Graphical Display of a Physics Simulation

Steven Oetjen
TJHSST Computer Systems Lab, 2006-2007

June 12, 2007

Abstract

A physics simulation, in order to adequately demonstrate physical laws and predict an unlimited number of scenarios, must implement a broad range of mathematical equations and provide the user with the ability to set up a scenario with whatever number of objects and arrangements of these objects that he desires. The goal of this project is to create such a simulation.

1 Introduction

The desired physics simulation and graphical display must govern a set of objects under the laws of physics, such as Newton's laws of motion. Accuracy and precision are important when measuring quantities for use in calculations, but it is also important to have a simulation process all objects and apply physical laws efficiently. The program must run in real time or at a speed chosen by the user. Such a simulation is worthwhile and valuable because it provides an opportunity to quickly and easily test or verify phenomena, utilizing a computer's ability to store, manipulate, and display data. Students may be attracted to the results of this project, especially students in need of an aid in their understanding of the physical world and how it behaves. The results can also be used to make or to confirm predictions on how certain scenarios will be resolved.

This simulation is intended to display objects placed by the user graphically and to display information about those objects on a graphical user interface. The project will allow the user to place any combination of objects, including particles, springs, and ramps, in a graphical display, input values for these objects, such as constants, coefficients, and variables, and run a simulation that will track these values and display the interactions and positions of the objects graphically in two-dimensions. The scope of the simulation will be limited to two-dimensions with particles, springs, and ramps. The concepts involved are kinematics, dynamics and Newton's laws, conservation of momentum and collisions, gravitational force, and electric charge and force. Variables such as mass, displacement, velocity, and charge, only to name a few, are required, as well as the relationships between these variables in the form of equations.

2 Background

A similar simulation was written in Java by Erik Neumann, named "My Physics Lab" [4]. Newmann's program consists of a series of java applets that run various scenarios based on user input for chosen quantities. Rather than having a large interface in which the user can create any scenario imaginable, with all the same laws governing each scenario,there are several pre-set scenarios, each with a set of laws governing it, as other laws might not be applicable. This organization allows Neumann to cover a broad range of objects, including linear springs, pendulums, colliding blocks, roller coasters, and molecules. It also graphs two quantities of the user's choice over time, and provides mathematical equations for all scenarios.

A model for pressure on a soft body was created by Matyka [8]. It takes into account gravitational and spring forces, and pressure. It also implements a first order Euler integrater to calulate force accumulations necessary for finding the volume of a body and pressure force distribution. He applies these calculations to a 2D soft ball model and a 3D pressure soft body model.

2.1 Kinematics and Integration

Two-dimensional kinematics involve the position, velocity, and acceleration vectors of an object [2]. Since velocity is the derivative of position, the linear approximation of x after a small time interval Δt is,

$$x = x_0 + v\Delta t$$

and the linear approximation of v after Δt is,

$$v = v_0 + a\Delta t$$

As $\Delta t \to 0$, these approximations become more accurate in calculating position and velocity, so a small time interval is necessary.

In order to make calculations more accurate, a modification of Simpson's rule is used to calculate velocity from acceleration. The equation of a parabola, $y = Ax^2 + Bx + C$, is calculated algebraically, using the points $(-t, a_{-2})$, $(0, a_{-1})$, and (t, a_0), where t is the time step, a_0 is the current acceleration value, and a_{-1} and a_{-2} are the previous two acceleration values.

$$A = \frac{y_3 - y_2}{(x_3 - x_2)(x_3 - x_1)} - \frac{y_1 - y_2}{(x_1 - x_2)(x_3 - x_1)}$$

$$B = \frac{y_1 - y_2 + A(x_2^2 - x_1^2)}{x_1 - x_2}$$

$$C = y_1 - Ax_1^2 - Bx_1$$

Once the quadratic equation has been calculated, the area under the acceleration curve during the current time step can be calculated by integration with the power rule from 0 to t.

$$\int_0^t (Ax^2 + Bx + C)dx = \frac{A}{3}x^3 + \frac{B}{2}x^2 + Cx]_0^t$$

This value is added to the velocity rather than the less accurate value of $a\Delta t$.

$$v = v_0 + \frac{A}{3}t^3 + \frac{B}{2}t^2 + Ct$$

2.2 Collision Detection and Response

A method for improved collision detection applies matrices and vector spaces to a sweep method over a polygon soup of triangles [7]. When all objects are two-dimensional circles, not polygons, a much simpler method to check for overlap is a simple comparison between the sum of the objects' radii and the distance between their centers.

$$r_1 + r_2 \geq \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

When a collision is detected, then for an elastic collision, the velocities of the objects involved must be changed so that both kinetic energy and momentum of the system are conserved [9].

$$\frac{1}{2}m_a v_a^2 + \frac{1}{2}m_b v_b^2 = \frac{1}{2}m_a v_a'^2 + \frac{1}{2}m_b v_b'^2$$

$$m_a v_a + m_b v_b = m_a v_a' + m_b v_b'$$

Solving these equations for v_a' and v_b',

$$v_a' = \frac{m_a - m_b}{m_a + m_b}v_a + \frac{2m_b}{m_a + m_b}v_b$$

$$v_b' = \frac{2m_a}{m_a + m_b}v_a - \frac{m_a - m_b}{m_a + m_b}v_b$$

the velocities of both objects after the collision are found. These equations can be extended to two dimensions by performing them for both an x- and y-component. Using these methods for collision response, however, does not take into account angular effects of collision. There are vector equations that can account for the loss of angular effects due to a separation of a collision into two components [1].

$$\mathbf{v}_2^A = \mathbf{v}_1^A + \frac{j}{M^A}\mathbf{n}$$

$$\mathbf{v}_2^B = \mathbf{v}_1^B + \frac{j}{M^B}\mathbf{n}$$

$$j = \frac{-(1+e)\mathbf{v}_1^{AB} \bullet \mathbf{n}}{\mathbf{n} \bullet \mathbf{n}(\frac{1}{M^A} + \frac{1}{M^B})}$$

By finding the scalar j, based on the elasticity of the two objects and the normal vector of there velocities before the collision, the velocities after the collision can be found. Momentum is still conserved in both components, and the effects of the initial angle of collision will be produced in non-head-on collisions.

2.3 Contact With Ramps and Response

In order to determine if a projectile is in contact with a ramp, the ramp can be represented by a linear equation of the form $y = mx + b$. A range of vertical positions is created to trigger a collision, the lowest being the bottom edge of the ramp, and the highest being the point at which the projectile begins to make contact, adding $\frac{r_{proj}}{cos\theta}$ to compensate for the radius of the projectile, and an additional $\frac{t_{ramp}}{cos\theta}$ to compensate for the thickness of the ramp. Two conditions must be true for contact between the ramp and projectile.

$$x_{ramp,1} \leq x_{proj} \leq x_{ramp,2}$$

$$mx_{proj} + b + \frac{t_{ramp}}{cos\theta} \leq y_{proj} \leq mx_{proj} + b + \frac{r_{proj}}{cos\theta} + \frac{t_{ramp}}{cos\theta}$$

These two conditions should be used if the angle of the ramp is less than 45°. If it is steeper, a different set of conditions should be used. The projectile's y-coordinate is compared to the y-coordinates of the ramp, and its x-coordinate is compared to the ramp's x-coordinate, based on its equation $\frac{1}{m}(y - b)$. The conditions should vary depending on the steepness of the ramp to avoid returning false for a collision that does occur.

If a projectile is in contact with a ramp, it exerts a force on it, and so by Newton's third law, that every action force has an equal and opposite reaction force, the ramp must exert an opposite force back. This force is perpendicular to the surface of the ramp. The equations for the force components exerted on a projectile due to the ramp, exerting horizontal force, are

$$F_x = -F_{horiz} sin^2\theta$$

$$F_y = F_{horiz} cos\theta sin\theta$$

while the equations for the force components exerted on a projectile due to the ramp, exerting vertical force, are

$$F_x = F_{vert} sin\theta cos\theta$$

$$F_y = -F_{vert} cos^2\theta$$

In addition, if a collision is occuring, the velocity of the projectile must be changed so that the component parallel to the surface of the ramp is unchanged, and the component perpendicular to the ramp is reversed.

$$v'_{\parallel} = v_{\parallel}$$

$$v'_{\perp} = -v_{\perp}$$

The components of the projectile's velocity are found parallel and perpendicular to the ramp, using the equations,

$$v_{\parallel} = v_x cos\theta + v_y sin\theta$$

$$v_{\perp} = -v_x sin\theta + v_y cos\theta$$

If the perpendicular component of the velocity is negative, meaning the projectile is going into the ramp, the velocity of the projectile needs to be changed so that the component parallel to the ramp is unchanged and the perpendicular component is reversed. In order to do this, adjustments are made using the following formulas:

$$v_x = v_{\parallel} cos\theta + v_{\perp} sin\theta$$

$$v_y = v_{\parallel} sin\theta - v_{\perp} cos\theta$$

By substituting equations for v_x and v_y into the equation for $v_{\parallel}$, the desired relationship is obtained.

$$v'_{\parallel} = (v_{\parallel} cos\theta + v_{\perp} sin\theta)cos\theta + (v_{\parallel} sin\theta - v_{\perp} cos\theta)sin\theta$$

$$v'_{\parallel} = v_{\parallel}(cos^2\theta + sin^2\theta) + v_{\perp} sin\theta cos\theta - v_{\perp} cos\theta sin\theta$$

$$v'_{\parallel} = v_{\parallel}$$

Similarly, by substituting those equations into the equation for $v_{\perp}$, the desired relationship is obtained as well.

$$v'_{\perp} = -(v_{\parallel} cos\theta + v_{\perp} sin\theta)sin\theta + (v_{\parallel} sin\theta - v_{\perp} cos\theta)cos\theta$$

$$v'_{\perp} = -v_{\parallel} cos\theta sin\theta + v_{\parallel} sin\theta cos\theta - v_{\perp}(sin^2\theta + cos^2\theta)$$

$$v'_{\perp} = -v_{\perp}$$

Ramps also may exert a frictional force on a projectile in contact with it. The force's magnitude is proportional to the normal force the projectile exerts on the ramp, and its direction is opposite the motion of the projectile. The constant of proportionality for this relationship, or the coefficient of friction, depends on the materials of the ramp and of the projectile [2].

$$F_{fr} = \mu F_N$$

2.4 Spring Force

Springs have a natural length, $l_{natural}$, the length at which they are at rest or equilibrium, and at which they exert no force. Different springs also have different stiffnesses, and this is reflected in a spring's stiffness constant, k. Springs exert a force on each end according to Hooke's Law,

$$F_{spring} = -kx$$

where x is the spring's displacement beyond its natural length. The negative sign is present because the force is a restoring force, meaning that it is in the opposite direction as the displacement [9].

Springs are also subject to a damping force,

$$F_{damping} = -bv$$

where b is the damping factor and v is the velocity component of the attached object parallel to the force of the spring. Again, a negative sign is present because the damping force acts against the motion of the spring [4].

2.5 Inverse Square Forces

Gravitational force is the force that attracts all bodies of mass towards each other with the force,

$$F_{grav} = G\frac{m_1 m_2}{r^2}$$

where G is the universal gravitational constant ($G = 6.6742 \times 10^{-11} \frac{N \cdot m^s}{kg^2}$), m_1 and m_2 are the masses of the two objects, and r is the distance between the two objects [5].

Electrostatic force, or Coulomb force, is the force that attracts objects with different charge signs and repels objects with like charge signs. The force behaves according to the equation,

$$F_{Coulomb} = k\frac{q_1 q_2}{r^2}$$

where k is the Coulomb constant ($k = 8.988 \times 10^9 \frac{N \cdot m^s}{C^2}$), q_1 and q_2 are the charges of the two objects, and r is the distance between the two objects [6]

3 Development

There are several requirements that must be met to ensure success of the simulation. Inputs must be doubles able to be parsed, given by the user through JTextFields, to any given precision, in the range of doubles able to be stored in memory, and given as often as desired by the user between runs of the simulation. Outputs must be displayed as objects in their respective positions and values in JTextFields and JLabels, to an accuracy of a $\pm 1\%$ error, in the range of doubles able to be stored in memory, and given as frequently as the Timer fires. Calculations for all the objects must be completed within the time elapsed during the timer delay in order to keep the simulation time proportional to real time. This proportionality will be created by the time scale factor the user provides to speed up or slow down the simulation as desired.

Success for most of the requirements can be observed qualitatively, such as if objects appear with different names and colors, if values are being recorded to a certain decimal place, if objects overlap or collide properly, or if ramps are drawn with the correct slope and thickness. The requirements that demand quantitative testing are the accuracy of the output, and the efficiency of processing time. The output can be compared to theoretical values or examined in terms of a known equation, and the processing time can be judged by comparing elapsed simulation times to elapsed real time for different numbers of active objects.

The project was developed by dividing functions into several iterations of design, programming, and testing. The tasks of these iterations include implementing each concept and set of equations and laws. These groups are kinematics, dynamics, momentum, gravitation, and electrostatics. Each iteration proceeded with the following steps:

Design Classes, methods, and variables were added or modified, and the function of each element and the means of communication of information between relevant classes was determined.

Programming The necessary changes to the program were made. All programming was done in Java, using Swing and AWT packages for graphics and GUI components.

Testing To test a given implementation of a law, the user will input arbitrary values for all necessary fields involved in or dependent on that law. The output generated by these random types of input will be compared to theoretical values derived from mathematical formulas. To analyze the error, a percent error calculation will be performed on these two values, and the accuracy will be gauged by that percent. In this way, the accuracy of the program's ability to model predictable phenomena can be measured and displayed.

The project was divided into several sections. The implementation of equations and principals of physics for kinematics, summation of forces, collision detection and response, and spring motion, was done in phases, as other sections of the project were completed. The other sections involving the format and structure of the program were the GUI structure and layout, graphical display, support of multiple objects and types of objects, support of cartesian and polar coordinates, scenario file storage, and improved integration methods.

The major classes involved are the `GraphicsPanel`, which displays the graphical output, and the `GUIPanel`, which displays the numerical output and reads in the user's input. Other critical classes include `Projectile`s, `Ramp`s, and `Spring`s, the objects themselves which store their own quantities, and `ProjectileInputs`, `RampInputs`, and `SpringInputs`, which provide a GUI layout for input to each object. The architecture is designed so that changes in the future can be easily accomodated. The program is modularized so that new features can be added in without interfering with existing ones. For example, additional types of objects can be created by creating a class for the object and a class for its inputs. Room on the GUI can be allocated for that input, and functions can be added to draw and implement appropriate laws for those objects.

4 Results and Conclusion

To be a successful and useful simulation, the program must implement all equations and relationships correctly, and process all data efficiently. Specific requirements must exist for processing time, kinematics, collisions, ramp forces, ramp collisions, spring motion, and inverse square laws.

4.1 Processing Time

The program's calculations must not take longer than the timer's firing delay, so that the program runs proportionally to real time. For the processing time analysis, the simulation was run for 20 seconds in simulation time, for various time factors. The real time that elapsed during the 20 seconds of simulation time was recorded, it was adjusted to a time for a scale factor of 1, and a percent error was calculated between the theoretical 20 seconds and the experimental adjusted times.

Elapsed Real Time for 20 Seconds of Simulation Time with Various Time Scales

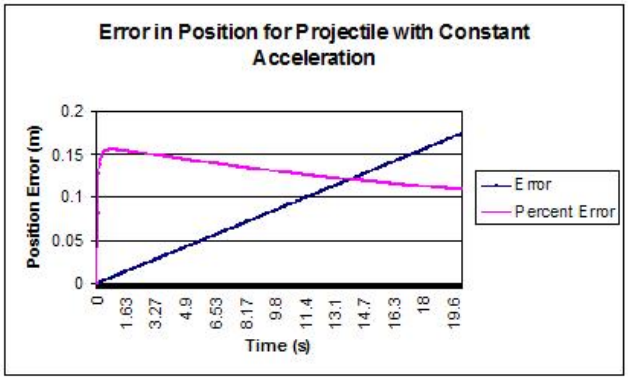
time scale	elapsed real time	adjusted simulation time	percent error
0.25	95.277	23.819	19.096
0.5	46.346	23.173	15.865
1	23.270	23.270	16.350
2	11.547	23.094	15.470
4	5.807	23.228	16.140

The program tends to lose efficiency in the lower end of time scale factors, as there was a notable increase in the percent error for the smallest time scale tested. It should be noted, however, that accuracy is greatest when the time scale, and therefore the Δt of the time step, is smallest. The user may therefore sacrifice efficiency for accuracy with a smaller time scale.

Although the percent error for the remaining 4 time scales is still large, they are all fairly consistent. So the program still runs proportionally to real time, even if the constant of proportionality is not exactly equal to the time scale factor. The error between the input factor and the actual constant of proportionality is about equal for all time scale factors 0.5 and greater. This disparity is most likely due to the unavoidable display time for Java graphics and updating labels on the GUI, not due to calculations.

4.2 Kinematics

Using any given inputs, a projectile's position at any time later should be accurate to its theoretically predicted position within $\pm 1\%$. To test the program's accuracy with kinematics, the program was given inputs of $x_0 = 0.0m$, $v_{x,0} = 5.4m/s$, and $a_{x,0} = 0.26m/s^2$, and the outputs for x were recorded over 20 seconds. These values are compared with theoretical values, and a percent error calculation was made. Theoretical values were calculated using the equation for constant acceleration, $x = x_0 + v_{x0}t + \frac{1}{2}a_{x0}t^2$.



The error for position increases linearly with time while there is constant acceleration. The percent error however, starts at 0 and quickly peaks at about 0.16%. It decreases afterwards and approaches about 0.1%. The accuracy of the kinematics, driven by the

modified Simpson's Rule for integration, meets the requirements, as percent error for a projectile moving with constant acceleration is well under 1%.

4.3 Collisions

For any collision between two objects, the momentum of the system before the collision must be equal to the momentum after the collision, and for elastic collisions, the same is true of kinetic energy. To test this function, two objects with different masses and velocities were collided, and the initial and final momentums and kinetic energy values were calculated.

Two Objects Before and After Collision

Object	v_x, v_y	p_x, p_y	KE	v'_x, v'_y	p'_x, p'_y	KE'
A	20.0	20.0	200.0	5.618	5.618	56.180
	0.0	0.0		-8.989	-8.989	
B	0.0	0.0	0.0	14.382	14.382	143.820
	0.0	0.0		8.989	8.989	
	Σp_x	20.0		$\Sigma p'_x$	20.0	
	Σp_y	0.0		$\Sigma p'_y$	0.0	
	ΣKE	200.0		$\Sigma KE'$	200.0	

Momemtum was conserved in both the x- and y-components, and kinetic energy was conserved as a scalar. The objects bounced off each other with the appropriate angle, meaning that angular effects of a collision were correctly produced.

4.4 Ramp Forces

When a projectile is exerting a force on a ramp, the ramp must exert an equal and opposite force on the projectile. With gravity acting on the projectile, it should slide down the ramp, accelerating parallel to the ramp's surface. A range of angles, both positively and negatively sloped, will be tested to ensure projectiles accelerate parallel to the surface in all cases. The angles tested were -60°, -45°, -30°, 0°, 30°, 45°, and 60°, all measured from the positive x-axis.

Acceleration Direction of Projectile Sliding Down Various Ramps

θ_{ramp}	$\theta_{acceleration}$
-60.0	-60.0
-45.0	-45.0
-30.0	-30.0
0.0	0.0
30.0	30.0
45.0	45.0
60.0	60.0

The projectile always is accelerated by the ramp in the same direction the ramp slopes in. Therefore, this test verifies that for any slope of the ramp, a projectile will

accelerate in that same direction. When a coefficient of friction greater than 0 was input, the projectile always accelerated in the same direction, but at a slower rate. The rate for each test was correct according to the sum of the forces in both the x- and y-components for the projectile.

4.5 Ramp Collisions

When a projectile has a velocity component perpendicular and into the surface of a ramp, and they collide, the angle of incidence must equal the angle of reflection after the projectile bounces off. A range of angles, both positively and negatively sloped, will be tested to ensure projectiles bounce off of ramps correctly. The angles to be tested will be -60°, -45°, -30°, 0°, 30°, 45°, and 60°, all measured from the positive x-axis. For each velocity adjustment test, the projectile had the same initial velocity of 10 m/s, directed at -90° clockwise from the positive x-axis, and the angle of the ramp varied.

Angles of Incidence and Reflection of Projectile Reflected Off Various Ramps

θ_{ramp}	θ'_v	θ_i	θ_r
-60.0	-30.0	30.0	30.0
-45.0	0.0	45.0	45.0
-30.0	30.0	60.0	60.0
0.0	90.0	90.0	90.0
30.0	150.0	60.0	60.0
45.0	180.0	45.0	45.0
60.0	-150.0	30.0	30.0

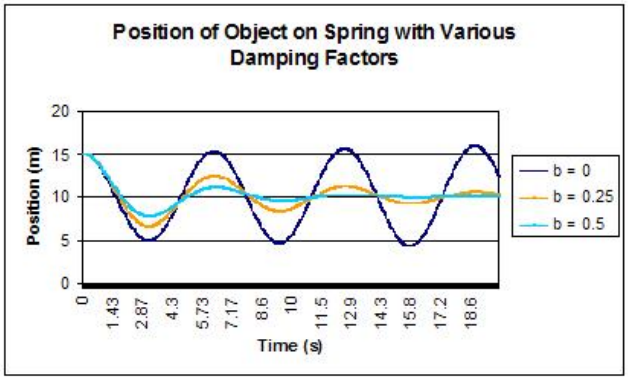
The angle of incidence is always equal to the angle of reflection, no matter what the angle of the ramp is. Therefore, this test verifies that for any slope of the ramp, a projectile will bounce off of correctly, as $v'_{\parallel} = v_{\parallel}$ and $v'_{\perp} = -v_{\perp}$.

4.6 Springs

Spring force was tested by recording the position and acceleration of an object attached to a spring for 20 seconds with three different damping factors, $b = 0$, $b = 0.25$, and $b = 0.5$. The initial position of the projectile was $x_0 = 15.0m$, the spring constant was $k = 1.0N/m$, and a natural length of $l_{natural} = 10.0m$.

When the spring has a damping force of 0, the projectile moves along a sinusoidal path centered at its natural length, 10. The projectile's peak position, however, increases over time. Because of the propagation of error during motion where acceleration is not constant, the amplitude of the position function increases over time. It is significantly less than previous methods used for integration, such as a Riemann sum.

When the damping factor is increased to 0.25, the position continues to center around the spring's natural length of 10, but the amplitude decreases over time. When the damping factor is increased to 0.5, the position curve approaches the natural length of 10 very rapidly, barely performing two cycles before it's motion is extremely small around the natural length of the spring.



4.7 Inverse Square Forces

Gravitational and Coulomb forces were each analyzed both qualitatively and quantitatively. A small object is attrated to an extremely massive object, and the greater the product of the two masses, the greater the force on the two objects. Two objects with like charges repel, and two objects with opposite charges attract. The greater the magnitude of the charges, the greater their acceleration towards each other. When two objects' values were recorded at a given point in time, their acceleration always was correct due to the circumstances of mass, charge, and position.

4.8 Conclusion

The simulation is successful in most areas, and is certainly useful for predicting physical phenomena and aiding students in their understanding of physics. The simulation runs proportionally to real time, even if not exactly at the time scale factor. All formulas for effects of collision have been verified by testing. Kinematics in one dimension are calculated well within $\pm 1\%$, but when acceleration is not constant, such as with spring motion, error does propagate over time. Other forces, such as forces exerted by ramps, frictional forces, gravitational forces, and Coulomb forces, are summed correctly and verified by testing.

References

[1] C. Hecker, "Physics Part 3: Collision Response", 1997. Definition Six, Inc. http://chrishecker.com/images/e/e7/Gdmphys3.pdf (March 5, 2007)

[2] D. C. Giancoli, Physics: Principles with Applications, Prentice Hall, 1995. http://wps.prenhall.com/esm_giancoli_physicsppa_6/0,8713,1113739-,00.html (November 2, 2006)

[3] E. J. Fuselier, Jr., "Investigations into Quadratic Curve Fitting", Southeastern Louisiana University Department of Mathematics, 2000. maa.mc.edu/proceedings/spring2000/Edward-Fuselier.pdf (May 11, 2007)

[4] E. Neumann, "My Physics Lab - Physics Simulation with Java", 2004. http://www.myphysicslab.com/ (November 2, 2006)

[5] E. W. Weisstein, "Gravitational Force", Wolfram Research, 2007. http://scienceworld.wolfram.com/physics/GravitationalForce.html (February 20, 2007)

[6] E. W. Weisstein, "Coulomb's Law", Wolfram Research, 2007. http://scienceworld.wolfram.com/physics/CoulombsLaw.html (February 26, 2007)

[7] K. Fauerby, "Improved Collision Detection and Response", 2003. http://www.peroxide.dk/papers/collision/collision.pdf (December 11, 2006)

[8] M. Matyka, "How To Implement a Pressure Soft Body Model", 2003. http://panoramix.ift.uni.wroc.pl/ maq/soft2d/howtosoftbody.pdf (November 2, 2006)

[9] R. Fitzpatrick, "Collisions in 1-dimension", 2006. http://farside.ph.utexas.edu/teaching/301/lectures/node76.html (November 30, 2006)

[10] R. Vawter, "Hooke's Law - Spring Force", Western Washington University, Department of Physics and Astronomy, 2006. http://www.ac.wwu.edu/ vawter/PhysicsNet/Topics/SHM/HookesLaw.html (February 5, 2007)