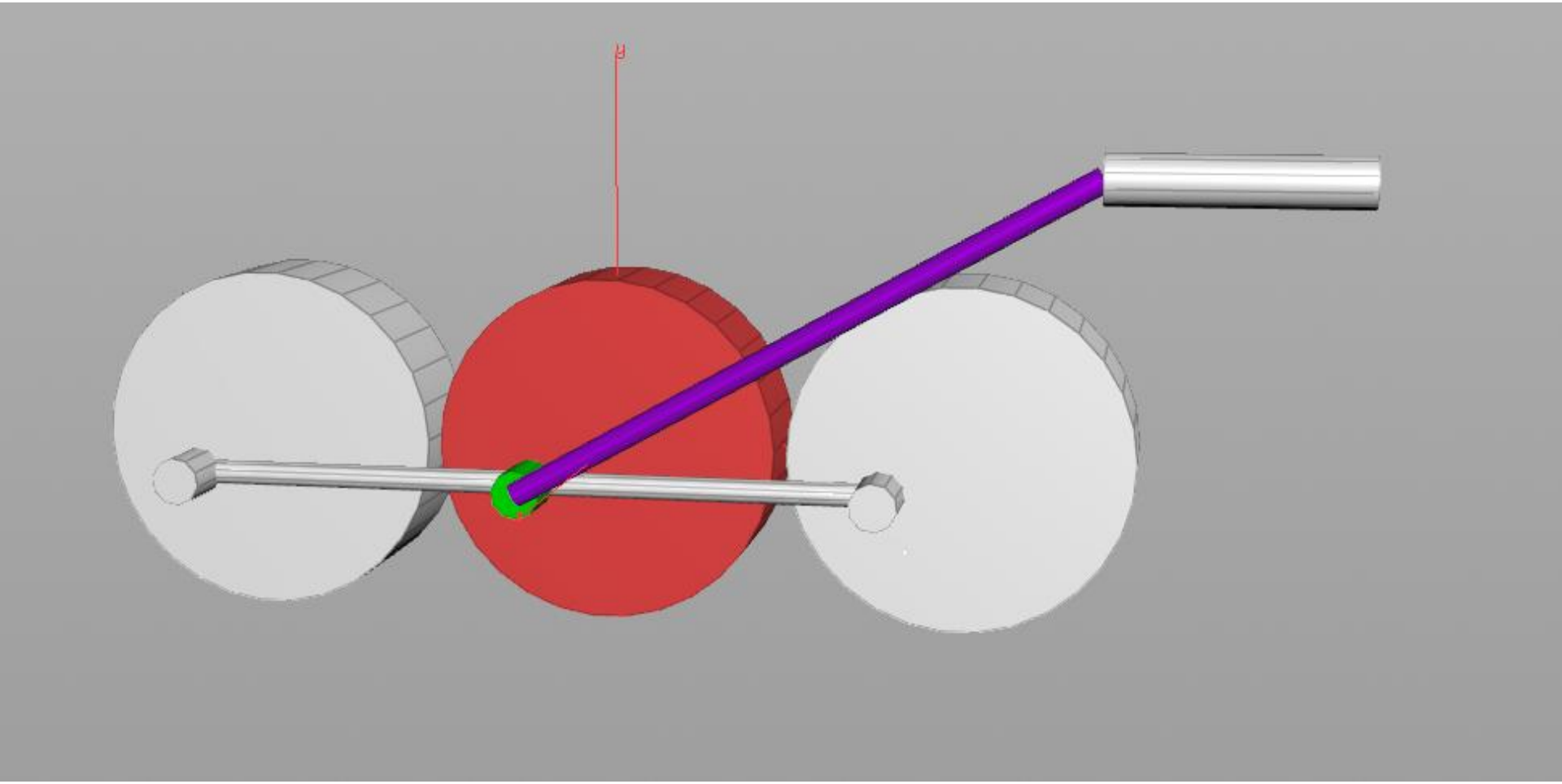
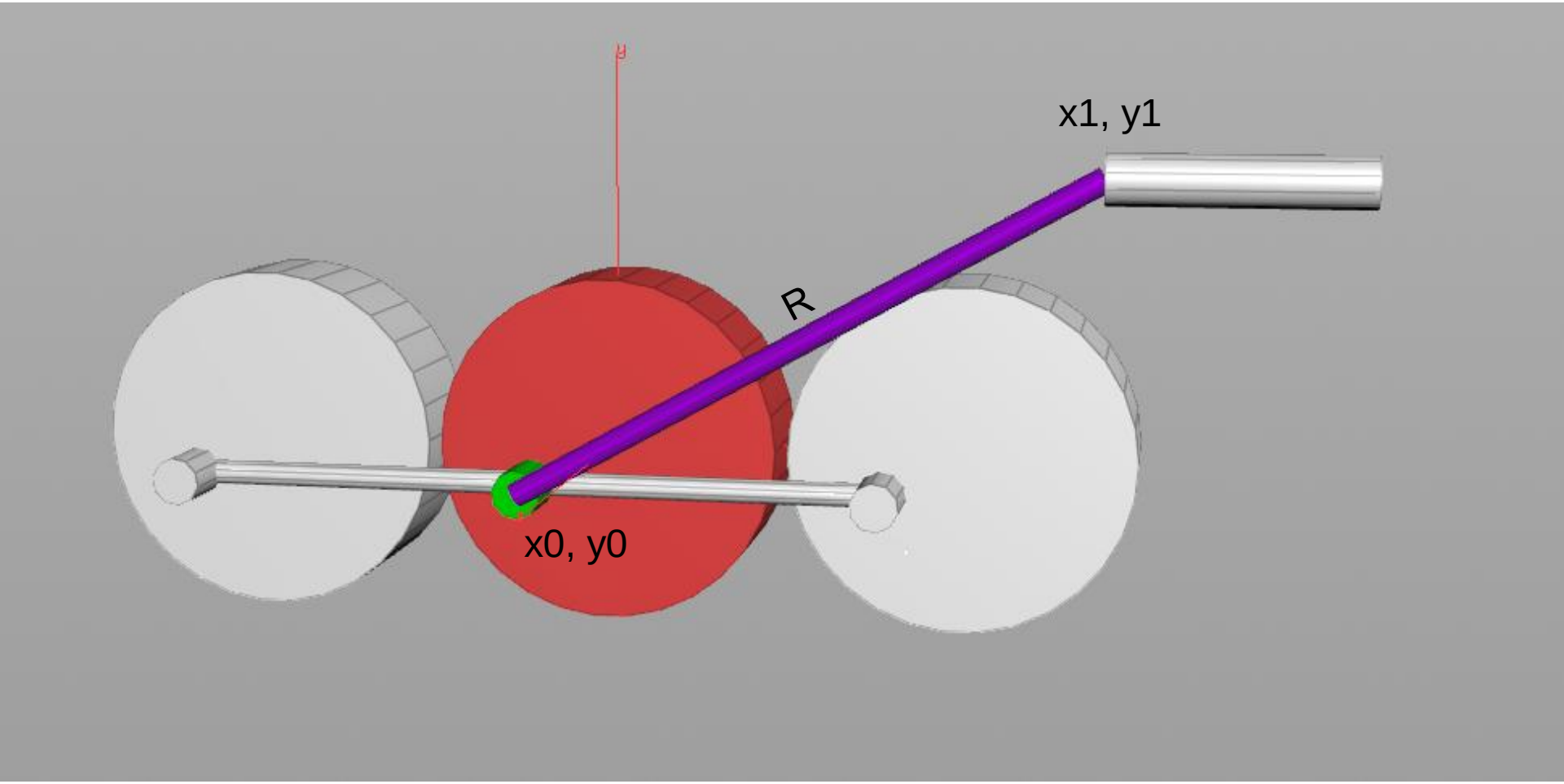
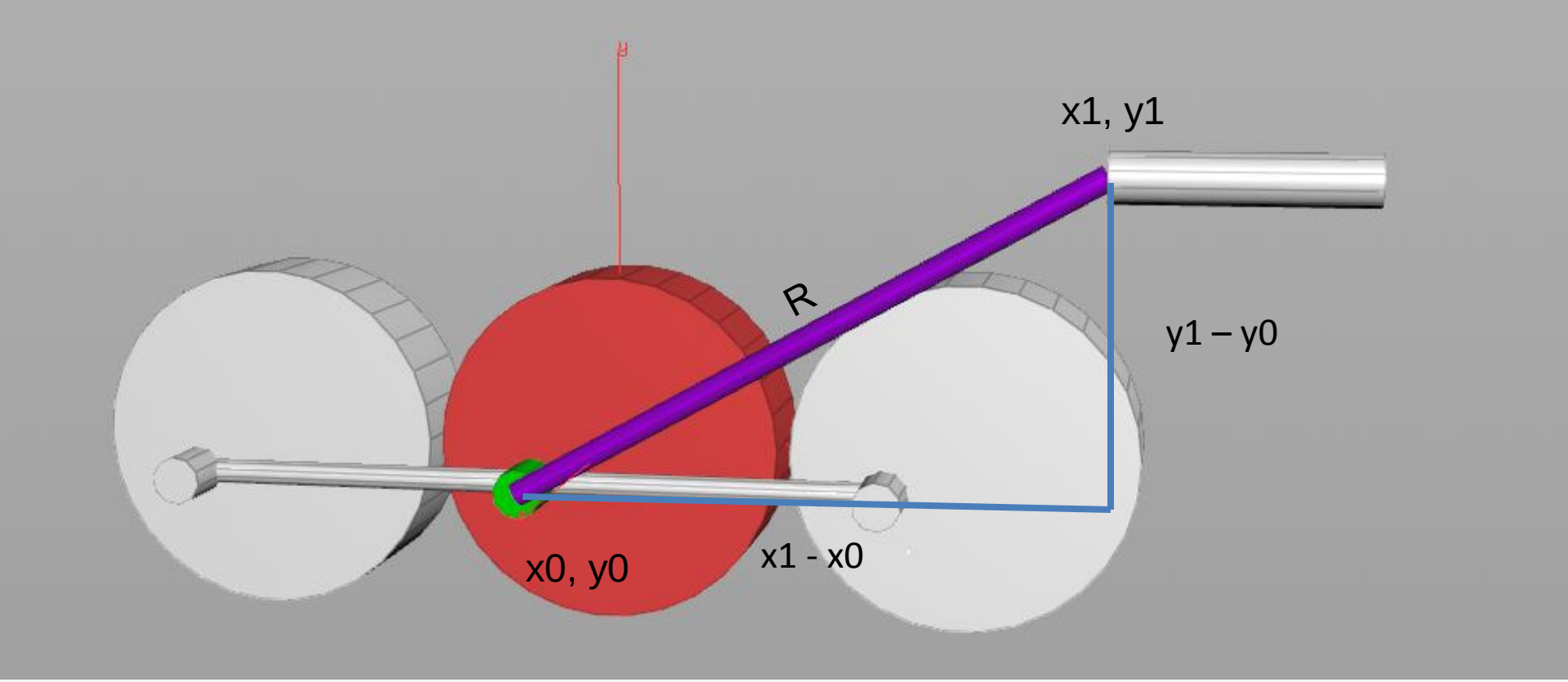
Suppose you want to move a “train wheel”?



Pythagorean to the rescue!

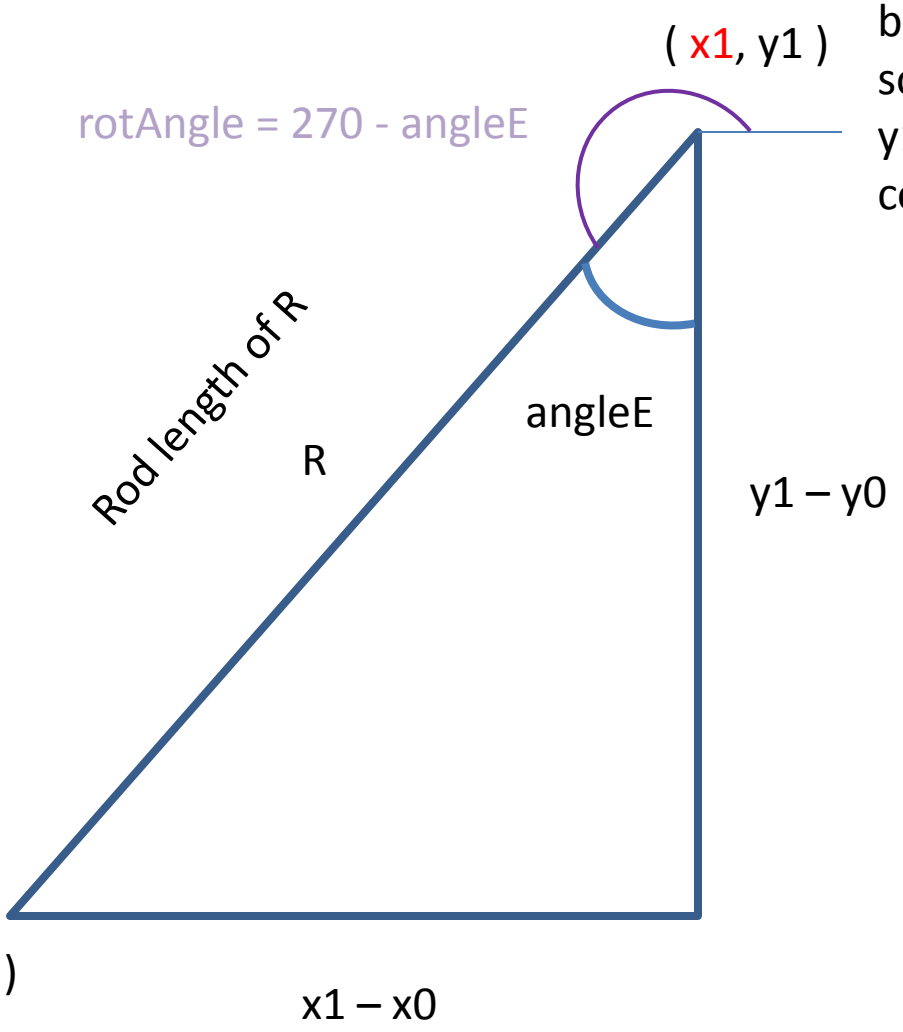


The wheel or the rod have to drive the motion - it's easier to drive the motion based on the wheel rotation (even though a real train is driven the other way).



We know the hypotenuse R. We also know each side. Thus we can compute the angle the Rod R rotates – by the definition of cos it will rotate acos((y1 – y0)/ R)
(recall the positive rotation is from the x-axis moving counter-clockwise so the resulting angle will be subtracted from 270)
We can then compute x1 from pythagorean
x = sqrt(Rsquared - changeInYsquared) where changeInY is just y1-y0

By pythagorean:
x1 = sqrt (R*R – changeY*changeY)
Where changeY = y1-y0



The rod is to be sliding at some height y1 so we can compute x1

We can reference the rotating circle giving x0, y0

From the sample hip file, in TRANSFORM_ROD, compute the angle that the ROD will rotate (seen on the right)

```
{
# Transform the ROD
R = ch("../lengthOfRod");
y1 = ch("../heightOfRod");
y0 = point("../WHEEL_Transform",160,"P",1);
angleE = acos((y1-y0)/R);

return 270 - angleE;
}
```

```
{
R = ch("../lengthOfRod");
y0 = point("../WHEEL_Transform",160,"P",1);
# y1 = point("../xform16",40,"P",1);
# to avoid infinite recursion - use the height of
the rod
y1 = ch("../heightOfRod");
changeY = y1 - y0;
changeX = sqrt(R * R - changeY * changeY);
x1 = point("../WHEEL_Transform",160,"P",0) +
changeX;
return x1;
}
```

In TRANSFORM_ROD, compute x1 of the ROD (seen on the left)