☑ X ☑ Y ☑ Z

Link 'b.r.piston right' to b.r.connecting rod right'. In the 'Hierachy' tab set the 'Inherit' and 'Move' check boxes for 'b.r.piston right' as shown.



b.r.piston right

- Locks

Move: □ X □ Y □ Z

Rotate: □ X □ Y □ Z

Scale: □ X □ Y □ Z

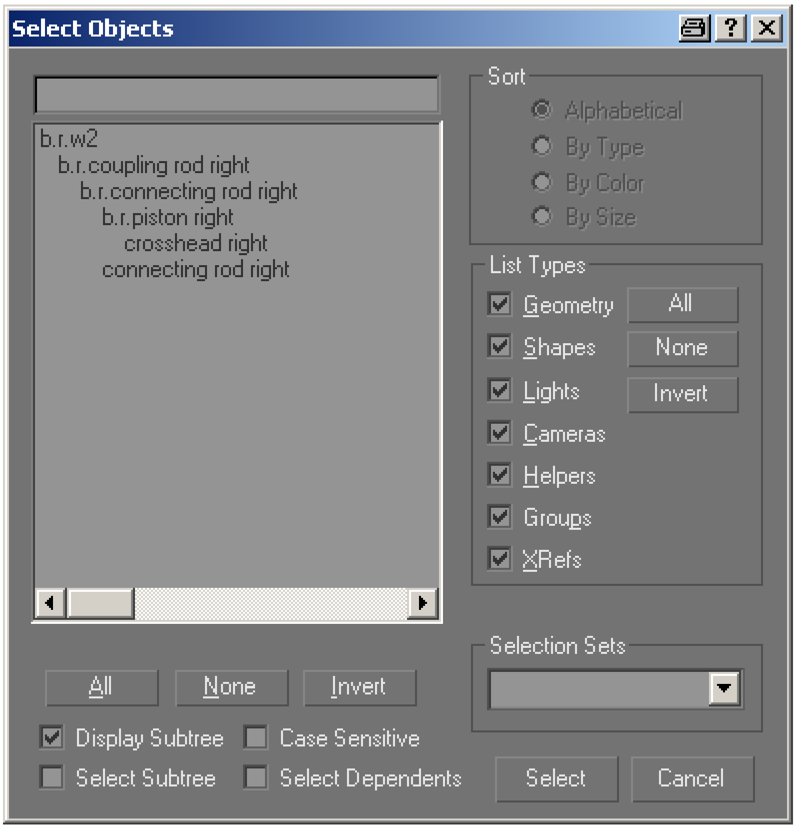
- Inherit

Move: □ X ☑ Y □ Z

Rotate: □ X □ Y □ Z

Scale: ☑ X ☑ Y ☑ Z

Link 'crosshead right' to 'b.r.piston right'and 'connecting rod right' to 'b.r.connecting rod right'. The tree should look like this now: If you play the animation now you'll see that the connecting rod is animated but doesn't point to the crank pin on the driving wheel, this is where the 'LookAt' constraint comes into play.



Select Objects

```
b.r.w2
  b.r.coupling rod right
    b.r.connecting rod right
      b.r.piston right
        crosshead right
      connecting rod right
```

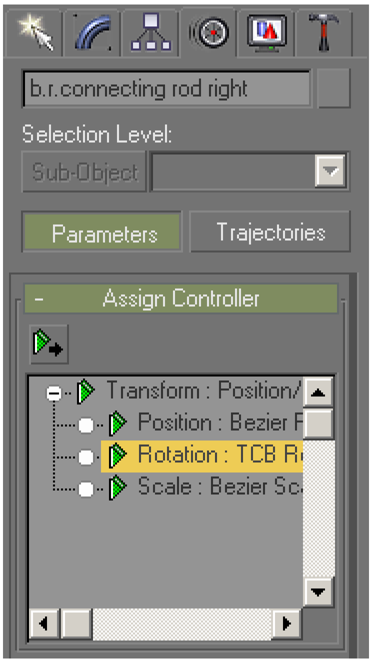
Sort: ● Alphabetical ○ By Type ○ By Color ○ By Size

List Types: ☑ Geometry ☑ Shapes ☑ Lights ☑ Cameras ☑ Helpers ☑ Groups ☑ XRefs

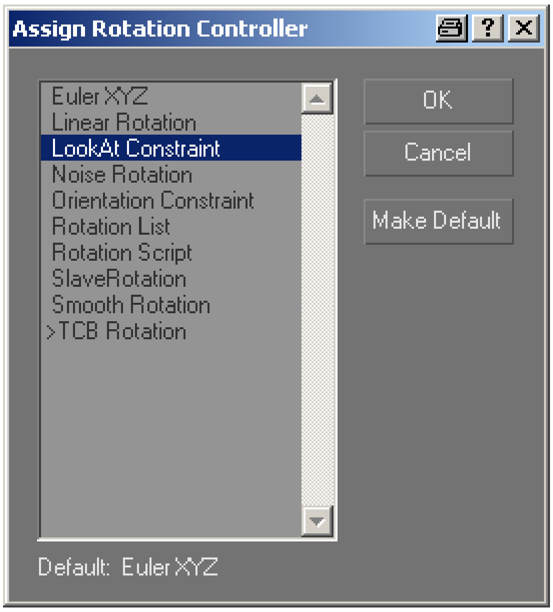
☑ Display Subtree □ Case Sensitive

□ Select Subtree □ Select Dependents

Select ‘b.r.connecting rod right’ and in the ‘Motion’ tab choose Parameters -> Assign Controller -> Rotation’. Click on the ‘Assign Controller’ icon (the green arrow under the text ‘Assign Controller’).



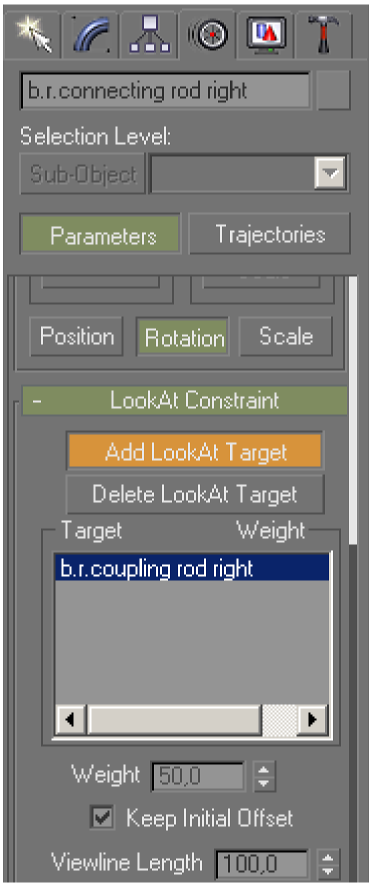
Choose ‘LookAtConstraint’ and then ‘OK’



Choose ‘Add LookAtTarget’ and the select ‘b.r.coupling rod right’.

Important! Make sure ‘Keep Initial Offset’ is selected!

If you now play the animation it should work correctly – and that really is all there is to it…


b.r.connecting rod right

Selection Level: Sub-Object

Target Weight
b.r.coupling rod right

Weight 50,0

☑ Keep Initial Offset

Viewline Length 100,0

Final Linking and Exporting to Trainz

If you have done the preparatory work, you will find the bone making process to be very quick and simple. It should be possible to do one side of the gear in 45 minutes.

For Trainz purposes make sure that every part is linked as a child of b.r.main. No orphans (except for the helper points and ‚Line01’, these will not be exported and so do not need to be linked). Don’t forget to export the ‘anim.kin' file as well as the bogey mesh.

If you try it now everything should work. Link all your lever and rod objects to the adjacent bones, sit back and admire your work. Not too long, you have the other side to do, same tasks, different names.

Those who haven’t fallen asleep by now may have noticed that the animation of the valve spindle itself has not been mentioned. The valve spindle is usually not very noticeable, but if you want to include it add a helper point just below the joint between the radius rod and the combination lever. Create a dummy in this position, lock it so it cannot rotate and link it to ‚b.r.bone' combination lever left’. Finally link the valve spindle object to the dummy.

The Gmax file of the bogey in the state shown above is available here. This has been exported successfully to TRS and shows the complete linking of the left side valve gear. The right side still needs doing so you can practice with that.

Troubleshooting

A few things could go wrong with the export resulting in bits not showing up in Trainz, or displaying wrongly.

If the rods linked to the bone systems don’t show at all in Trainz check that the names of the bones start with ‘b.r.’.

If objects appear inside out in Trainz unlink them, perform a ‘Reset Xform’ and relink them.

If parts of the valve gear appear wrongly positioned double check that dummies were created in top view and that all the bones and objects are aligned to world coordinates (unlink, Reset Xform’ and then relink). Obviously you’ll only need to investigate the objects that don’t show up correctly.

If the animation doesn’t work at all in Trainz you’ve probably forgotten to export the ‘anim.kin’ file or you haven’t called it in the bogey config file.

Any questions, requests for more explanation or corrections needed – please mail me at paul.hobbs@web.de