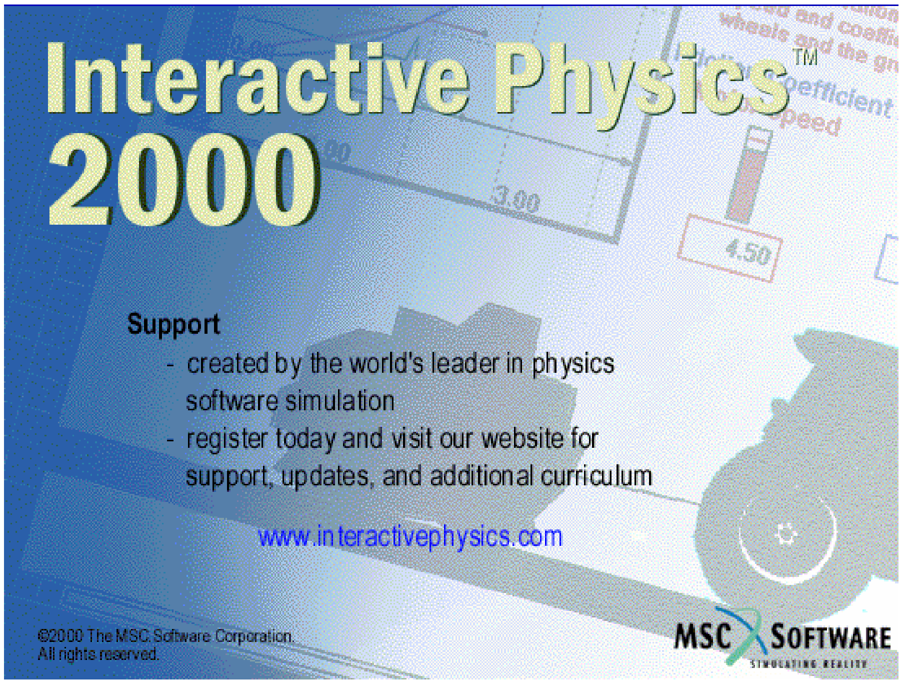
Ontario Association of Physics Teachers

Conference May, 2004

Creating Simulations using some of the Advanced Features of


Interactive Physics™ 2000

Support
- created by the world's leader in physics software simulation
- register today and visit our website for support, updates, and additional curriculum

www.interactivephysics.com

©2000 The MSC.Software Corporation. All rights reserved.

MSC SOFTWARE SIMULATING REALITY

by Roland Meisel and Paul Passafiume 2004/05/06

Introduction:

This tutorial assumes that you have experience in using the basic features of Interactive Physics™ 2000. If not, you may wish to work through the introductory tutorial before attempting this one.

Tutorial:

This tutorial is designed as a self-paced, hands-on exploration of some of the advanced features of Interactive Physics™ 2000. The examples are suggested by the expectations in the Ontario secondary school physics curriculum, but are applicable to any introductory physics course.

Further Information:

You can find further information, links to web sites of interest, and packaged simulations, many of which are freeware, at the following sites:

http://www.interactivephysics.com/

http://ist-socrates.berkeley.edu:7521/projects/IPPS/Contents.html

http://www.scienceman.com/

http://www.enc.org/resources/records/full/0,1240,004489,00.shtm

Interactive Physics 2004:

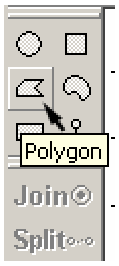
As of this writing, a new version, Interactive Physics 2004, has been released. A demonstration copy may be downloaded from

http://www.interactivephysics.com/

The Cart and the Ball

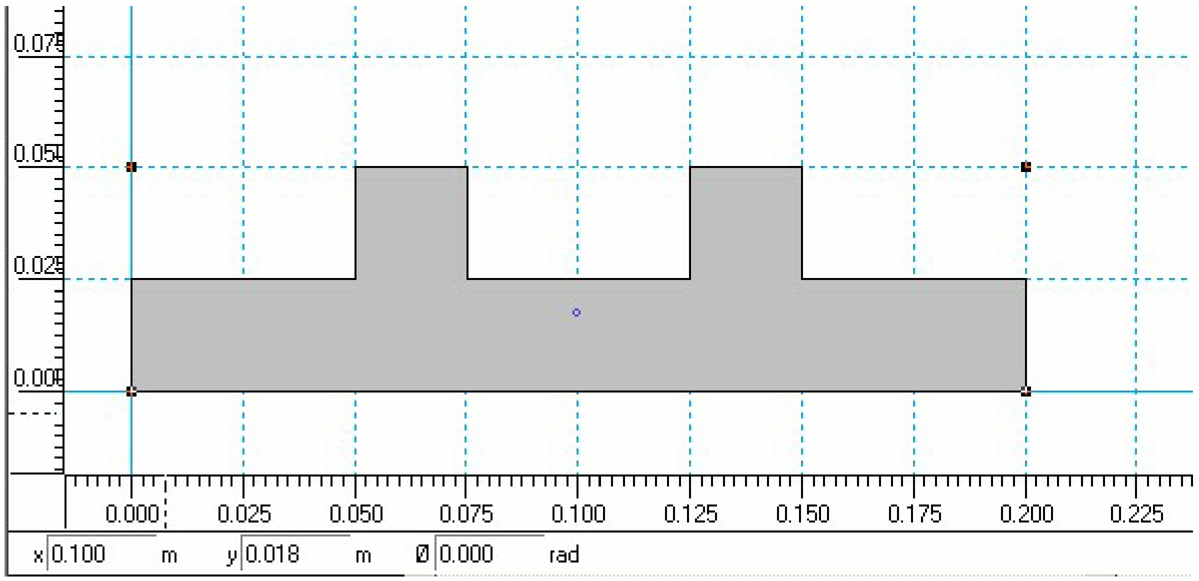
You will begin by programming IP 2000 to simulate the cart and the ball demonstration. A cart moves across a plane at constant speed. At some point, a ball is launched upwards from the cart. Will the ball land in the same place on the cart, or will it land ahead, or behind it?

You will program the simulation such that the ball can be launched directly upwards, relative to the cart, at an angle forwards, or at an angle backwards. This will be controlled using two sliders. The ball will not launch until the cart reaches a position 0.5 m from its starting point. The flight of the ball will be tracked.



First, you will construct a cart 20 cm long, as shown in the screen shot. Use the View/Workspace menu to turn on the grid, rulers, and axes. Adjust the horizontal and vertical scroll tabs until the origin is in the lower left-hand corner of the workspace. Then, use the View/View Size menu to set the workspace width to 30 cm. This will let you use the grid as a convenient way of sketching the cart.

Select the polygon tool. Draw a cart as shown.



x 0.100 m y 0.018 m Ø 0.000 rad

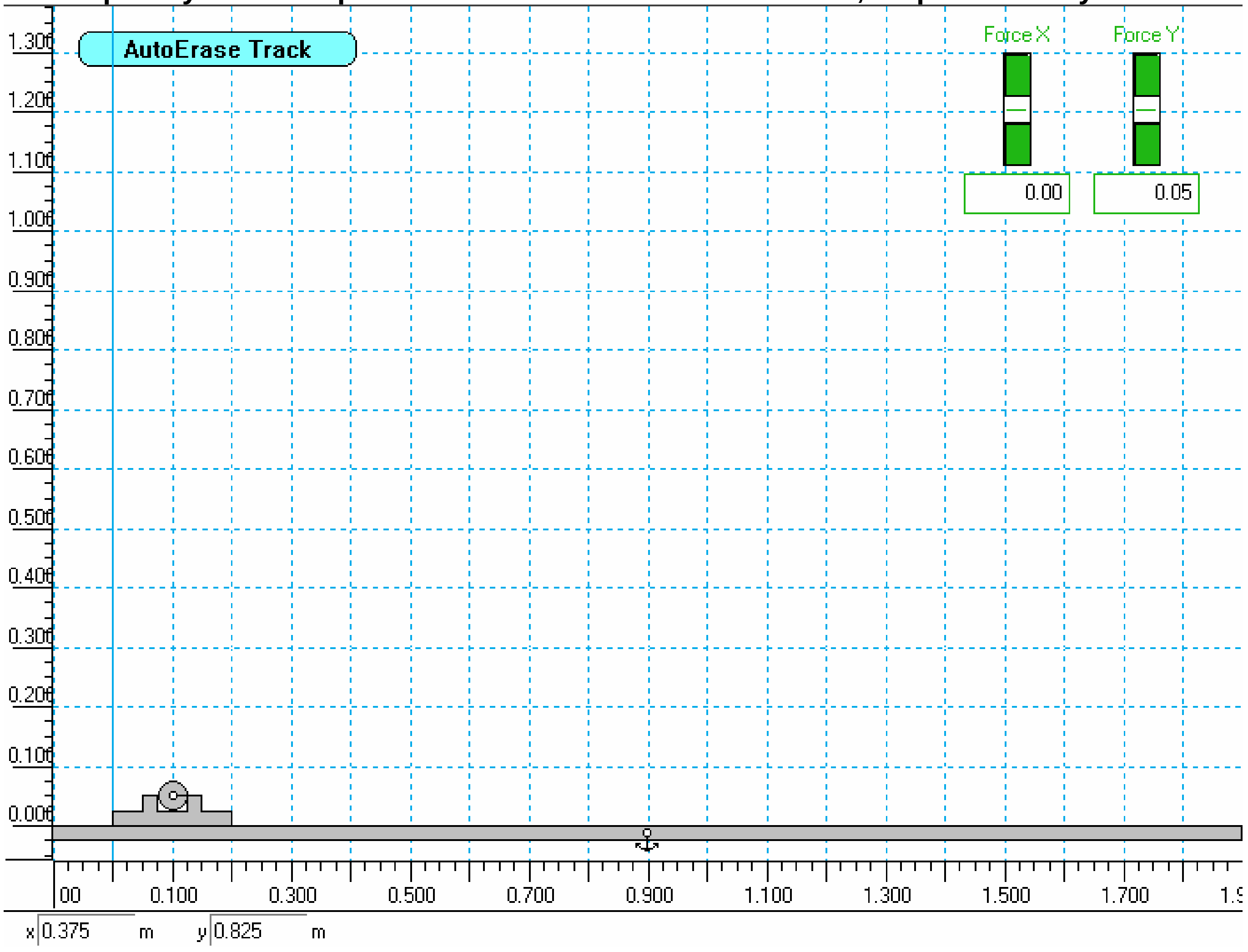
Use the View/View Size menu to set the workspace width to 2.0 m. Use the Rectangle tool to draw a track across the screen, below the cart. Add an anchor to the centre of mass of the track.

Select the cart. Open the properties window. Set the coefficients of friction and elasticity to 0. Set the initial horizontal speed to 0.4 m/s. Switch to the track. Set the coefficients of friction and elasticity to 0.

Add an Auto Erase Track button from the Define menu. Adjust the colour to a pleasing hue. Create a ball using the Circle tool of the correct size to fit into the cart. Open the Appearance box for the ball, and track the outline. Switch to the cart, and turn off tracking for the cart.

Create a generic control from the Define menu. Rename it Force Y. Set the minimum to 0 and the maximum to 0.1. Create another generic control from the Define menu. Rename it Force X. Set the minimum to - 0.5 and the maximum to 0.5. Adjust the colours if you wish, and move the controls to the upper right-hand corner of the workspace. Adjust each control to the middle of its travel.

At this point your workspace will look as shown. If it does not, stop and fix any errors.



Select the Force tool. Apply a force to the centre of mass of the ball. Select the force. You will now set the Properties of the force such that it is turned on as the centre of mass of the cart passes 0.50 m, and turns off as the centre of mass of the cart passes 0.51 m. You will do this by using a nested "if" function.

First, take note of the name of the cart by selecting the cart and looking at the Properties window. It should be something like Body[5]. Then, take note of the names of the sliders. They should be something like Input[9] and Input[10]. Then, select the force. It should be something like Constraint[8]. In the Fy field, enter the formula:

if(Body[5].p.x>.5, if(Body[5].p.x<.51, Input[9], 0),0)

The "if" function consists of three parameters if(x, y, z). The first, x, is a condition. The second indicates the value if x is true. The third indicates the value if x is false.

Note that we have used a nested "if" to handle a finite range of values.

Body[5].p.x returns the x-position of the centre of mass of the ball. If it is greater than 0.50, but less than 0.51, a force is applied, controlled by the position of the slider Force Y.

Study this expression until you are confident that you understand how it works.

A complete list of functions is available in Appendix B of the IP 2000 manual.

In the Fx field enter the formula:

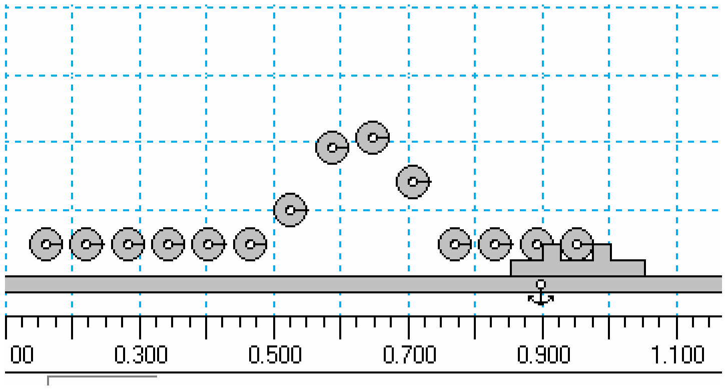
if(Body[5].p.x>.5, if(Body[5].p.x<.51, Input[10], 0),0)

Note: After applying this formula, the arrow on the body will vanish; the force, however, has been programmed.

The horizontal force applied to the ball between position 0.50 m and 0.51 m will be controlled by the position of the slider Force X.

In order to extend the flight of the ball, reset gravity to the lunar value, using the World/Gravity menu.

Run the simulation. With the sliders in the centres of their travel, you should obtain a result as shown.



Adjust Force Y slider to other values, and run the simulation. Then, try adjusting the Force X slider. Note the different results. You can even add some friction between the cart and the track so that it does not remain at a constant speed.

Create another slider. Use it to control a force applied to the cart, so that it accelerates down the track. Experiment to find a reasonable range of values for this force. Run the simulation with different combinations of sliders.

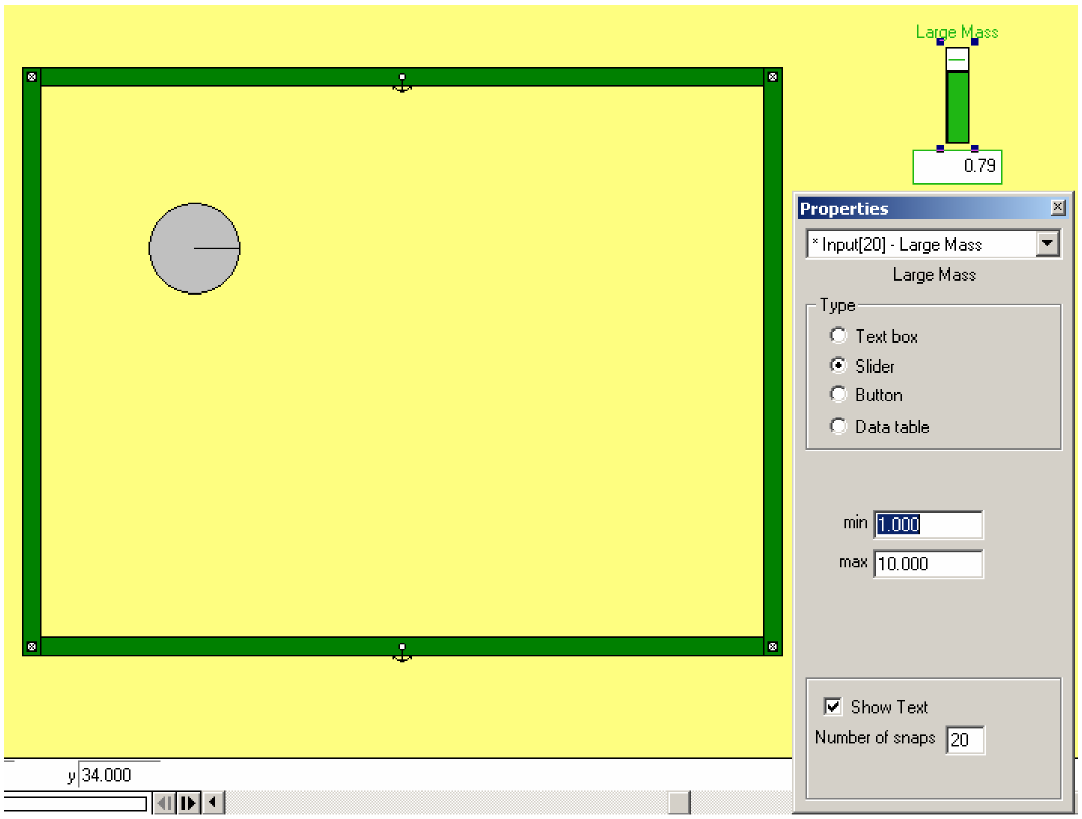
Dress up your simulation with a tasteful selection of colours and backgrounds, as well as some explanatory text.

Brownian Motion

Since the explanation of Brownian motion was the subject of one of Albert Einstein's three famous papers of 1905, you will program a Brownian Motion simulator to illustrate this phenomenon. The masses of both the "large" and "small" particles will be controlled by sliders. All collisions will be elastic. The initial velocity of each "small" particle will be generated at random using a slider to control the maximum.

Open a new workspace. Turn off gravity in the World/Gravity menu. Construct a rectangular box using four thin rectangles, rigid joints to attach them, and two anchors to ensure that the box does not "wander". Set the elasticity of the rectangles to 1. Adjust colours throughout this construction as you see fit.

Create a "large" particle. Adjust the elasticity to 1. Take note of its mass, M. Create a slider to adjust the mass from M to 10M, using round numbers. When you are finished, you should have a workspace similar to that shown.



* Input[20] - Large Mass

Large Mass

Type
- Text box
- [✓ Slider]
- Button
- Data table

min 1.000

max 10.000

[✓ Show Text]

Number of snaps 20

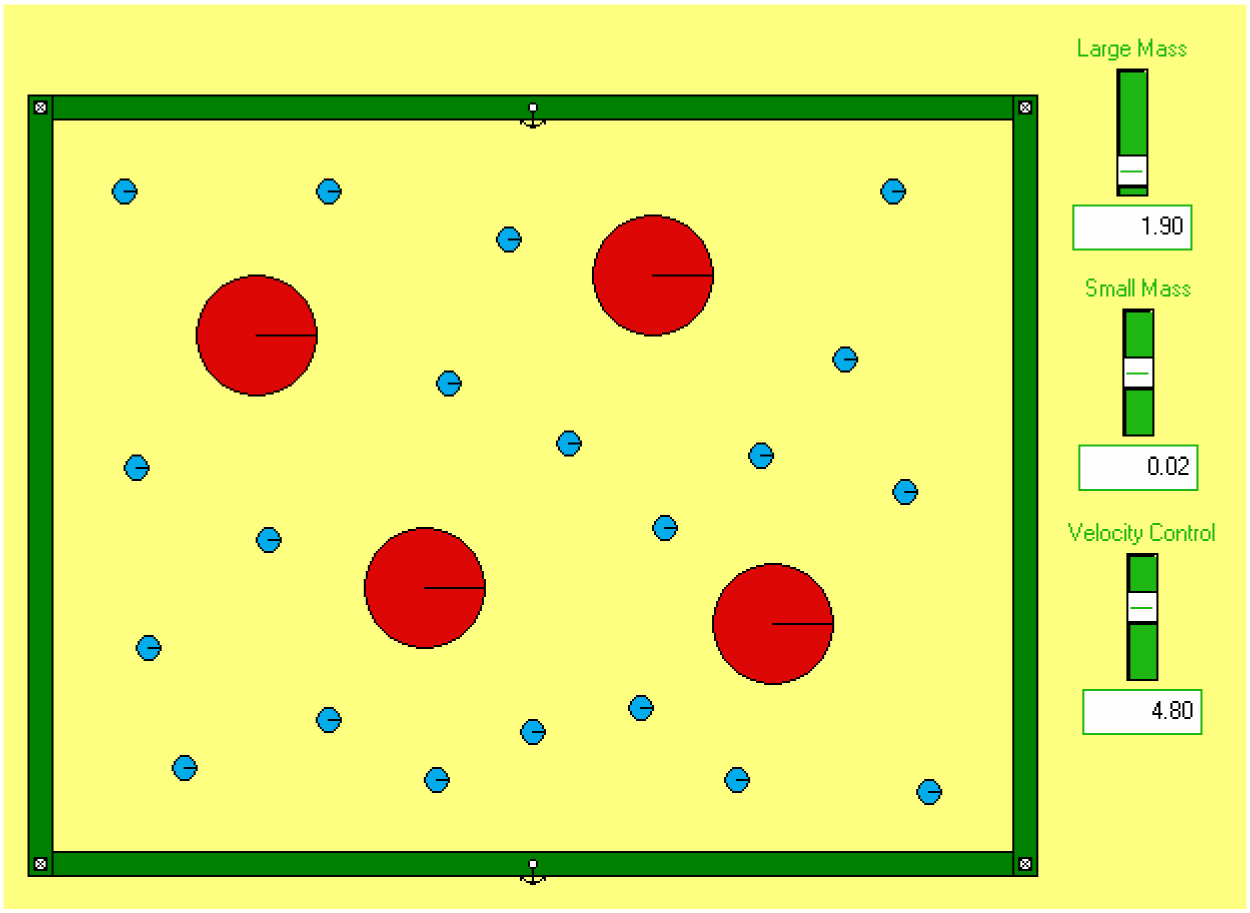
Make three more copies of the "large" particle, and place them in random locations around the box.

Construct a "small" particle. Adjust its elasticity to 1. Note its mass m. Create a slider to control its mass, ranging from m to 10m.

Now you will use the random number, rand(), function available in IP 2000 to adjust the initial velocity of the "small" particle to a random value. Create a generic slider, set its minimum to 3 and maximum to 6. Take note of its control number. It should be Input[26], or something similar. To generate a random number use the formula

-Input[26] + 2*Input[26]*rand()

Suppose that the slider is set at 5. This formula will use the result of the rand() function, which is from 0 to 1, and generate a random number from -5 to +5. Enter this formula into the Vx and Vy boxes of the "small" particle. Note: this will not make Vx = Vy. Each time the rand() function is used, a different random number is generated. Make another 19 copies of the "small" particle, and place them at random around the box. When you are finished, your workspace will look similar to that shown.



Set each slider in the centre of its range. Run the simulation. Follow one of the "small" particles as it undergoes various collisions. Then, follow one of the "large" particles as the "small" particles collide with it. Adjust each of the sliders in turn, and note the effect on the motion.

Predict what might happen if you remove the anchors from the box. Do so, and run the simulation.

IP 2000 has many built-in functions, similar to those found in spreadsheet software. You can find a complete listing of these in Appendix B of the IP 2000 manual.

In all of the simulations so far, you have been advised on the ranges to use for sliders. When you are developing your own simulations, you will have to determine these values for yourself, often by doing trial runs, and making adjustments as required. Because IP 2000 uses finite time intervals to do its calculations, unexpected results sometimes arise. For example, if you set the Velocity Control upper range too high, a particle can "escape" from the box, an interesting analogue of electron tunnelling! Try this.

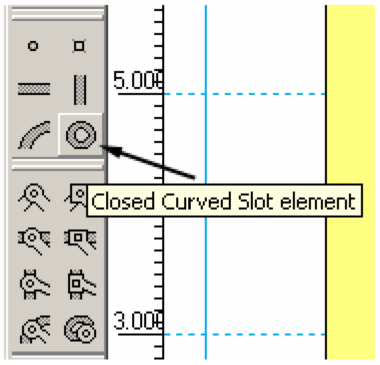
Return the velocity slider to its previous range.

You can refine the Brownian motion simulation in several ways. For example, you have just assigned a random initial velocity to each of the particles. In fact, you should make this a thermal distribution, and might even consider using a slider to control the temperature.

You have not assigned an initial rotation to each of the "small" particles. You can use the existing velocity slider to help you assign a rotation at random, or you can create a new slider. Do this, and experiment with the range values until you have a simulation that seems to work.

A Roller Coaster

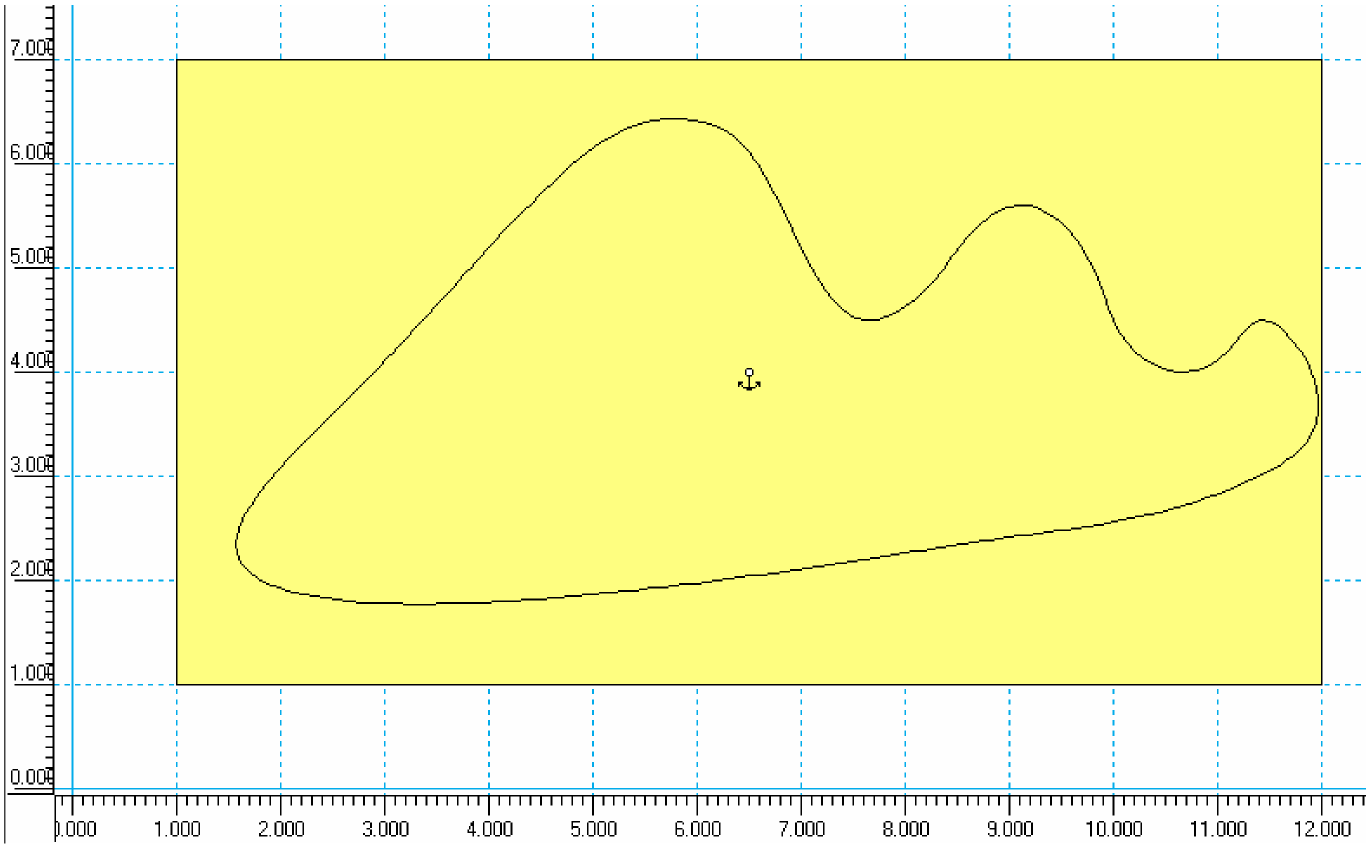
You can use IP 2000 to design amusement park rides. You will design a roller coaster using what you have learned about controls and formulas, as well as introducing slot elements and other features.

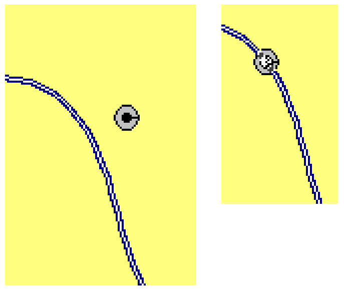


Open a new workspace. Use the View/Workspace menu to turn on the grid, rulers, and axes. Adjust the horizontal and vertical scroll tabs until the origin is in the lower left-hand corner of the workspace. You can use the View/View Size menu to set the workspace width to the real size of a roller coaster. However, your simulation will run faster if you leave the width at the default value.

Draw a rectangle to fill most of the workspace, and anchor it to the background. Select the Closed Curved Slot element.

Draw a closed slot on your rectangle by moving the cursor, and clicking once for each "control point". This is tricky. To get a feel, experiment with the slot and control points until it becomes clear to you how this works. A double-click ends the construction of the slot. Use CTRL-Z to remove the slot, and try it again. When you have the hang of it, construct a roller coaster similar to the one shown. Avoid very sharp "hills" or "valleys".





Draw a small circle just to the right of the first drop. This will be the "car". Add a point element in the centre of the small circle. Use the Shift key to select the point element, and the curved slot. Press the Join button. Your circle will be joined to the slot. To see how it works, Run the simulation. Your "car" may leave the slot if it picks up too much speed. To slow things down (and perhaps prevent your car from flying off the track), set gravity to a smaller value using the World/Gravity menu.

You can reshape your slot graphically. Select Reshape from the Edit menu. Then, select your slot. Your control points will be displayed. Drag a control point to reshape the slot. You can also add more control points by clicking on the slot in the location you want a new control point. You can delete a control point by clicking on it, and then selecting Delete from the Edit menu.

Note that your car will not climb back up the first hill to bring it back to the starting position. You will need to apply an upward force on the car to help it climb the hill. This force will operate only while the car is climbing the hill. You will need to experiment to determine how strong a force is required.

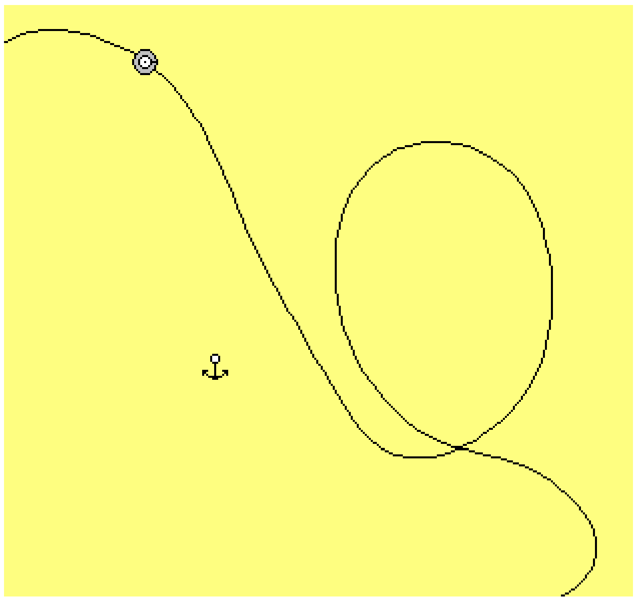
Create a generic control. Rename it Force. For now, set the range from 0 N to 10 N. Use the Force tool to apply an upward force to the "car". You want to use the slider to control the upward force on the car while it is climbing back up the first "hill". In the screen shot above, this runs from about 2.0 m to 5.0 m. However, you only want the force to operate if the car is on the hill, not when it is returning from the right side of the workspace. This happens if the vertical position is greater than 3.0 m. You will need a triple "if" formula:

```
if(Body[6].p.y>3, if(Body[6].p.x>2, if(Body[6].p.x<5, Input[10], 0),0),0)
```

Body[6] is the car (use whatever ID the circle has in your simulation), and Input[10] is the slider (use whatever ID the slider has in your simulation). Once you understand the operation of the formula, insert it as Fy. To better understand the formula, try writing down the coordinates over which the force is to be active.

Test the formula by running the simulation. Adjust the slider until you are satisfied with the operation of the simulation. If you find the slider too coarse, change the range to give yourself more control.

Once you find a workable value for the force, you can make that the permanent value of Fy, you can restrict the range of the slider to whatever range of values you feel work, or you can leave it as is.



Add a meter to measure the kinetic energy of the car. Add another meter to measure the gravitational potential energy of the car. Run the simulation, and note what happens to each of these. Explain any unexpected anomalies. Change the format of the meters to graphs, and run the simulation again.

You can make loops using curved slots. Open a new workspace, and make a roller coaster with at least one loop, as shown.

Force Fields

IP 2000 simulates gravity, electrostatic effects, air resistance, and other physical concepts using force fields. A force field can be defined such that it acts on all bodies in a simulation, using the individual characteristics of each body to determine the effect of the field on that body.

For example, consider a uniform gravitational field. The force exerted by the field on a body with mass m is given by F = mg. IP 2000 will use the equation:

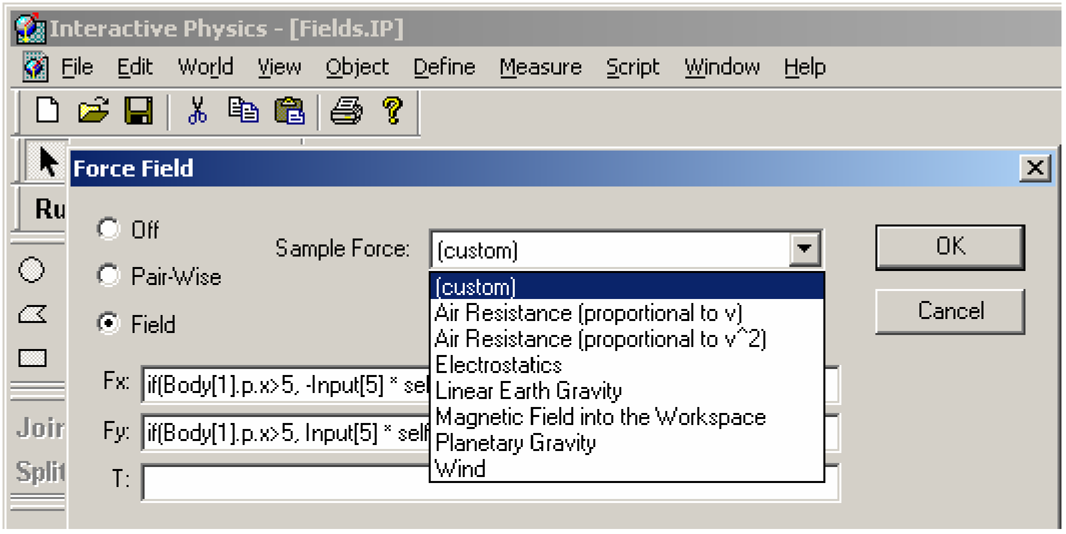
Fy = - self.mass * 9.8

to calculate the force on each body in the simulation. The variable self.mass tells IP 2000 to calculate a different force for each body, depending on its mass.

The force may also be applied pair-wise. This is useful for simulating the gravitational forces in a planetary system, or for simulating the electrostatic forces among a number of charged particles. If there is a large number of bodies in the simulation, many calculations are required, since each pair will have a unique force associated with it.

As an example, you will program IP 2000 to simulate a charged particle which is fired into a region containing a vertical electric field, and a magnetic field perpendicular to the workspace. This arrangement can be used to simulate a mass spectrometer, the measurement of the charge to mass ratio of the electron, and other experiments.

Open a new workspace. From the World menu, ensure that gravity, electrostatics, and air resistance are turned off. Select the Force Field menu, and pull down the Sample Force menu. Note the different force fields available. You can experiment with these, if you wish. When you are finished, return to the (custom) selection.


Interactive Physics - [Fields.IP]

Force Field

Off [✓ Field] Pair-Wise

Sample Force: (custom)

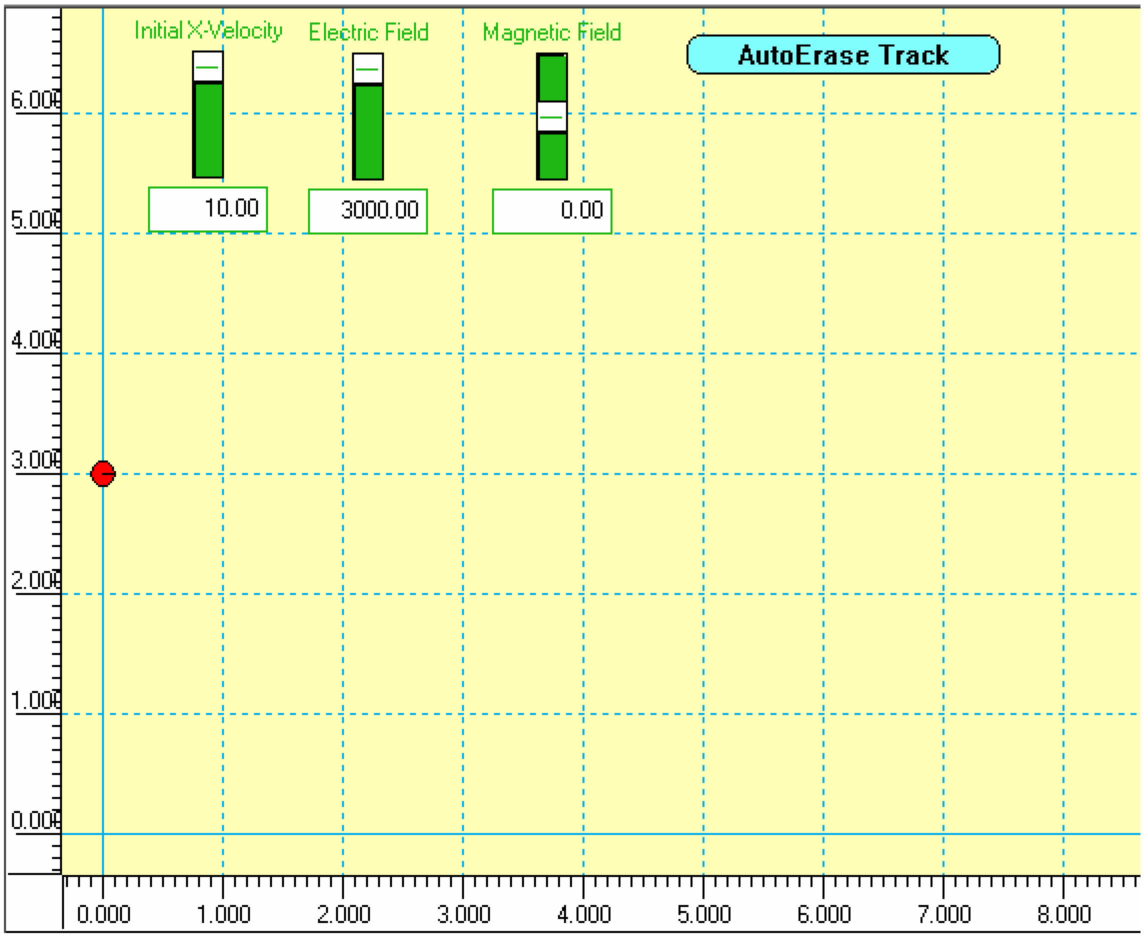
Fx: if(Body[1].p.x>5, -Input[5] * sel

Fy: if(Body[1].p.x>5, Input[5] * self

T:

Close the Force Field menu, and set up your workspace. Turn on the grid, rulers and grid lines. Move the origin to the lower left hand corner of the workspace. Create a small circle around the middle of the vertical axis. Create three sliders. Label the first slider Inital X-Velocity, and set the range from 1 to 10. Associate this slider with Vx for the circle.

Label the second slider Electric Field, and set the range from -3000 to +3000. Label the third slider Magnetic Field, and set the range from -1000 to 1000. Take note of the IDs of the sliders, and of the circle. Turn on Tracking from the World menu. Add an AutoErase Track button. When you are finished, your workspace should look much like that shown.



Return to the Force Field menu under the World menu. Select (custom). Enter these formulas for Fx and Fy, respectively:

if(Body[1].p.x>3, -Input[5] * self.charge * self.v.y,0)

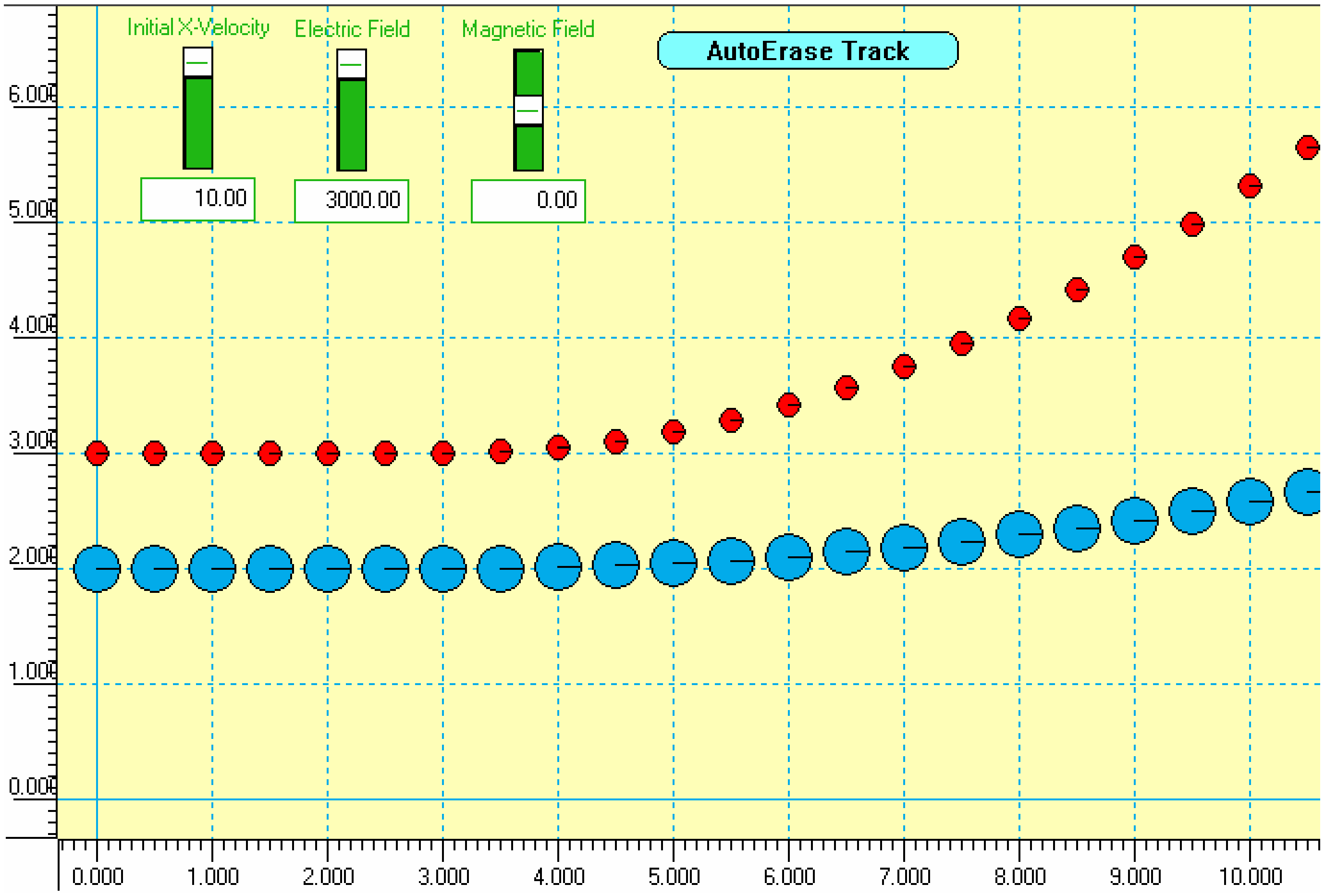
if(Body[1].p.x>3, Input[5] * self.charge * self.v.x + Input[3]*self.charge,0)

Study the formulas. An if statement is being used to turn on the fields when the circle reaches a horizontal position of 3. You can adjust this number to fit your workspace, if necessary.

Input[5] controls the magnetic field, and Input[3] controls the electric field. You can adjust these IDs if necessary. The force applied to the circle by the magnetic field depends on the speed and the charge of the circle. These are taken care of using the variables self.charge, self.v.y, and self.v.x. The force applied to the circle by the electric field depends on the charge of the circle, which is self.charge.

Run the simulation. Experiment with the settings of the sliders. You might try to "balance" the electric and magnetic forces.

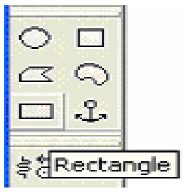
Note that, should you add another particle to this simulation, the field equations will use the characteristics of that particle to calculate the electric and magnetic forces acting on it. Create another particle of different mass. Use the first slider to control its initial speed.


Initial X-Velocity: 10.00
Electric Field: 3000.00
Magnetic Field: 0.00

Note that it follows a different trajectory from the first particle, simulating a mass spectrometer.

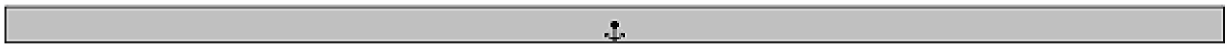
Conservation of Energy: A Cart Rolling Up a Ramp

IP2000 can be used to simulate a cart rolling up and down a ramp; with, or without friction. This powerful, and easy to create simulation can be used to demonstrate the concept of conservation of energy and energy conversion. It can be used in lieu of, or in addition to standard classroom probeware.



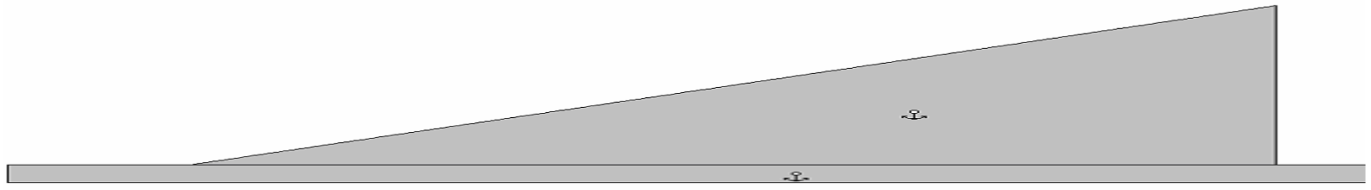
To begin, select a floor onto which you can place the ramp. You can do this by selecting the *rectangle button* indicated to the right.

Next, you'll need to anchor the floor (at its centre of mass) to the background. Select the *anchor button* ( ), and move your cursor until you reach the centre of the rectangle. A crosshair will appear once you've positioned the anchor on the centre of mass, and all you need to do is left click to create the anchor. You should end up with something similar to the image below.

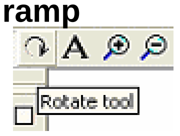


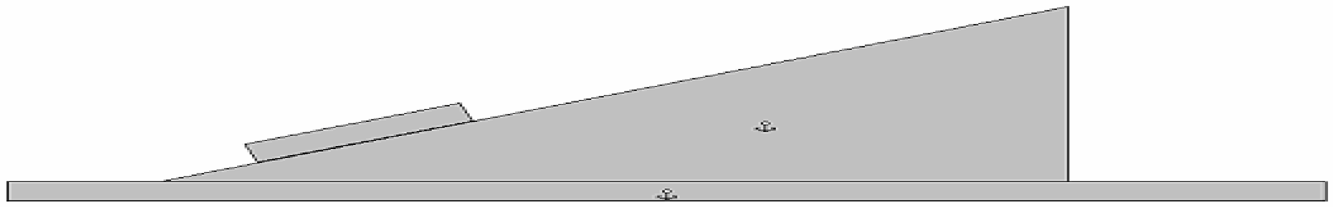
With the floor firmly attached, you now need to create the ramp for the cart to travel on. The *polygon button* ( ) is best for this, and is used in the same way that it would be in a word processing package. Left click, hold and release to draw one side of the polygon. Repeat this two more times to create your ramp (a triangle!).

Once your ramp is in place, you'll need to anchor it so that it doesn't move with the cart. Do this in the same way that you did with the rectangle. Below is the result.

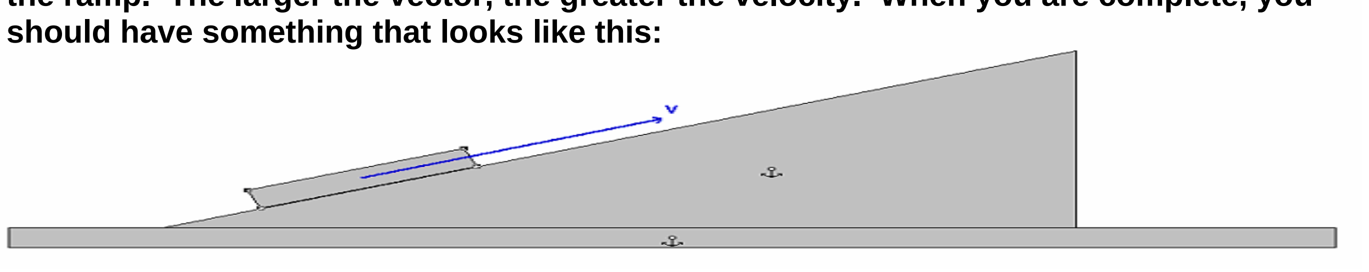


The next thing that needs to be done is to make your cart. You can get as fancy as you'd like, but let's keep it simple for now.   Make your cart in the shape of a rectangle using the *rectangle button*.   Your 'cart' will then need to be aligned with the incline of the ramp using the *rotate tool* indicated off to the right. Left click on the rotate tool and bring it next to the centre of mass of your cart. Left click on the cart to rotate it. Hint: It may be easier to position your cart on the ramp with the *grid snap* option switched off. Go to the *view* drop down menu to turn this option off. When done, you should have something that looks like this:



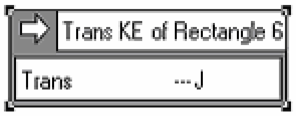


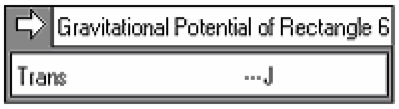
The next thing to be done is to give your cart a velocity. Left click on your cart, and then go to the *define* drop down menu. From this menu, select *vectors* and then *velocity*. No other vectors should be selected. A blue dot should appear at the centre of mass of your cart. This indicates that vector is currently zero. Let's give the cart a velocity by left clicking on the dot (and holding), and extending the vector in a direction that is parallel to the ramp. The larger the vector, the greater the velocity. When you are complete, you should have something that looks like this:



At this stage, you may wish to test that your cart's velocity is not so large that it shoots off the ramp. You may do this by left clicking on the *run button* (Run ▶). Stop, and reset the simulation before proceeding to the next paragraph.

It's now time to create the graph that will display the energy of the block throughout its trip. There are different ways of accomplishing this, but it's often easiest to use existing tools rather than reinventing the wheel. This is what we shall do! Left click on the cart, and then go to the *measure* drop down menu. From this menu, go to *kinetic energy* and select *translational*. Repeat this procedure to obtain the meter which displays *gravitational potential* energy. You should end up with two displays as follows:





Now, we only want a single display that will show: Potential Energy (PE), Kinetic Energy (KE), and Total Energy (TE) simultaneously. Let's begin by renaming the gravitational potential energy meter. Left click on this meter, and then go to the *window* drop down menu. From this menu, select *appearance* and change the title of the metre to something appropriate (like: Potential Energy (PE), Kinetic Energy (KE), and Total Energy (TE) of Cart).

We now need to put the appropriate equations into this meter. Let's first prepare the meter for this information. Double click on your cart's energy meter (the one you just renamed) in order to open its properties box, and rename the label *y1* to PE (for potential energy). Next, give *y2* the label KE. With the cursor still in the *y2* box, right tab over to the equation box that is next to it. This will open that box for the next equation to be entered. Now, left click on the *KE meter* (the property box for this meter is now displayed), and copy the contents of row two, column two (ie.: get the kinetic energy equation of the cart). Now, left click on your cart's energy meter and paste the kinetic energy equation into the cell located in row three, column three.

Next, prepare your cart's energy meter for the Total Energy equation (TE). Move your cursor to the *y3* label, and name it TE. Tab over to the equation cell that is next to *y3* in

order to create an open space. Now copy the *y1* equation (PE), and paste it into the newly created *y3* equation cell. Next, copy the *y2* equation (KE) and *add* it to the equation that you just pasted in the *y3* cell. This creates your total energy equation. Be sure to copy the contents of the cells exactly, and to separated each equation with a '+' sign. Your total energy equation should look something like this:

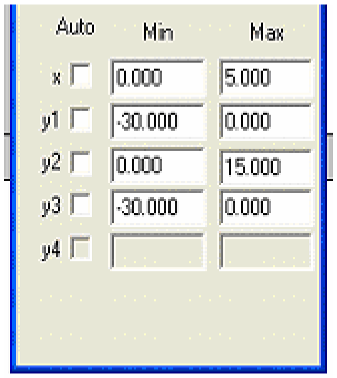
-Body[5].p.y * constraintforce(10002,5).y + 0.5 * Body[5].mass * sqr(Body[5].v)

This equation essentially consists of two terms. The first (-Body[5].p.y * constraintforce(10002,5).y) utilizes the 'constraintforce' function, and allows us to calculate the gravitational potential energy of a body. The first portion of this expression (-Body[5].p.y) defines the vertical position of the body (ie.: its height). The second portion defines the gravitational force acting on the body.

This 'constraintforce' function takes on the following form: constraintforce(a, b).c. The first variable (a) defines the force that we need to use. In this case, the ID number 10002 indicates that we require the force of gravity (other constraint force constants are given in Appendix B). The second variable (b) defines the body to which the force is being applied, and the third variable (c) defines the direction in which the force is to be measured (in our case, the 'y' direction). This expression will then return a value of the force of gravity acting on Body[5].

The second portion of the above expression (0.5 * Body[5].mass * sqr(Body[5].v)) allows us to calculate the kinetic energy of the body. Study this term until you are confident that you understand its essence.

You're now ready to scale the axes of your graph. Begin with the x – axis (time). In the properties box of your cart's energy meter, select the maximum range to be five seconds. For *y1* (PE), select the minimum range to be – 30 and the maximum range to be 0. For *y2*, select the minimum range to be 0 and the maximum range to be 15. For *y3* (TE), select the minimum range to be – 30 and the maximum range to be 0. To ensure that the axes do not scale automatically, *uncheck* the little boxes under 'auto'.



	Auto	Min	Max
x	☐	0.000	5.000
y1	☐	-30.000	0.000
y2	☐	0.000	15.000
y3	☐	-30.000	0.000
y4	☐		

You're nearly there! To simplify the simulation, you may wish to eliminate friction as a factor. To do this, double click on either the ramp or the cart and set the coefficients of friction to be zero.

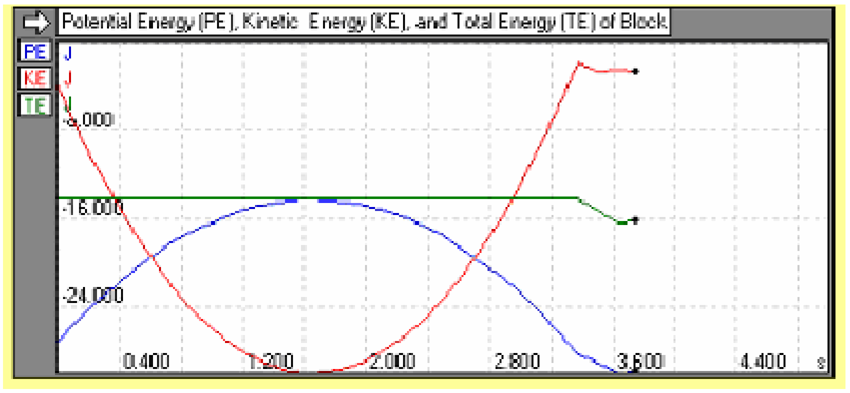
Now we need to set the meter to graphical display. Left click on the arrow in the top left corner of the meter until the graph appears. Adjust the width of the meter so that the title appears in the window by left clicking on a corner, and dragging.

Now, just clear up the workspace by deleting the unneeded KE meter and closing the properties box.

Your simulation can now be run!

Note: A button may be added to clear the graph, but just moving it will do the same thing. Also, it is worth playing with friction here to see how this affects the graphs.

The final result should look something like this:



This completes your tutorial on some of the advanced features of IP 2000. There is still much to be learned, but you should now have the tools to simulate many of the topics and concepts in an introductory physics course.

Try out some of the canned simulations provided with IP 2000 (which have been assessed on the following page), or available on the Internet. Browse through the IP 2000 manual. IP 2000 is a powerful tool, very useful for simulating situations which are difficult to create in the school physics lab, for student investigations, or for student performance tasks.

Analysis of IP Physics Experiments

Here is an overview of several canned IP scripts available as part of the IP2000 software package. These scripts can be found in the IP2000 folder that should be located on the C: drive following a standard install. The following path should assist you in locating these files: C:\Program Files\IP2000\PhysicsExperiments.

The *folder* column below indicates the folder in which the simulation may be found. The *program* column indicates the program to which the script may apply (10 = grade 10; 3U = grade 11 physics; 4U = grade 12 physics).

Note: 4 Star Rating given below is based on applicability to the high school curriculum.

Folder	Program	Overview of Folder Contents	4 Star Rating
Advanced	Other	Simulations of gears, machines, and statics	✫✫
Cams	Other	1 simulation	✫
Collisions	4U	13 simulations	✫✫✫✫
Electrostatics	4U	4 simulations	✫✫✫
Equilibrium	4U	4 simulations	✫✫
Evaporation/Condensation	Other	3 simulations	✫
External Data	4U	Shows how exp. data can be imported to IP for simulation purposes (1 simulation)	✫
Friction	4U	2 simulations	✫
Fun Stuff	Other	4 simulations	✫
Kinematics	10/3U/4U	2 simulations	✫✫✫
Kinetic Theory of Gas	Other	3 simulations	✫
Magnetics	4U	3 simulations	✫✫✫
Math	Other	1 simulation	0
Miscellaneous	3U/4U	15 simulations	✫✫
Momentum	4U	4 simulations	✫✫
Motion in One Dimension	10/3U	2 simulations	✫✫
Motion in Two Dimensions	4U	1 Simulation	✫
Newton's Laws	4U	2 simulations	✫✫
Oscillations	4U	9 simulations	✫✫✫
Particle Dynamics	3U/4U	3 simulations	✫✫
Projectiles and Rockets	4U	4 simulations	✫✫
Pulley Systems	4U	2 simulations	✫
Rotational Dynamics	4U/Other	9 simulations	✫✫✫
Sound	3U	1 simulation	✫
Waves	3U	1 simulation	✫✫✫
Work and Energy	3U/4U	3 simulations	✫✫✫✫