Real Physics Help (V1.4.4, June 2014)

 (realphysicssupport@kirkkaminsky.com)

User Interface Help (By Function)

Move a Body

Method 1: To move a body simply drag it. To get its position before moving it, single tap the body. (If you wish to prevent accidental scrolling if you don't precisely touch the body, set 'Lock World Bounds' in World Settings accessed by double tapping an open spot on the world background.)

Method 2: Double tap on the body to manually set the body's position to high precision. (Also see ‘Snap Objects to Grid’ under World Settings.)

Locate a ‘Lost’ Body

Double tap with two fingers on a open part of the world view, and the world view will re-center itself on the nearest body. Repeat as necessary.

Set a body's velocity

Method 1: To set a body's velocity directly, double tap a body but do not lift your finger at the end of the second tap. The length and direction over which you subsequently drag your finger will assign the speed and direction to the body.

Method 2: (V1.4.3) Long press on a body. After one second the velocity and position will be displayed, and moving your finger will graphically set the velocity.

Method 3: Double tap on the body to manually set the velocity under the 'Body State' section.

Note: Only singly constrained or unconstrained bodies can have non-zero initial velocities assigned in this way. (Otherwise, constraints could be violated.) Bodies whose single constraint is a normal force constraint can be assigned initial velocities tangent to the curve/surface, while bodies whose single constraint is a tensional force can be assigned an initial velocity perpendicular to the tensional direction.

Create a body

Tap the 'plus' sign icon on the toolbar, and select 'Create Body'. A new body with a random color and position within the current world view boundaries will be created. Depending on the setting in 'World Settings', the body will initially be subject to either no forces or a single constant downward directed gravitational force.

Destroy a body

Double tap the body you wish to destroy. Then tap the garbage icon at the top of the body view. At least one body must always exist.

Modify a body

Double tap the body you wish to modify. From the body detail view you can change:

- body kinematical state (position and velocity)
- body mass and appearance (color)
- body tracking features (e.g. FBDs, trajectories)
- explicit (nonconstraint) forces (e.g. gravity, custom forces)
- constraint forces (normal forces and tensions)

View, edit or graph a force (V1.4)

After you have added forces to a body, you can view a complete list of them, their current values, and the parameters or functions that define them by tapping on the 'Force Viewer/Editor' button in the body detail view.

A table of all the force titles/types, their last computed magnitudes, Cartesian components, work done (if enabled) since last evolution, and the parameters that define them (e.g. spring constant for a linear restoring force, drag coefficient for the drag forces, the functions that define a custom force, and so on) are presented. From there any force can be deleted by tapping on the trash icon, graphed by tapping on the graph icon, or edited (where applicable) by tapping on the appropriate parameter field. This table also allows you to identify a particular force in a complex free-body diagram from its magnitude, understand how any example was set up in detail, and easily edit the parameters defining any explicit force without having to remove and then re-add it.

Note: that gravitational forces, normal constraint and tensional constraint forces have no editable parameters. (Change the mass of the body, or the gravitational acceleration to affect the gravitational force.) Also, by default, the computation of 'work done' is dis-

abled for any new simulation for performance reasons on old devices. To enable computations of work (and produce work graphs) enable the setting 'Compute Work' in World Settings first (accessed by double-tapping on an open spot in the world view).

Set a halting condition

If you want to halt the simulator when a physical property of a body reaches a certain value or after a time interval has elapsed, tap the 'stop-equal' icon on the toolbar in portrait mode. Then either tap the clock to set a halting time, or a body to set a specific halting condition on the property of a body (like its position).

The ability to specify a halting condition can be used to solve a wide variety of physics problems, especially when several physical properties are graphed at once. For example: by halting the simulation of a projectile when a body reaches a given vertical position, while simultaneously graphing speed vs time, can be used to determine how fast a projectile lands. It works best (i.e. most reliably) for physical properties that can change sign. If the simulator fails to halt when you expect (because it 'overshoots' or misses the halting value), try setting a smaller time-step and/or slowing down the simulation rate. (V1.4) If you wish to increase the numerical accuracy of a solution via 'halting', then decrease the simulation time-step *dt* directly and/or decrease the simulation rate below real-time evolution (which also decreases *dt*).

Halting tolerance is specified within World Settings (accessed by double-tapping on an open spot in the world view.) The simulation will halt when the halting property dynamically falls within the tolerance of the specified halting value.

Add a graph

Method 1 (All devices): Double tap the body of interest. Scroll down to the Body Tracking section, and click the plus sign beside either

a) the 'Add Versus Time Graph' table entry **OR**
b) the 'Add General Graph' table entry. (V1.3)

The former produces a graph on evolution of a single body property versus time, while the latter produces a plot of two body properties you wish to plot against each other.

(V1.4) To graph a specific force's components, magnitude, or the work done by the force vs time or vs another body property, tap any 'graph icon' in the Force Viewer/ Editor.

To see the graph on the iPhone/iPod Touch, exit the body view. The graph will have been added to the list of graphs to be generated below the world view at evolution (in portrait mode), and will have the same color as the body. Swipe through the list at the bottom of the screen to find it. On the iPhone/iPod Touch there is space for a single graph at a time, whereas the iPad can display four graphs at a time before having to swipe. Also, by pinching a graph on the iPad you can zoom in on (or away from) a specific graph.

Method 2 (iPad Only): Double tap an empty portion of the 'graphing' part of the screen in portrait mode. Select 'Add Versus Time Graph', select the property you wish to graph, tap done and then tap the body to which the graph will be associated. (To add a general two-variable graph, follow method 1.)

Show coords of a point on the graph

Long press on the graph until the coordinates of the point are displayed. By dragging your finger the crosshairs will track over the graph's underlying data points. All ordered pairs are in the form (t,propValue), where t is in seconds, and propValue is in the SI/metric units of the property.

Notes:

- Graphs are updated every tenth of a second by default (i.e. at real-time evolution). To affect the time-resolution of the graph, In V1.4, change the simulation rate away from real-time. Simulation rate can be changed in World Settings.
- As of V1.3, you can plot any two body properties against each other, in which case dragging yields (propValue_x,propValue_y) and the time at which that pair occurred; and in V1.4, you can graph any force (magnitude, Cartesian component, or work done) versus any body property.

Export data from a graph (V1.4)

Tap any graph with two fingers to export the data contained in the graph to e-mail, the clipboard, or any app that can open comma-delimited text files (e.g. a spreadsheet app). To adjust the time resolution at which data is captured by the graph on evolution (0.1 s at the default real-time simulation rate), change the simulation rate in world settings. (One point is always added to the graph every 0.1 s in 'real' time regardless of the simulation rate.)

Detect collisions

In 'World Settings' (accessed by double tapping on an open spot in the world view) toggle the 'Detect collisions' switch to on. Bodies that are subject to collision detection are represented as round (so that the line of impact can be determined).

Ensure you set the coefficient of restitution (the 'springiness' of interbody, or body-surface collisions), which by default is set to exactly one (which enforces perfectly elastic collisions). After a collision occurs, the scattering angles (for interbody collisions) or the impact-rebound angles (for body-surface collisions) will be displayed by default if the 'Show Collision Data' is enabled in World Settings.

(V1.2.1) Only bodies that are unconstrained or singly-constrained by normal forces or tension constrained by a double connection to a fixed point are allowed to participate in collisions. The constraint takes priority over momentum conservation in the direction of the constraint normal or tensional force, as the normal or tensional force is 'external' and hence momentum violating in that direction. Momentum is conserved perpendicular to the constraint force during the collision.

Also at the possible expense of performance, collisions can be analyzed at the per iteration instead of at the per frame time step (default).

Set a constraint curve

Double tap an open spot on the main world view, i.e. away from a body, spring, or pulley. Select 'Set Constraint Curve/Surface'. The function parser in which you can specify a function of the form y=y(x) will appear. Most of the 'functions' (trig, exponential, etc.) are available by tapping '2nd' button -- as you would on a scientific calculator. You also have cancel, clear, and delete buttons available. To 'validate' your function, press 'd/dx' -- which will compute a symbolic derivative for you. Note that with this feature, you also have effectively a simple graphing calculator!

Note: You still will have to specify which bodies are constrained to the curve/surface. See the help section on 'Constrain a body to a curve' as well as general Real Physics help for more details.

Alternatively, if all you need is a single or double *straight* inclined plane/surface, double tap an open spot on the main world view and select 'Inclined Plane Builder' to more easily define inclined plane directly in terms of angle, and start and stopping points.

Constrain a body to a curve

First ensure that you have a constraint curve/surface defined for the world by double tapping on an open portion of the world. (See help on 'Set a constraint curve'.) Then double tap the body, scroll down to the 'Body Constraint Forces' section of the body view, and turn on the 'Normal Force Constraint' switch. At that point, you can also subject the body to kinetic or static friction along the curve/surface (with specified coefficients of kinetic and static friction), by turning on the friction switch below the normal force constraint switch. If the body is already constrained by a tension, then only kinetic friction is available (i.e. static friction halts aren't applied to a system of bodies connected by rigid tensional constraints).

Constrain a body to a trajectory (V1.4)

Double tap any body and in the first 'Body State' section, select 'Constrain Kinematically'. From there you can directly define the trajectory of the body parametrically with respect to time by explicitly defining x(t), y(t) for a body. This in particular fixes the components of the velocity, the components of the acceleration and hence the components of the net force that acts on the body. Hence, this immediately removes all 'explicit' or other 'constraint' forces from the body, and makes the body extremely massive. Other bodies can still interact with the body subsequent to kinematical constraint via an interbody spring, interbody custom force, or a single direct tension. Bodies that are joined via a direct tension constraint, will have their initial velocities set to that of the kinematically constrained body. For more complicated setups, instead of kinematically constraining the body, make it extremely massive, and apply a custom explicit force to it that is equivalent to kinematically constraining it. (See the black body in the default V1.3 world for an example of this.);

Add a fixed point constraint (pulley)

This type of rigid constraint enforces that the sum of the distances to a particular point (represented visually by a 'pulley') from two bodies remains constant: envision a rigid cable passing over an ideal pulley connected to two bodies. The total length of the cable is constant. To add this kind of rigid constraint, either:

Method 1: Double tap an open part of the world view and select 'create light pulley' at the location you wish to create the pulley. You can then fine tune the position of the center of the small (i.e. point-like) pulley with the sliders or coordinate fields. Tap 'done', and then tap two bodies to connect via the pulley to each other. If no bodies are visible, double tap with two fingers to find the nearest one or pinch the world to zoom out. You can also tap the same body twice to create a pendulum.

Method 2: Double tap a body, scroll down to the end of the 'Body Constraint Forces' section, and tap the plus sign under 'Interbody Tension (via Pulley)'. After setting the fixed point coordinate location, tap a second body to connect them.

Note: Starting in V1.2, there is no slackness permitted in these constraints: as a result, the envisioned cables/rods can support both tensional (i.e. extensional) and compressional type forces. Put another way, the constraint is an equality constraint.

Remove a fixed point constraint

Either double tap a fixed point (pulley) itself, or double tap the body you wish to disconnect, and then scroll down to the 'Body Constraint Forces' section, and tap the minus sign to remove the most recently added interbody pulley tension constraint.

Add a direct distance constraint

This type of constraint enforces that the distance between two bodies remain fixed: envision an inextensible and incompressible cable (or rod) connecting two bodies. To add this kind of rigid constraint, either:

Method 1: Tap two bodies simultaneously, and select 'Add Rigid Interbody Tension'.

Method 2: Double tap a single body, scroll down to the 'Body Constraint Forces' section, and tap the plus sign under 'Direct Interbody Tension'. Then tap a second body to connect them.

Remove a direct distance constraint

Method 1: (V1.4) Double tap a constrained body, then tap on ‘Force Viewer/Editor’ button. Under the constraint forces section, identify the direct distance tension constraint you wish to remove (the other body to which it is joined is indicated), and tap the trash can icon.

Method 2: Double tap a constrained body, scroll down to the 'Body Constraint Forces' section, and tap the minus sign to remove the *most recently added* direct interbody tension constraint.

Track the constraints

On the iPad: In portrait mode, double-tap an open spot on the graphing area and select 'Add Constraint Tracker'.

On the iPhone/iPod Touch: In portrait mode, double tap on the region containing the clock and status message, i.e. the 'zeroth' graph, and select 'Add Constraint Tracker'.

This will produce a list of all constraints and their values (or errors in the case of normal forces) that the physics engine will attempt to enforce during evolution, via a constraint system of equations. Ideally, during evolution, these constraints SHOULD NOT CHANGE AT ALL, as constraint forces required to maintain normal and tensional constraints are computed exactly at each time-step. In reality, numerical integration error and accumulating round-off error means that constraints will eventually drift away from their initial values, though the drift might be very small: especially for small time steps and/or high order integrators. As such it is an effective way to compare integrators, and time-steps. If the time-step is too large, or the integrator too low-order, the constraints will drift appreciably. This drift will show up in the constraint tracker, and eventually the system might 'blow-up'.

Note: the velocity Verlet integration algorithm should not be used for constrained systems, as normal and tensional constraints are *implicitly* velocity dependent.

User Interface Help (by Gesture)

Body drag

Changes the position (initial condition) of the body.

Body long press + drag OR
Body tap + press + drag

Changes the velocity (initial condition) of the body. When the '(0,0) m/s' appears, moving your finger then assigns magnitude (speed) and direction to the body. For convenience, and only while collision detection is off, x and/or y components of the velocity are 'zeroed' if your finger remains close to the body and/or one of its axes to aid in precision.

Body double tap

Enters the 'body view' mode, in which you can change the body's kinematical state, mass, appearance, forces to which it is subject, and set up body tracking and graphing.

Two-Body Tap

The fastest way to setup interbody springs, custom forces and tension constraints is to tap on both bodies simultaneously. Select whether you want to add a spring, a custom force, or an interbody tension constraint, and you will then be prompted as necessary for further data needed to connect the two tapped bodies with the type of force you wish to apply.

World drag

Dragging on an open spot of the world view changes the world viewing boundaries: observe the coordinates at the corners of the view. It is does not change the 'size' of the world view.

These boundaries can be manually (and hence precisely) changed in 'World Settings' as well, which is accessed by double-tapping on an open spot of the world view. World bounds can also be locked from ‘World Settings’ which disables this gesture.

World pinch

Pinching the world view will zoom in or out on the world view: the 'width' and 'height' of the world view will change.

Note: Currently all bodies in Real Physics are modelled as point-like particles, and so their size is independent of the zoom level. Zooming will only lead to visual changes (other than the bounds coordinates themselves) if a constraint curve or surface is present. Nevertheless, bodies will be re-sized visually (up to the maximum or minimum permitted pixel sizes) if 'Resize bodies on zoom' is enabled within the 'Detect Collisions' section of 'World Settings'.

World double tap

Double tapping on an open spot of the world allows you to:

- set a constraint curve/surface (like a wire or the ground) of the form y = f(x)
- set a single or double inclined plane (you set the angle(s) and starting/ending points)
- create a fixed pt (pulley) tension constraint to which you can connect two masses via a light rope
- modify 'World Settings', and persistent app-level settings
- view the description of the current Real Physics world

World two finger double tap

Double tapping on an open spot of the world with **two** fingers finds the nearest body to the center of the current view, and then re-centers the view on that body. No change in scale (or zoom) occurs.

Graph swipe

The bottom third or so of your screen is where the time and world status is kept, as well as all of the physical property graphs that you may have requested (from the body views) are kept. On the iPhone/iPod Touch, by default there is only the time/status view, but as soon as you have requested that a property of a body be 'tracked' graphically, you can swipe along the bottom to access the property graphs in the order you requested them. On the iPad, the time/status view is always visible, as well as groups of four graphs at a time.

Graph long press + drag

After you have evolved a Real Physics world, or while it is evolving, you can read a specific point on a graph by tapping it and holding down your finger. After a moment (long enough that Real Physics determines you are not trying to swipe to view another graph), 'cross-hairs' should appear with the coordinates of the point on the graph displayed.

On a versus time graph, the first coordinate is time. As of V1.3, you can plot any two variables versus each other, in which case you are given instead (horiz_property,vertical_property) as an ordered pair followed by the time at which that phase plane point occurred. You can then track over the graph by dragging your finger.

Graph pinch (iPad Only)

You can enlarge a single graph to fill the entire graphical region by pinching outwards or restore it to its default size by pinching inwards.;

Graph double tap

Double tapping a property graph removes and destroys it from the list of graphs stored. (You cannot dismiss the time/status view.)

Graph two finger tap (V1.4)

Using two-fingers to tap on a graph allows you to export the data contained in the graph in a standard comma-delimited tabular format via e-mail, the clipboard, or any other app able to open a .csv (comma-delimited) file. The default time resolution of the data is 0.1 s, when the simulation rate is 1. Changing the 'simulation rate' away from real-time allows you to capture data at a higher (or lower) resolution.

Spring double tap

Double tapping an anchored spring on the anchor (whether it is free or attached to a single body) deletes it. (To delete an interbody spring instead, double tap one of the bodies, select the ‘Force Viewer/Editor‘, find the interbody restoring force you wish to delete and tap on the trash icon **OR** scroll down to 'Interbody Spring' in the explicit force section, and click on the minus sign to remove the most recently added interbody spring.)

Spring anchor drag

Dragging an anchored spring's anchor relocates the position of that anchor **AND** resets the equilibrium length of that spring so that when you let go, the body to which it is attached is again in equilibrium with respect to the spring. Put another way the spring is unstretched and uncompressed.

Move the body instead if you wish to preserve the original equilibrium length of an anchored spring.

Fixed point (pulley) double tap

Double tapping a fixed point (pulley) deletes it and the light cable with which it is associated. More importantly, this removes the tensional constraint force between the one or two bodies that had been connected to the fixed point.

Fixed point (pulley) drag

Dragging a pulley moves the inertial fixed point that represents the pulley's location. This does not preserve the 'length' of the light cable which passes over the pulley (i.e. through the fixed point), but the actual length is irrelevant for most simulation purposes. What's important is that it remains of fixed length during evolution of the simulation.

Real Physics Help - Major Concepts

Real Physics Bodies

The basic object in Real Physics is the 'body': a **point-like** body subject to various forces, and free to move in the plane of the screen. Bodies are drawn with extension only so that they can be visually represented and manipulated by touch. Their visual size (in points) is used only during interbody, or body-surface collision detection for the purposes of establishing the line of impact.

Rigid bodies are not currently implemented in Real Physics.

Pre-defined Explicit Forces

Real Physics comes with several standard pre-defined forces that can be applied explicitly to a body. They are:

1) constant gravity (with the gravity directed towards the bottom of the screen).
2) velocity-dependent resistive forces of the form $-bv$ (linear drag). You set b.
3) velocity-dependent resistive forces of the form $-cv^2$ (quadratic drag). You set c.
4) linear Hooke's law restoring forces of the form $-k(x-l_0)$, visually represented by a spring. You set k. These can be either anchored or interbody springs. For the former, you also set the direction relative to the body the anchor is positioned.
5) interbody inverse square of the form k/r^2. You set k: <0 attractive, >0 repulsive

Note: For kinetic frictional forces (semi-explicit), which require a normal force constraint, see the help section on 'Constraint forces'.

Custom Forces

With the help of Real Physics' built in function parser/symbolic calculator, you can apply virtually any position-dependent, velocity dependent or time-dependent force you can think of to the body. Thus you can simulate things like ‘external’ inverse square law forces, nonlinear springs, or external time-dependent driving forces.

You specify the custom force in terms of:
- the individual Cartesian components of the force
- magnitude and direction
- tangential and centripetal (aka normal) components.

Notes: 1) Tangential forces are applied parallel/antiparallel to a body's instantaneous velocity, and centripetal/normal/radial forces are applied towards the center of curvature of a body's instantaneous motion (This is great for specifying forces on bodies that move along curved, and especially circular paths.). 2) I have not figured out the best way to handle degenerate cases of normal-tangential specified forces, which occur, for example, when the body is at rest (or moves along a straight line) and the unit tangential and unit normal directions are not well-defined. Some bizarre behavior may occur when degeneracies are encountered! Suggestions welcome. 3) To constrain a body to a circular path using this kind of force (as opposed to a fixed point constraint), include a centripetal custom component equal to the body's $mass * v^2 / radius\ of\ desired\ path$. Negative centripetal components are interpreted as 'centrifugal' applied forces.

Custom Interbody Forces

One of the most powerful features of Real Physics is the ability to specify user-defined interbody forces to create truly interacting simulations. They can be created by either double tapping a body and selecting custom interbody forces (at least two bodies must exist), or using the two-body tap gesture: tapping two bodies at the same time. All interbody forces are assumed to be central: that is the forces always act along the line joining the bodies. This precludes forces like magnetic forces, but those require three dimensions anyways, and explicitly violate Newton's third law.

The 'variables' now available for custom interbody forces are:
- r = interbody distance,
- v = interbody relative speed,
- t = time

Positive interbody forces are interpreted as repulsive, and **negative** ones attractive.

For example, an attractive inverse cube law can be simulated with '$-1/r^3$', or one could re-simulate interbody springs (albeit less efficiently than the built-in springs) with an interbody force of the form '$-k(r-l)$', where k and l are specific numerical values for the effective spring constant and equilibrium separation of the bodies.

As some of the examples attest, you can also simulate, under certain situations multiple tensions by using a strong interbody relative-speed dependent custom force (or a very stiff spring: though these will generate well-known artifacts at large time-steps). However as of Version 1.2, a general constraint solver has been implemented to solve rigid multiple tensions exactly.

Constraint Forces

The two classes of constraint forces implemented by Real Physics are:

1) **normal force constraints**, which occur when a body is restricted to move along a curve or surface defined by a function of the form y=f(x);

2) **tensional constraints** between two bodies which occur when they are connected to each other via an inextensible/incompressible cable or rod

In turn there are two kinds of tensional constraints:

2a) **Direct interbody tensions**: these constraints attempt to enforce that the distance between two bodies is constant as the system evolves, as if they were connected by an inextensible/incompressible cable or rod.

2b) **Indirect Interbody tensions via fixed pulleys**: these constraints attempt to enforce the sum of the distances between two bodies to the fixed point represented by a pulley remain constant as the system evolves.

Note: The same body can be connected to both ends of the cable, to simulate systems like pendulums.)

Constraint forces are solved for exactly in Real Physics by the construction of a constraint matrix system of equations, which is solved robustly and efficiently at each time step. Therefore, as of Version 1.2 of Real Physics, multiple tensional forces can be applied to the same body, enormously expanding the types of systems that can be simulated. An overconstrained system will (usually!) be detected by the constraint solver, and reported as a singularity in the constraint system (and the simulation halted).

The validity of the constraints during evolution can also be tracked if it is suspected that a system is behaving poorly, and the integrator/time-step can be altered in response.

Frictional forces (via normal forces)

Once a body is subject to a normal force constraint, kinetic and static frictional forces can be 'added' to the body -- even though they aren't constraint forces in the traditional sense of the word, they require a contact interaction to exist, i.e. a 'normal force', and the magnitude of kinetic friction in the standard Coulomb model of friction, depends on the magnitude of the normal force. (Tension constrained bodies are not currently allowed to experience static friction halts: doing so with connected bodies would explicitly violate the tension constraints.)

The 'Solver'

By specifying a 'halting condition', Real Physics can be essentially used as a physics problem solver. Any property of a body that can be graphed can be used to define a halting condition; or you can have a simulation halt at a specific time in its evolution.

For example, consider a body sliding down a constraint surface that you've defined. Suppose you wish to know the speed or time at which it leaves the surface. Simply have Real Physics graph the body's speed and normal force magnitude (see UI help) as functions of time. Then:

1) tap the 'stop equals' icon.
2) tap the body you wish to set the halt on
3) set the halting condition on the body's normal force magnitude.

If you want the speed at which its normal force reaches zero, you'll need to request a graph of the speed versus time for that body.

Then run the simulation, and when the body leaves the surface the simulation will automatically halt unless the time-step is too large or the halting tolerance is too small: some experimentation might be required.

(On the iPhone/iPod Touch when the simulation halts, you then get the final speed by swiping over to the body's speed vs time graph where the final speed is displayed at halting time. On the iPad, the halting speed will already be displayed if one of the four spots for graphs is the speed vs time graph for that body.)

Note: If a simulation doesn't halt when you think it should, you may need to adjust the integration time-step (make dt smaller), the simulation rate (less than one), and the halting tolerance (smaller) in order to get the simulator to halt, or if you wish to produce very accurate stopping results. (See the related 'Fricionless sphere' bundled example.)

Constraint Curves vs Surfaces

Other than the obvious visual difference, in Real Physics surfaces are distinguished from curves in the following sense: the vertical (y) component of the normal force for a body on a surface can only be non-negative, so that a body can leave (fall off) a surface. In other words, the surface can only 'push' up on a body.

In contrast, there is no such restriction on the vertical component of the normal force for a body restricted to a 'curve': think about a bead threaded along a wire. Now the normal force can point in any direction, and the body cannot leave the curve (unless there is a singularity in the curve, or the presence of the step or sgn function in the curve/surface specification.

In terms of constraints: the normal force associated with a surface is an inequality constraint, whereas the normal force associated with a curve is an equality constraint.

Springs

Springs are the visual representation of a linear Hooke's law restoring force applied to a body. Real Physics supports two built-in types of linear restoring forces: anchored springs (defined by a stiffness in N/m and an initial direction), and interbody springs (defined by a stiffness alone).

Anchored springs are springs with one end located at a fixed point in the 'world'. They can be permanently attached to a body (and allow both compression and extension), or can be 'free' springs, and only allow compression.

Other kinds of restoring forces (e.g. isotropic or non-linear) can be specified by applying a custom force to a body, but will not have a visual representation associated with them.

Free Springs

Other than bodies themselves, the other 'primary' object in Real Physics is a light semi-anchored spring not associated/attached to any body, but one that can capture bodies that collide with their free ends. This allows you to simulate physics problems that involve bodies that collide with and compress (but not stretch) springs. See the examples. Tap the 'create' (plus) icon to get started.

Bodies that pass within the free end of the spring's capture radius attach themselves to the 'free' spring as if they collided with the spring (the spring will slightly resize its equilibrium length so as to prevent numerical artifacts and conserve mechanical energy), and allow compressions only: the body will separate again from the spring if the engine detects that the spring is being elongated.

Also, to prevent odd simulation behavior that can arise because a numerical simulation requires a finite capture radius, a body that loses contact with a spring can't be recaptured within a certain amount of time: your mileage may vary, and some experimentation might be required if you stray too far outside of the default settings.

As of version 1.2, you can also drop a body onto the end of a free spring, and the spring will 'remember' that it is free in the sense that the spring will only allow compressions and not extensions. This way you can easily set up 'pinball' type initial conditions, where the body moves away from the initially compressed spring once it returns to equilibrium.

Pulleys (Fixed Points)

Pulleys are the visual representation of fixed points over which a light cable can pass and connect two bodies to each other via a rigid tension constraint. Pulleys are modelled currently as massless, and with frictionless axles.

In version 1.2, the cables are modelled as rigid equality constraints, and hence allow both (ex)tensional and compressional forces. In particular no 'slackness' conditions are permitted.

Free-Body Diagrams

The basic pedagogical tool in solving physics problems in Newtonian mechanics is the free body diagram: a diagram in which all of the external forces that act upon the body are drawn with their magnitudes and directions indicated with arrows. Real Physics dynamically generates such diagrams to scale by default.\n\nEach body can also display the 'net force' vector acting on it: this is the vector sum of all the forces that act on the body, and by Newton's second law, points in the direction of the instantaneous acceleration. This is what Real Physics uses in each time step to evolve the system forward in time: hence

$$d\vec{p} = \vec{F}_{net} dt \qquad \textbf{OR} \qquad \vec{F}_{net} = m\vec{a}$$

Kinematically Constrained Bodies

In Newtonian mechanics, the normal state of affairs is to specify the forces acting on a body, and attempt to solve for the velocity and position of the body -- either analytically (if possible) or numerically (as in Real Physics). We might call these dynamically determined bodies. With kinematically constrained bodies this situation is inverted, and we instead specify their positions as functions of time: x(t) and y(t) in 2D Cartesian coordinates. This fixes their velocities, accelerations (and hence net forces) by direct differentiation instead of integration.

The ability to specify x(t) and y(t) directly, allows us to:
1) explore kinematics (the description of motion) independently of dynamics (the causes of motion). Kinematics is the normally the first topic taught in physics;
2) verify analytical solutions to dynamics problems as follows: create two bodies. Subject one to the forces of the problem to be solved, and kinematically constrain the other with the proposed solution. Evolve the system and compare positions and velocities;
3) easily simulate certain 'forced' problems, like a pendulum in an accelerating train or elevator.

Real Physics - General

What is Real Physics?

Real Physics is a touch-based, numerical, real-time, point-particle dynamics simulator in two dimensions. It is capable of modelling a wide class of high school and University level mechanics problems. It can also be regarded as a fairly general 2D physics engine with an exact constraint solver, capable of obtaining accurate numerical results for a large class of problems that cannot be solved analytically.

Dynamically generated free-body diagrams, trajectories, and versus-time graphs, the ability to specify essentially arbitrary forces, constraint curves and multiple tension constraints, and the 'halting condition' solver make Real Physics a very powerful tool in the exposition and application of Newtonian mechanics.

The Toolbar

The toolbar allows you to, left-to-right:

1) create a new body or new free spring and add it to the world within the current view;
2) 'evolve' (or stop the evolution of) the current world state forward in time;
3) 'rewind' to the world state present in the simulator at the instant the 'evolve' (play) button was last tapped (i.e. this undoes any changes made since the last time evolve/ 'play' was tapped);
4) load, save or create a new world;
5) obtain in-application help or load a bundled example;
6) set a simulator halting condition with respect to either time or the currently displayed graphical property.

The iPad additionally supports the ability to conveniently store and retrieve Real Physics Worlds in a stack with 7) push the current world onto the world stack, 8) empty the world stack, 9) pop the topmost world off of the stack and make it the current world.

Numerical Integrators (How Real Physics Works)

Conceptually, Real Physics 'evolves' the world through the numerical integration of Newton's second law to find changes in velocity from net force through $m\Delta\vec{v} = \int \vec{F}_{net}\, dt$ (Newton II, impulse-momentum theorem), and changes in position from velocity through $\Delta\vec{r} = \int \vec{v}\, dt$.

As is well-known, the primitive integration scheme directly implied by those relations (the so-called Euler method) is not sufficiently accurate, or stable for numerical simulation purposes. Even most video game engines, which typically use simple but fast algorithms like semi-implicit Euler are interested more in visual realism than in numerical accuracy. Thus, Real Physics implements a selection of higher-order numerical integration algorithms that are dramatically more numerically accurate than most video game engines.

Currently (V1.4), five different user-selectable integrators are implemented:

1) **Velocity-Verlet** (2nd order). Essentially only for pedagogical/comparison purposes, it should not be relied on except in simple, 'slowly' varying simulations or in simulations which do not involve constraints and non-dissipative velocity dependent forces. It assumes constant acceleration over interval dt, and then after updating the forces, the velocity is corrected as v(t+dt) = v(t) + (a(t)+a(t+dt))/2 * dt, thus achieving better accuracy and far better stability than a constant acceleration integrator would.

2) Pre V1.4 **Predictor-Corrector** (3rd order in velocity, 4th order in position). This is a custom variation of an Adams-Bashforth predictor, in conjunction with a post-force evaluation Adams-Moulton corrector, specially modified for application to Newton's laws (i.e. a second order ODE). Also known as a linear multistep method. Conceptually, it uses a 'history' of accelerations (a(t-dt) and a(t-2*dt) here) to obtain higher-order accuracy by matching with the Taylor series expansion of velocity and position to third order in velocity and fourth order in position. As efficient as Velocity Verlet in that it still requires only one force evaluation but more accurate. In V1.4, a second force evaluation after correction (PECE instead of PEC) can be applied as a user option, and is defaulted to if work calculations are enabled. Note this integrator is as of V1.4, completely superceded by the higher order predictor-corrector method (see below), which is equally efficient and much more accurate. Linear multistep methods such as these, must be kickstarted by a non-linear-multistep method (such as Runge-Kutta-Fehlberg discussed below), in order to build the initial 'history' of accelerations.

3) **Runge-Kutta-4** (4th order): The 'standard' and famous Runge-Kutta fourth order integrator. Works by computing a weighted average of force evalutions within a single time-step to update the acceleration. Computationally expensive (four force evaluations per time-step required), but quite accurate and stable. Sometimes referred to as the 'workhorse' of integrators, and the standard to which other integrators are compared.

4) V1.4 **Predictor-Corrector** (4th order in velocity, 5th order in position) By retaining three past accelerations instead of two, we obtain one higher-order with this new default predictor-corrector integrator at the same excellent computational cost (one force evalution per time step in PEC mode, two in PECE mode) as the previous PC method and velocity Verlet. This integrator is therefore 2-4 times as efficent as RK4 (important on processor constrained devices), and more accurate in my numerical testing. As with the lower-order PC method however, this integrator requires a higher order (or at least equivalent order) non-linear-multistep method to get it kickstarted.

5) Runge-Kutta-Fehlberg (5th order) This is the most accurate and stable integrator currently implemented in Real Physics, but requires six force evaluations per timestep (and hence is 3-6 times more computationally expensive than the PC methods) and can quickly bring iOS devices to their knees if dt is made sufficiently small and if the simulation is sufficiently complex. Used to 'kickstart' the predictor-corrector methods by generating the two or three 'past' accelerations required for those methods to work correctly without a large initial error. (Since the method actually computes both a 4th order and a 5th order approximation, as is well-known, it can be used as a basis of an adaptive time-step algorithm based on a desired error tolerance. This is not yet implemented in Real Physics however.)

All integrators can be directly compared by watching the constraint drift in the default V1.4 example (varying dt as well as the integrator).

Separately from the choice of integrator, the numerical time-step (equivalently the number of internal timesteps for each frame update) can also be modified manually in 'change world settings' view, but currently, there is no adaptive time-step algorithm implemented because of the real-time simulation emphasis of Real-Physics. If the simulation does not involve rapidly changing elements (e.g. high oscillation frequencies due to stiff springs), the default setting using the predictor-corrector algorithm at 25 updates/frame (1000 updates/second) should be mostly adequate... especially with the higher order variant default in V1.4. Where high-frequency elements or rapidly varying constraints are present (system changes occurring on the time scale of ~5 ms or less) or very high accuracy is needed, a simulation should be re-run using shorter time-steps, higher order integrators, or a simulation rate below 1 (real-time). Be aware however, using a very short-time step, especially in conjunction with the Runge-Kutta integrators or where there are many bodies, forces and/or constraints present, can easily slow the simulation well below 'real-time' evolution in the usual time versus accuracy tradeoff inherent in all numerical methods - especially on older devices.

V1.4 addition: Also, the **accuracy of numerical results can be dramatically improved by slowing the simulation rate below real-time** (another way to decrease dt), if accuracy is more important than real-time evolution.

Options and Defaults

Several commonly needed options are provided under the miscellaneous category within World Settings (double tap a free space on an open space in the world view): some are tied to and saved with a specific world, the others are persistent defaults applied to Real Physics as a whole.

You can set the sensitivity of the halting condition: if the value of the halting variable you chose is within tolerance (on an absolute basis), the simulator will halt if the time-step is sufficiently small.

You can set a default body size (important reminder: except for collision detection, body size is purely visual and has no meaning to the simulator: Real Physics is a **point-particle** simulator), and whether or not you want free-body diagrams to automatically be displayed for newly created bodies. Sometimes the FBDs can clutter a simulation or adversely affect drawing performance (especially if there are many bodies and/or forces) or aren't of interest; at other times, they are specifically the thing of interest.

You can set the default acceleration due to gravity (which is saved with the current world, and becomes the default for new worlds). You can choose whether or not to automatically assign a constant downward directed force of gravity to newly created bodies: this saves some time if you specifically want a near-earth-surface simulation versus a disentangled-from-Earth simulation.

Whether or not you want objects and touch-determined values 'snapped' to round values (depending on the zoom level) can be set from here, or disabled.

The current world background color can be set (e.g. for 'outer space' versus 'near Earth surface' coloration), and whether or not body orientation adjusts with the instantaneous velocity.

Restored in V1.4, you can set trajectory widths, and whether or not trajectories are anti-aliased. (If you notice a occassional stutter when trajectory updates are transferred to the non-OpenGL background layer, turn off trajectory anti-aliasing. This appears to be only an issue on the iPad 3, which quadrupled the pixels but only doubled the GPU power of its predecessor. It shouldn't be an issue on the iPad 2 or 4, iPad mini, or on any iPhone/iPod Touch which supports Real Physics.)

Questions? Bug reports? Feature Requests?

Contact realphysicssupport@kirkkaminsky.com.