NAME

xspringies − a mass and spring simulation system for X windows

SYNTAX

xspringies [*options*]

DESCRIPTION

xspringies is a simulator which allows you to interactively create and edit a system of masses and springs. The parameters of the masses and springs (such as mass elasticity and spring K) as well as those of the surrounding system (such as air viscosity and gravity) can be changed.
These systems can be loaded and saved into a file.

I guess you could use **xspringies** for real work, but it's really intended to be a time waster.

OPTIONS

−bg *color*
: Specifies the color of the window background. The default is black.

−d *dispname*
: Specifies the display screen on which **xspringies** displays its window. If the display option is not specified, **xspringies** uses the display screen specified by your DISPLAY environment variable.

−display *dispname*
: This option is the same as the **−d** option.

−fg *color*
: Specifies the foreground color of the window. The default is white.

−geometry *geom*
: Specifies the width, height, and location of the **xspringies** window. If the geometry option is not specified, **xspringies** uses default values. The geometry option has the format
[=][*width*x*height*][+/-*xoffset*+/-*yoffset*].

−nbb
: This option turns off the bounding-box optimization. To produce smooth animation, **xspringies** redraws the smallest screen region which contains all objects. For smaller objects, the performance improvement is very noticible -- this is the bounding-box optimization. By disabling it with this option, **xspringies** redraws the entire display window. On slower machines or larger window sizes, this can produce slow results. The main reason for using this option is if the bounding-box changes size rapidly, causing uneven animation.

−hl *color*
: Specifies the color of the button and selection highlights in the window. This defaults to the foreground color, or to green on color displays.

−rv
: Specifies that the foreground and background colors be reversed.

−st *thickness*
: Specifies the thickness of the springs in pixels (0 is the default value)

SUMMARY OF OPERATION

The left side of the **xspringies** window contains the controls, and the right side contains the display. Masses can be created and placed with the mouse when in **Mass** mode, and springs can be created when in **Spring** mode. Temporary springs that connect the mouse and any mass can be used to pull on objects. Masses and springs can be selected in edit mode, and moved around. Parameters of the masses and springs (such as Mass or Elasticity) can be set upon creation or if they are selected.

Forces (such as gravity) can be enabled by pressing the appropriate force button with customizable parameters. Environment parameters such as viscosity of the medium and stickiness of the walls can also be set. Each of the four walls can be disabled.

The animation/simulation is activated by pressing the **GO!** button.

The entire system (masses, springs and parameters) can be loaded and saved to files. **Xspringies** comes with many demonstration files.

CONTROLS

There are three types of controls (widgets) in **xspringies**. There are push buttons, checkboxes and sliders. The push buttons and checkboxes act in the obvious manner. If you click on them, they get activated.

Sliders are a little more compilicated. They consist of a left arrow button, a right arrow button, and a text box. The text box displays the current value. Clicking on this text box causes it to become highlighted. All text input then goes to the text box. After entering a value, return accepts it, and escape cancels.

The value displayed can also be changed by pressing the arrow buttons. Using the *Left mouse button* causes the value to be incremented or decremented by one step. The *Middle mouse button* is the same as the left mouse button, but holding it down causes it to scan by one step at a time. The *Right mouse button* scans like the *Middle mouse button* except that it scans 10 steps at a time.

EDITING MODES

There are three main modes in **xspringies**. These are *Edit*, *Mass* and *Spring.* In *Edit* mode, you can select, move and throw objects. In *Mass* mode, you can create masses. In *Spring* mode, you can create springs and tug on an object with a spring connected to the mouse.

More specifically,

Edit **mode:**

- *Left mouse button* selects objects.

  If you click on or near an object, it becomes selected, and all other objects become unselected. If you hold down shift while clicking, the object becomes selected (or unselected if it was already selected), and all other objects remain the same.

  If you do not click near an object, dragging the mouse causes a selection box to appear. Anything within the selection box when the mouse is released becomes selected. All other masses become unselected, unless the shift key was held down for the initial click.

- *Middle mouse button* moves objects.

  All selected objects move with the mouse. The masses are frozen in their positions after the initial click. They continue to move relative to the mouse movement until the middle button is released.

- *Right mouse button* throws objects.

  This acts the same way as moving objects with the middle button, except for the fact that the mouse velocity is transferred to all selected objects when the right mouse button is released.

  Note: a good way to stop an object from moving is to simply select it and click the right mouse button.

Mass **mode:**

When you click with the mouse, a mass appears. The mass takes on the values of the *Mass* and *Elasticity* sliders. The mass is placed when the mouse button is released.

Spring **mode:**

When you click with the mouse, if there is a mass nearby, one end of a spring is connected to it. The other end of the spring is connected to the cursor until the mouse button is released. If no mass is nearby when the spring is released, the spring is discarded. The new spring takes on the values of the *Kspring* and *Kdamp* sliders. The rest length of the spring is equal to the length of the spring when it was created.

- *Left mouse button* adds a spring between two masses while actively affecting the first mass.

- *Middle mouse button* adds a spring between the first mass and the cursor, actively affecting the first mass. The spring is discarded when the mouse button is released.

- *Right mouse button* adds a spring between two masses. The first mass is not affected by the spring until the spring is in place after the mouse is released.

OPERATION

Masses and Springs

Accelerations on the masses are calculated according to gravity (and other special forces), viscous forces, and by forces from the springs. When a mass collides with a wall, its velocity in the direction of the wall is reversed. The resulting velocity is multipled by the *Elasticity* of the mass. So, an elasticity of 0.0 causes an inelastic collision (it stops on the wall), and an elasticity of 1.0 results in an elastic collision.

If a mass is fixed, all forces on it are ignored. It simply does not move. Think of it as a nail (a really good one).

The *Mass* and *Elasticity* of a mass can be changed by selecting the mass and changing the values on the corresponding sliders. To make a mass fixed or unfixed, check or uncheck the *Fixed Mass* checkbox while the mass is selected.

A spring has three parameters associated with it. *Kspring*, *Kdamp* and rest length. The spring force is calculated as follows (according to Hooke's law):

F = - *Kspring* * (length - rest length) - *Kdamp* * (velocity in spring direction)

To change the *Kspring* or *Kdamp* of a spring, change the values of the sliders when the spring is selected. Pressing the *Set Rest Length* button changes the rest length of a selected spring to its current length.

Forces and sticky stuff

There are four special forces. They can be turned on and off by clicking their appropriate box. When highlighted, the force is on. Each of these forces has two parameters associated with it (for example, *Magnitude* and *Direction* for gravity). Only one forces's parameters are displayed at a time (below the force buttons). Which particular force is shown by a darker box around that force. This force selector box is moved to another force whenever a force is turned on.

Some of the forces are applied relative to some specified origin, or center point. By default, this center point is the center of the screen. It can be changed to be any one particular mass by selecting a single mass, and pushing the *Set Center* button. If no masses are selected, the current center is changed to be the center of the screen.

Center points are marked by a box around the center mass.

There are four forces that can be enabled. The first one, *Gravity*, acts in the familiar manner. It accelerates masses by the value specified by *Gravity* in a direction specified by *Direction.* The *Direction* is measured in degrees, with 0.0 degrees being down, increasing counter-clockwise.

The second force is a bit strange, and isn't real. Its a force which attracts the center of mass of all the objects toward the center point. It has a *Magnitude* and a *Damping* coefficient.

The third force is a force which attracts all masses toward the center point. This force has a *Magnitude* and an *Exponent* associated with it. The *Exponent* is simply how the force relation works. A value of 2.0 means inverse-square force (the force is inversely proportional to the distance squared). A value of 0.0 is a constant force independent of position. If the *Magnitude* of this force is negative, it becomes a repulsion force.

The fourth force is a wall repulsion force. Masses are repelled by a force from each wall that is on. This force has a *Magnitude* and *Exponent* associated with it. The *Exponent* behaves

similarly to that of the third force.

For the most part, most everything obeys $f = ma$. The only exceptions are wall bounces and wall stickiness. Another unphysical aspect is found in some of the special forces (the second and third ones). If a center point exists, that mass does not receive any force response from other masses due to the special force. In other words, these two special forces are not *equal and opposite* forces. They're pretty much just unreal.

Viscosity is a viscous drag force which applies a resistive force on the masses proportional to their velocity.

Stickiness is not a real force. When a mass hits a wall, it loses part of its velocity component in the direction of the wall (in an amount proportional to the *Stickiness).* If it loses all of this component, it remains stuck to the wall. It will remain stuck to the wall until a force (which exceeds an amount proportional to the *Stickiness)* pulls it off the wall.

Numerics

The internals of **xspringies** consist of a fourth order Runge-Kutta (RK4) differental equation solver. Consult a Numerical Methods text for more information. The *Time Step* that is used by this solver (the **dt**) can be set using the slider.

The solver can be selectively made into an adaptive RK4 solver using the *Adaptive Time Step* checkbox. An adaptive solver chooses the best time step value according to an error calculation. The error is not allowed to exceed the *Precision* value. Lower precision values result in smaller time steps. While this is more accurate, the simulation runs slower.

You will notice that some objects will tend to "blow up" easily. This is because the objects are unstable, or are sensitive to small numerical errors. An object will tend to "blow up" less with smaller time steps. By using an adaptive solver, the simulation can be made more accurate only when necessary. This results in a more stable system which runs at a reasonable speed.

Walls

There are four walls. In case you haven't guessed by now, they are the *top, left, right* and *bottom* walls. They are located at the window boundaries; they move with window resizes. Individual walls can be enabled and disabled by checking the corresponding checkboxes.

Walls are only one-way. An object moving from the screen toward a wall will bounce off the wall. But an object moving from off screen toward the screen will pass through the walls.

Nearby is the button to enable mass-to-mass collisions. Note that for large numbers of masses, this can get pretty slow.

Saving and Restoring State

The state of the world (at least for **xspringies**) includes all of the masses and springs, as well as the system parameters. By pushing the *Save State* button, the current system state is saved. By pushing the *Restore State button* the previously saved system state is restored (if no previous save was made, the initial state is used).

This is useful for temporarily saving a system configuration that you do not feel like setting up again (or saving to a file), that you might disturb with experimentation. If you break it, you can *Restore State* any number of times you like.

The *Reset* button resets **xspringies** to its initial configuration. All the masses and springs are removed, and the default system parameters are used.

Saving and Loading Files

The system state can be saved to and loaded from files. By pushing the *Load File* button, a previously saved file can be loaded. This will load up all the masses and springs that were saved, as well as the system parameters. Any previous state before the load is cleared. Pushing the *Save File* button saves this information in the designated file. The *Insert File* button is much like the *Load File* button, except that the current state is not cleared. Instead, only the masses and springs are loaded (the system paramaters are not changed), and are added to the current collection. If no objects are selected when the *Insert File* button is pressed, then all the objects in the file are selected after they are loaded.

The filenames are entered in the text window, which is located at the bottom right of the window. For consistency, the filenames should terminate with "**.xsp**". When a file is loaded or saved, this extension is automatically added if not added by the user. Standard emacs-like editing features are present. The following key controls can be used:

control-B move cursor backward
control-F move cursor forward
control-A goto beginning of line
control-E goto end of line
control-K kill to end of line
control-Y yank from kill buffer
control-D delete character under cursor
control-U erase all input
control-T transpose character under cursor with previous character
Escape exit from filename edit mode

By default, the directory which contains the **xspringies** files is present automatically. If the environment variable **SPRINGDIR** is set, then the default directory is changed to reflect it.

If a file error occurs (for example, the file does not exist), a beep is emitted.

Other stuff

The *GO!* button simply turns on and off the animation. With *GO!* turned off, **xspringies** uses up little CPU time.

The *Show Springs* checkbox controls whether or not the springs are drawn. If there are a lot of springs, animation may go faster with this option on. Sometimes an object will even look better with only the masses.

When placing masses or springs, objects can be placed in a gridlike fashion if the *Grid Snap* checkbox is enabled. Masses will be separated (vertically and horizontally) by the amount specified by the *Grid Snap* slider.

When the *Duplicate* button is pushed, all selected masses and springs are duplicated. The copy is left in the same place, unselected.

By pushing the *Select All* button, all masses and springs are selected.

By pushing the *Delete* button, or pressing the *Delete* key, all selected objects are deleted. Note that if a mass is deleted, all attached springs are also deleted (even if they were not selected).

The *Quit* button quits the program. This same effect is found by pressing the *Q* key.

FILE FORMAT

The **xspringies** file format is human readable. Each line consists of a four letter command string, followed by the parameters for that command. The file must start with the command "**#1.0**". (This 1.0 refers

to file format version 1.0, and not the version of xspringies being run) Each of the commands below is on a line by itself. There are no blank lines allowed. The file must end in a newline.

The file consists of the following commands:

cmas <current *Mass* value>
elas <current *Elasticity* value>
kspr <current *Kspring* value>
kdmp <current *Kdamp* value>
fixm <boolean value for *Fixed Mass*>
shws <boolean value for *Show Springs*>
cent <mass id number of center mass>
If there is no center mass (i.e. - center of screen is to be used), then the value of *-1* is used.
frce <force id number> <boolean active> <parameter #1 value> <parameter #2 value>
The <force id number> sequence is as follows:
0 - Gravity
1 - Center of mass attraction force
2 - Center attraction force
3 - Wall repulsion force
visc <current *Viscosity* value>
stck <current *Stickiness* value>
step <current *Time Step* value>
prec <current *Precision* value>
adpt <boolean value for *Adaptive Time Step*>
gsnp <current *Grid Snap* value> <boolean enable>
wall <boolean top> <boolean left> <boolean right> <boolean bottom>
mass <mass id number> <x position value> <y position value> <x velocity value> <y velocity value> <mass value> <elasticity value>
For each mass, the <mass id number> must be unique. They do not need to be in any order. If a mass is fixed, then the <mass value> field is negated.
spng <spring id number> <mass #1 id number> <mass #2 id number> <Kspring value> <Kdamp value> <rest length value>
For each spring, the <spring id number> must be unique. They do not need to be in any order. The order of the <mass id number>'s is not important.

All *values* are floating point numbers. All *id numbers* are positive integers, and all *boolean* values are non-zero/zero for True/False. It is possible to feed **xspringies** bogus values. It may produce interesting or amusing side effects, but will most likely cause an object to explode or **xspringies** to crash.

AUTHOR

Douglas DeCarlo (dmd@gradient.cis.upenn.edu)

Please send demo files, comments, suggestions, bug reports, bug fixes and enhancements.

With thanks to:

Elliott Evans
Bitmap slave.

Nathan Loofbourrow
I bothered him a whole lot about the user interface.

Drew Olbrich
The blame for "stickiness" falls on him, as well as a few other things.

Andy Witkin
For teaching a really great physically based modeling course at CMU. Many ideas (both

methods and interface) came from that class.

And thanks to the many other people who helped in testing **xspringies** and make some of the neat demo files, including James Helfrich, Brian Kelley, Patrick Lopez, Chris Newman and Jef Poskanzer.

RESTRICTIONS

xspringies runs faster on a monochrome display than on a color display (usually).

You probably don't want to run **xspringies** on a slow machine or a machine which does slow bit-blitting operations. Well, I guess you could... But you would be sorry for even trying.

Here is a good rule: If **xspringies** isn't fun to use, then your machine is either too slow, or it is overloaded. Or maybe you just aren't a fun person. :-)