Accelerated Particles vs. Metastatic Cells

New Mexico

Supercomputing Challenge

Final Report

April 7 2010

Team 30

V. Sue Cleveland High School

Members:

Matthew Bradley

Jeremy Wright

Ryan Cortez

Kevin Clay

Teacher:

Debra Loftin

Project Mentor:

Nick Bennett

Contents

Executive Summary

Cancer is often a highly debilitating disease in most animal species. Radiation and chemotherapy are common modern treatments, yet these can prove harmful and can leave lasting consequences even after their treatment. Particle accelerators allow for high amounts of energy to be delivered through small spaces and allow for accuracy on the atomic level. This project was initially intended to model the collisions of various particles with cancerous cells, collisions that could be replicated in the real world with actual cancerous tissues in a particle accelerator to create enough heat to kill the cell. This model, coded and visualized in NetLogo, would display particle motion and make calculations to determine whether a cancerous entity the particle would collide with would be destroyed by the heat (a reflection the real world would hopefully represent). Our final project, while limiting in particle variability, does accurately track electron motion at near-light speeds (including accounts for mass-variations at such speeds) and displays gradients of energy transfer during collision with metastatic (cancerous) cells. Our program tackles this task from a mainly physics based point of view.

Problem

Many cancer treatments are dangerous and inprecise methods of attacking the ailment. High energy linear transfer from a near light speed electron in collision with a cell to the aformementioned cell could be a more precise, less detrimental alternative to current superficial cancerous tumors. This is because electrons transfer heat via friction during a collision, therefore potentially melting and therefore breaking down the DNA nucleotides, which cancerous cells have a harder time repairing than healthy ones. Based purely on effectiveness (disregarding costs), is this "Electron Therapy" a pheasible method of treating cancer tumors on the skin or other external surface?

Research

Electricity and Magnetism

Particle accelerators are true to their namesake. Though individual models vary from sophisticated pieceof technology to sophisticated piece of technology, they tend to bear a common goal: the achievement of propelling particles at extremely high velocities. The means by which this goal is achieved varies, though for our model, we are assuming an electromagnetic-based propulsion system. Such a system would guide and accelerate the particles based on their negative charge by using carefully placed positively and negatively charged plates or wiring.

The term “Electricity and Magnetism” is somewhat redundant, as electricity and magnetism are both two parts of the same fundamental force. This force is known as electromagnetism. This force differs from gravity mainly in the aspect that it can be repulsive as well as attractive. This feature of electromagnetic force plays a bit part in particle acceleration and the control of the testing apparatus.However, electromagnetism's use continues within the test, as at the quantum level, electromagnetic waves characterize the transfer of heat through photons, mainly at the infrared portion of the electromagnetic spectrum. This allows us to control the motion of the fired electrons (to an extent) as well as heat transfer, improving the precision of eliminating the attacked cell.

Relativistic Phenomena

Once any object with mass approaches speeds of a notable percentage of the speed of light (about 10% c), it begins to experience certain alterations to its actions and composition (i.e. mass change) as widely accepted and theorized within Einstein's Theory of Special Relativity and Mass-Energy Equivalence. The relavent sections of these theories state that as an object approaches the speed of light, it gains mass relative to a stationary viewer and according to its velocity as compared to that of light within a vacuum. Such phenomena becomes an important factor within particle acceleration tests, especially within vector forces and collisions. As an object's mass changes, so does the force required to accelerate it (as well as the momentum of the object itself). Within our program, such phenomena is taken into account and will be covered later.

Thermodynamics and Entropy

The transfer of heat from one body to another is also a large factor within our program, as it is the main considered method of cancer treatment within the program. Thus, thermodynamics became a boon to the process. Once the electron (or any body, for that matter) collides with a cell (or any OTHER body, for that matter), especially within an inelastic collision, much of the kinetic energy of the colliding bodies becomes heat as the Work done between them is transferred via friction and therefore output as internal energy. However, the amount of energy transferred is never quite what was put in. This property is known as entropy and embodies the randomness of such reactions, much to the annoyance and confusion of thermodynamic physicists for years. For the computer's sake, we therefore added the assumption that entropy is negligible.

Methodology

Existing Methods of Treatment:

There is a variety of different methods of cancer treatment the most common being chemotherapy, radiotherapy, and surgery depending on the type of cancer. Each has its own benefits and each its side effects.

Surgery as always is a dangerous procedure, but it is a very common form of treatment for many ailments including some forms of cancer. Surgical excision of cancerous tumors and tissues is often conducted via amptutation, the direct physical removal of the tissues. Often, such a treatment type is used as a last resort, and does not cover the entirety of the cancerous areas, leaving metastatic cells to replicate and repopulate the tumor area, causing a resurgence of the cancer.

Chemotherapy, despite new advances, is also incredibly dangerous, with equally debilitating side-effects. Chemotherapy works off of the principle of using poisonous chemicals which target only fast replicating cells. While this does fit the description of cancerous cells, it also describes hair follicles and sometimes bone marrow cells. Flaws like these in such treatment bring with them a staggering list of side-effects. **(www.chemocare.com/managing)**

Radiotherapy is a fairly useful treatment method which uses high power gamma rays, often mediated by photons, to attack and irradiate the cancer cells, killing them. However, the radiation can have long term health effects, including weakening of the immune system, bone weakness, and sometimes, radiotherapy can serve as a carcinogen itself. **((http://www.lymphomation.org/side-effect-radiation.htm)**

Our Method:

Our method is a plausible alternative to the above mentioned cancer methods when used upon superficial tumors. It involves the acceleration of beta particles (electrons and positrons, though electrons specifically in this situation) to measurable fractions of light speed and the direction of these hyper-accelerated electrons towards a dedicated tumor. The particles are to then collide with the cancer cells, transferring their kinetic energy through friction into heat. This heat will then melt and break down the nucleotides which form cellular DNA. Since these cancerous cells are mutated and damaged, they are ineffective at repairing damaged DNA and will therefore be rendered incapacitated, unable to replicate. This causes no lasting irradiation damage, and any healthy cells actually damaged by the treatment should be able to repair their own DNA in short order.

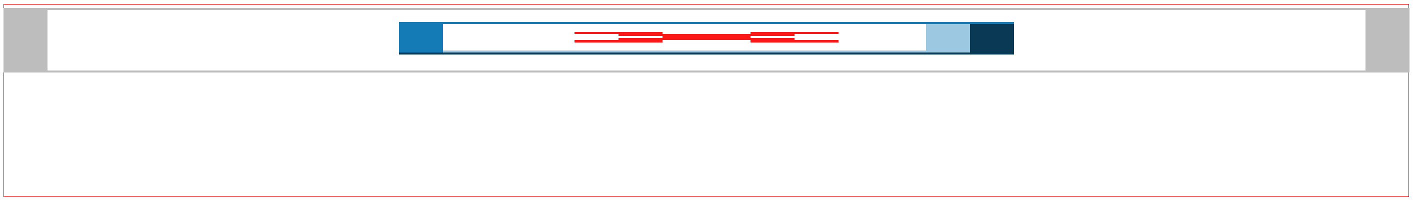
The reason electrons are used as opposed to bulkier subatomic particles is because of their substantially low "Bragg's Peak". The "Bragg Peak" of a particle is the point or "peak" in a collision where the greatest amount of energy is transferred. Larger particles such as protons have a higher Bragg Peak. This translates to a greater potential for surrounding, healthy tissues to be damaged. The goal of this model is to demonstrate a minimization of this, yet still guarantee the destruction of cells. As such, the numerical quantities of the particles in our program appropriately reflect those of electrons.

Acceleration:

The basic principle on which particle accelerators work is the use of electromagnetic force to push and bump a particle around and eventually launch it out into a particle stream. Once this has been accomplished, The electrons move into the controlled field, gaining or losing acceleration based on multiple factors. Acceleration itself is the second derivative of displacement,

$$acceleration = d''(t) = d/t^2$$

This means it is the rate of the rate of displacement. The standard physics unit is meters persecond, however, in our program, the units are expressed in millimeters per picosecond. Using standard units defeats the purpose of our program as a visual model as the particles move at speeds beyond both the human eye's AND most computers' abilities to process information. By lowering the scale of speed, the motion of our particles, be they cells, electrons, or otherwise, becomes apparent.



Relativity:

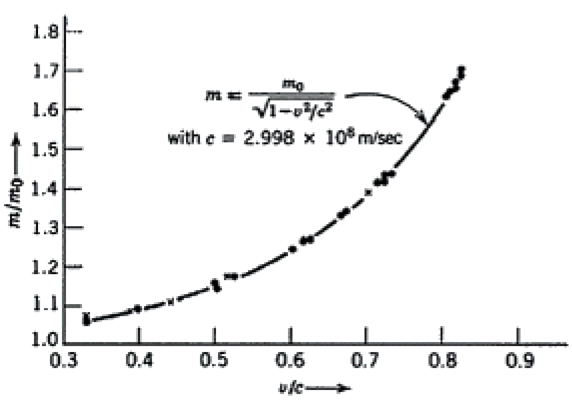
However, certain problems arise in physics when a particle is accelerated to large fractions of *c,* being the speed of light in a vacuum. These relativistic phenomena were incorporated into the program as well, by means of the equation:

$$E = mc^2$$

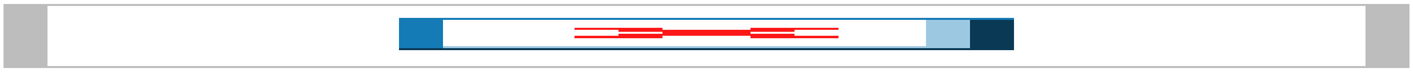
and the modified version:

$$M_{rel} = \frac{M_0}{\sqrt{1 - \frac{v^2}{c^2}}}.$$

These are used to express the relativistic concept that mass increases as velocity *v* approaches *c*. At small fractions of c, this change is negligible, but the mass gain increases exponentially once one passes about 20% the speed of light. This concept can be shown with a relation of $\frac{m_{rel}}{m_0}$ to $\frac{v}{c}$ as a function:



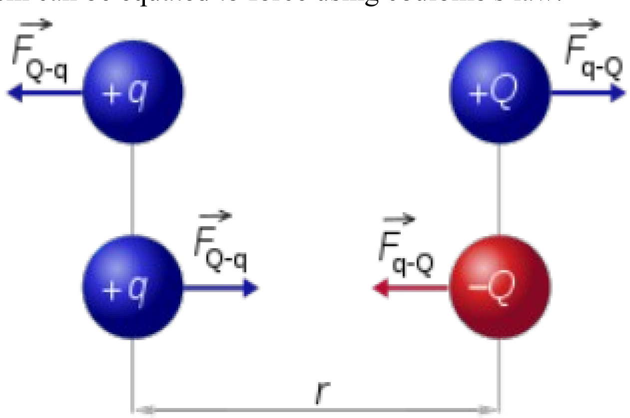
img courtesy of blazelabs.com <http://www.blazelabs.com/pics/relativisticcurve.gif>



Electromagnetism:

In our experiment, we attempt to keep the area as controlled as neccesary. The biggest factor in this is ensuring that electrons do not escape the test field. This factor is eliminated by placing three charged plates into the test chamber. One at the highest y-line, one at the lowest, and one at the farthest x-line. The two on the top and bottom are negatively charged, the actual charge of which controllable as an Independent variable, which causes them to repel the electrons that near them, keeping them within the designated area. The remaining plate has a positive charge designed to capture and pull any stray electrons to the end, assumably recycling them back into the accelerator.

However, electrical charges are not simply a result of the charged plates, but of the electrons as well. The charge of two objects and distance between them can be equated to force using coulomb's law:

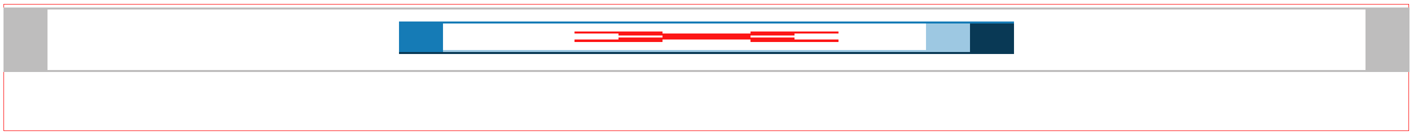


$$F_{electric} = k_c \frac{q_1 q_2}{r^2}$$

$$|\overrightarrow{F_{Q\text{-}q}}| = |\overrightarrow{F_{q\text{-}Q}}| = k \frac{|q \times Q|}{r^2}$$

img courtesy of wikipedia <en.wikipedia.org/wiki/coulomb's_law>

where $F_{electric}$ is the electric force, q_1 is the charge of one object, q_2 is the charge of the other, r is the shortest distance between the objects, and k_c is coulomb's constant, which is equal to approximately $8.99 \times 10^9 \frac{Nm^2}{C^2}$.



Multiple electrons (N-Body Systems):

Although normal projectiles act independently of each other when fired, the electrons in our simulation are unique in their charged status. In addition to the electromagnetic forces of the testing apparatus, they also experience forces from the proximity they share between each other. They interact, powerfully repelling each other the closer they approach. Within our code, these interactions are included to preserve realism in what is known as an N-body system.

N-body systems are ways for programs to handle a near-limitless number of integers that all must interact with each other. In the case of our program, these are the electrons. Whenever a new electron is fired, it is incorporated into this N-body system. As part of their update-velocity function, each electron is told to check its position in accordance with the other electrons in the N-body system. Based on Coulomb's Law, the electrons' forces of repulsion will all apply to each other in addition to the vectors already established by the electromagnetic patches. This calculation is made in accordance with every electron present as every electron is incorporated in the N-body system, thus making as little as three or as many as hundreds of electrons all interact with each other with regard to their charges and positions. *(note: In all algorithims: ← means to store the right object as the left)*.

Thermodynamics:

In the NetLogo coding, we used mathematics to calculate then simulate that amount of energy transfered from the electron to the cancerous cell. Since cancer cells are just normal cell that have been mutated to attack you body and replicate themselves, they don't have any improved resistance. This means that the mutated cells can still sucome to the same [weaknesses as any other cells of the same type.] The problem is killing the cancerous cells without killing the healthy ones. The heat transfer between Electron and cell is modeled by standard calorimetry, assuming that energy transferred q is equal to kinetic energy of the electron:

$Q = \Delta T \times m \times C$,

Where Q equals energy transferred, ΔT is the change in temperature, in celsius or kelvins, m is the mass of the object the heat is tranferring to, and C is the specific heat capacity of the medium. In the case of m and C, both are derived from that of human flesh/skin cells. Those of the nucleotides are an average of the specific heat capacities of the elements that form them: N, O, C, H.

As a simulation:

Assumptions:

The following is a list of assumptions included in our model:

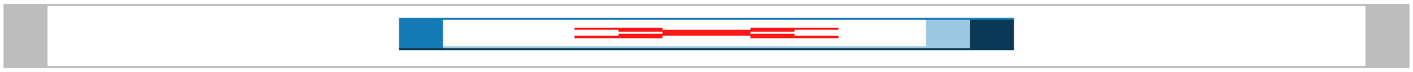
- We are working in a 2-dimensional vacuum.
- Electron-cell collisions are perfectly *in*elastic, and all kinetic energy in such collisions is transferred uniformly throughout the cell as heat.
- We have an infinite supply of electrons that are either discounted and removed from the program when they have lost all of their energy OR are recycled by the apparatus should a collision not take place.
- Cells are initially non-ionized, static, and openly available.

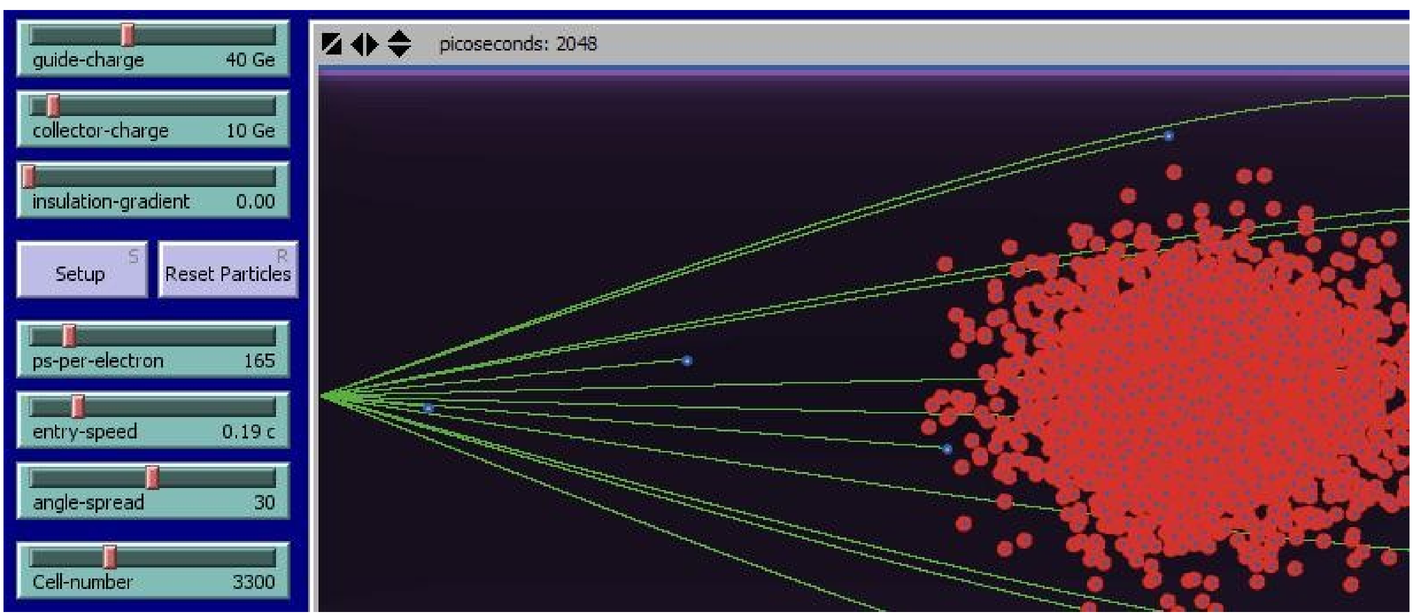
The Environment:

The program environment consists of a tube-like apparatus with charged plates on all sides of the rectangular ‘tube’ save for the side where electrons are fired from. Electrons are ‘fired’ from the center-left side at a mass of cells on the right. Many aspects of the firing can be controlled by the user to simulate different circumstances. These include:

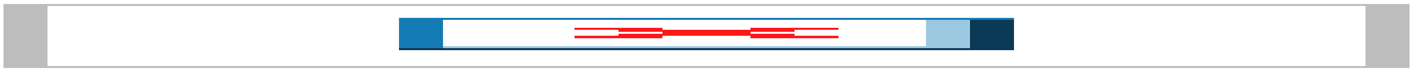
- Picosecond Per Electron Delay (how much time passes before another electron is fired.)
- Charge of the Guiding Plates (-)
- Charge of the Collecting Plate (+)
- Initial Velocity of the electron.
- Insulation gradient of the testing apparatus.
- # of cells present in the apparatus.

In addition, we also have a setting that allows for the display and color of the paths the electrons take.





Upon electron-cell collision, the program calculates the amount of energy transferred based on the electron's current kinetic energy. If the heat transfer is enough to increase the temperature of the cell's nucleotides beyond their melting points, then the cell will be reported 'Dead.' This process continues for every electron fired indefinitely (though of course, no cells will be reported dead if there are no cells left or if an electron completely misses. In the latter case, the electron is simply recycled by our testing apparatus).



Essential Algorithms:

Electromagnetic Forces (Patches):

The program uses NetLogo-based entities known as patches to handle a majority of the electron motion. A patch is a small area of the program's environment that can hold its own values to be applied to other entities in the program, so long as that entity is told to check the patch it is on and alter its values accordingly. The program takes the distance between the various electromagnetic plates and their charges and then, for each patch, assigns a force vector based on Coulomb's law (See Appendix A). The electrons are told to, for every 'tick' of motion, check the patch they are on and alter their force and, consequently, acceleration and velocity vectors. These alterations of velocity, etc, are all accomplished by code-appropriate modifications of kinematics equations. This altogether allows for an efficient manner of modeling the wave-like motion of our particles.

Here is psuedo-code for the lines that the program uses in the "setup patches" function that apply the forces to the patches *(note: In all algorithims: ← means to store the right object as the left)*.:

In order to setup the patches.

Ask guide patches: charge←charge-slider-scale x guide-charge *This changes the user*

guide-charge to usable units

by multiplying by a constant.

Ask collector patches: charge← -(charge-slider-scale) x collector-charge *Same as above,*

except with the collector charge.

Ask field patches:

$$fieldx1 \leftarrow ((1 - Insultiongradient) \times k_{cmod}) \times \Sigma(\frac{patchscale \times (xcor_{patch} - xcor) \times q}{patchscale \times distancetopatch}) Of_{guide}$$

$$fieldx2 \leftarrow ((1 - Insultiongradient) \times k_{cmod}) \times \Sigma(\frac{patchscale \times (xcor_{patch} - xcor) \times q}{patchscale \times distancetopatch}) Of_{cllct}$$

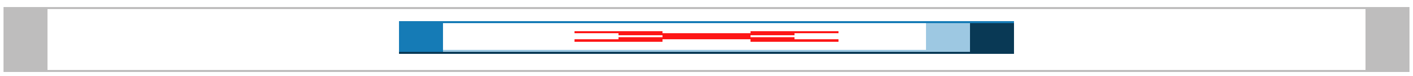
$$fieldx \leftarrow fieldx1 + fieldx2$$

$$fieldy1 \leftarrow ((1 - Insultiongradient) \times k_{cmod}) \times \Sigma(\frac{patchscale \times (ycor_{patch} - ycor) \times q}{patchscale \times distancetopatch}) Of_{guide}$$

$$fieldy2 \leftarrow ((1 - Insultiongradient) \times k_{cmod}) \times \Sigma(\frac{patchscale \times (ycor_{patch} - ycor) \times q}{patchscale \times distancetopatch}) Of_{cllct}$$

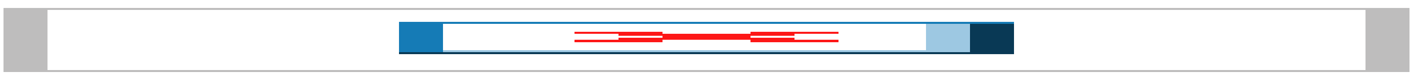
$$fieldy \leftarrow fieldy1 + fieldy2$$

$$field_{magnitudes} \leftarrow \sqrt{fieldx^2 + fieldy^2} of_{fieldpatches}$$



This whole algorithim does a preapplication of all the forces that a patch will exert electromagnetically on the electron based on the fact that we already know the electron's charge, and the guide (negative, the ones that repel the electrons) and collector charges (positive, the one that attracts electrons). It checks the distance between the field patch and charged plates, and using coloumbs law, can determine its component force vectors in the x and y directions. (*note: q=charge, cllct means collect, as in collector patches.)*

In addition, the program enables the user to account for various testing environments simply by changing the forces of the patches accordingly. Should the apparatus be assumed to be insulated, the user can simply input the insulation gradient and the program will adapt and alter the patches accordingly. If the apparatus is capable of maintaining higher or lower charges, then those too can be altered, and the patches in turn. Incredibly convenient is the ability for patches to hold a color. We've taken advantage of this by altering the colors of the patches based on the magnitude of the force they exert. A sort of 'well' can be observed by the culmination of the electromagnetic forces. Electrons have a tendency to follow this well in a wave-like manner, depending on their initial launch spread, a phenomena that can be observed when the "draw-paths" option is turned on for a span of time. The simplistic flexibility patches offer allows for a wide variety of realistic, controllable testing conditions.



Relativistic Mass/Velocity/Momentum:

Relativity is a huge part of the realism of the electron motion in the program. The electron, as it approaches *c,* the speed of light in a vacuum, gains mass exponentially. This mass gain was incorporated into the program and recycled in order to accommodate for the mass change. *(note: In all algorithims: ← means to store the right object as the left)*.

M0←1 — *The rest mass of an electron is defined as 1, the units have been altered accordingly*

$S \leftarrow \sqrt{speedx^2 + speedy^2}$ *The speed is constructed out of its component vectors, speedx and speedy*

$Mrelative \leftarrow \frac{M_0}{\sqrt{1-\frac{S^2}{c^2}}}$, *when c= speed of light in a vacuum in millimeters per picosecond.*

This mass is then used to alter the effects of the electric force on the electron by effecting the mass and therefore the acceleration:

$acceleration\ scale \leftarrow timescale \times \frac{1}{mass_{rel}}$ *This directly alters the acceleration due to electric force by dividing the electron charge by the new mass.*

speedxnew←speedx+(fieldx x acceleration scale) *This incorporates in the change in mass through acceleration, where fieldx is the magnitude of the electric force on the electron in the x direction*

speedynew←speedy+(fieldy x acceleration scale) *Same as above, but in the y direction.*

newxcoordinate←xcoordinate+(speedx x speed scale) *This adds the number of patches the electron has moved according to speed. The speed-scale is simply a constant used to reconvert the calculations into usable units*

newycoordinate←ycoordinate+(speedy x speed scale) *Same as above, but with the y direction*

These algorithms normally serve their purpose of improving realism with current theories very well. However, they bring with them their own problems which result due to the limitations of computers and such a large tick scale (a picosecond, 1E-12 seconds) relative to Planck's time unit (about 5.4E-44 seconds). As the electron approaches the speed of light (in a vacuum), its mass approaches infinity, meaning that the force required to accelerate it also approaches infinity. However, as the electrons approach the collecting plate, the force on the electron increases, approaching infinity. On such a large tick scale, the force increases before the mass has the chance to accommodate, resulting in an object moving faster than the speed of light, and therefore having an imaginary mass. However, such is a physical impossibility, and results in an error statement of either division by zero, or a radical of a negative number, the first of which resulting in an undefined number, and the second resulting in an imaginary number. In order to avoid this error, a Cheating Algorithim is incorporated in order to "Fix Physics" In multiple parts of the code. The algorithim functions as so:

Physics Fixing Cheat Algorithm

$S \leftarrow \sqrt{speedx^2 + speedy^2}$ *The speed is formed out of its component vectors, speedx and speedy*

If $\frac{c}{S} > 1$ *If the ratio of c to the current velocity is greater than* *then k←1* *one, meaning the object is traveling slower than the*

else: k← $.99 \times \frac{c}{S}$ *speed of light, then don't alter the object's velocity.*

end *However, if it WOULD be going faster than light, then it alters its velocity to .99c*

speedxnew←speedx x k

speedynew←speedy x k

This particular algorithm removes a great deal of stress from the computer system, leaving it to avoid having to change tick units to the planck time, which a standard computer, especially a lower-grade one, cannot handle well. However, it DOES indeed live up to its name as the Physics Fixing Cheat Algorithim, as this does to a degree limit the closeness to which an electron can get to the speed of light. However, the electron accelerating to .99 c within the tumorous area is highly improbable, and therefore negligible.

Collisions and thermodynamics:

In order to model collisions, the program checks to verify whether a cell and an electron's radii are in each other's direct vicinity. If the distance between the two is such that the two are occupying the same space, just as in physics, a collision occurs. Upon registering a collision, the program uses a recorded quantity for the kinetic energy of the electron.

Standard Kinetic Energy:

$KE = \frac{1}{2}mv^2$<!--[if !vml]--><!--[endif]-->

Relativistic Kinetic Energy:

$$E_k = \frac{m_0c^2}{\sqrt{1-\frac{v^2}{c^2}}} - m_0c^2$$

Basic calorimetry energy transfer:

$$Q = \Delta T \times m \times C$$

The collisions remain relatively basic in this program. It is assumed that once an electron collides with a cell, it becomes part of an inelastic collision and transfers all of its kinetic energy into the cell in the form of heat. By modifying the calorimetry equation above, one can solve for the change in temperature, demonstrating what temperature the cell would uniformly reach under the aforementioned assumptions. standard kinetic energy equations are generally unused in the program, substituted for relativistic kinetic energy instead in order to increase realism, as with the relativistic mass.

Results

As Shown by the model:

The model displays a consistent, rapid transfer of energy to the mass of metastatic cells. With the assumption that "incapacitation" is achieved when a cell reaches its melting point, the cells are totally incapacitated with a vast degree of consistency when an electron collides, due to the massive amounts of energy transferred at near light velocities. This is of course, going by our many assumptions, and therefore in a perfectly ideal situation. The data trend at a 200 picosecond delay between fired electrons and a thirty degree angle spread showed a strong trend to the following systems of equations for a rate of cell death (d(t)) to time (in microseconds):

$$d'(n,t) = (\frac{1}{5}n)t$$
$$upperlimit = e$$

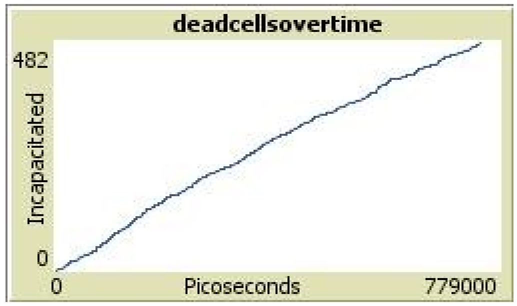
Where n equals starting amount of cells in apparatus, t equals time in microseconds, and e is the number of electrons fired. However, this data trend is predicted to change to an almost asymptotic curve with a horizontal asymptote at y=n. This is caused by the fact that as the number of cells to shoot at decreases, so does the likelihood of an electron hitting a cell, causing the rate of killing to slow.

Program Verification

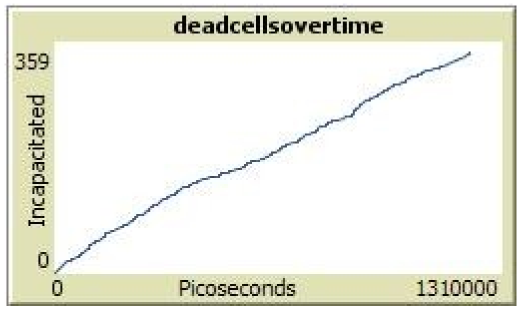
To be honest, the program features more discrepancies than we would have liked, mainly within the realm of thermodynamics. However, the electron motion was a process totally verified by known and accepted physics equations. The kill rate from the program was incredible, at about 0.5% of the cells being incapacitated in the first microsecond with ranges from 1000 to 10000 cancerous cells. Even at a lower electron firing rate, the kill rate is still impressive, providing easily more than enough of the expected kill rate, about 2% in a second. However, this kill rate doesn't hold, as as the number of vulnerable cells decreases, so does the "accuracy" of the electrons. Despite this, the short time taken to reach the point where a reduced kill rate becomes measureable is noteworthy, at below the time it takes for blood to circulate around an average human body, therefore automatically beating out chemotherapy, which usually circulates through the blood and targets all fast-replicating cells. Though the experiment was pursued under impossibly ideal conditions, even when entropy plays its role, we as a team have no doubt that the electron therapy will still prove effective.

Conclusion and Analysis of Results

Though the program had many imperfections, it produced an impressive, and steady (to an extent, see above) kill rate. In a mere three-fourths of a microsecond, the electron stream (at 200ps firing delays and a thirty degree angle spread) killed nearly 500 metastatic cells. The rate at first tends to follow a linear pattern, as seen here:



The function holds a similar trend, though with a smaller slope, when there is a 500ps firing delay. However, the kill rate still maintains a linear trend and is actually proportional to the previous graph:



Acknowledgments and Thanks

Nick Bennett: Without his help, the bulk of the programming, and consequently, our understanding of it, would never have gone underway. In addition, he supplemented our programming knowledge with real-world information and considerations that only served to improve the project as a whole.

Debra Loftin: Our team sponsor, Mrs. Loftin managed to help keep us on track and provided snacks and other forms of sustenance during many of our after-school coding crunches. Thanks for all the help and for providing the opportunity!

The Supercomputing Challenge: For providing students like us with a grounds and motive to further our knowledge and the application of it.

Penn. University team: we might not have continued with this idea if they didn't start working with it to begin with, encouraging us and showing us the true breadth and worldwide intellectual and medical impact our work could have.

Appendix A: Equations and Formulas

Coulomb's law

$$F_{electric} = k_c \frac{q_1 q_2}{r^2}$$

Mass-Energy Equivalence

$$E = mc^2 - m_0 c^2$$

Relativistic Mass

$$M_{rel} = \frac{M_0}{\sqrt{1 - \frac{v^2}{c^2}}}$$

Basic Kinetic Energy

$$KE = \frac{1}{2} mv^2$$

Relativistic Kinetic Energy

$$E_k = \frac{m_0 c^2}{\sqrt{1 - \frac{v^2}{c^2}}} - m_0 c^2$$

Acceleration

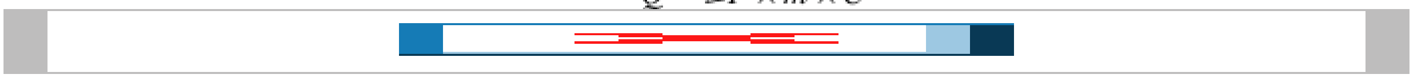
$$a = \frac{d}{t^2}$$

relative to force

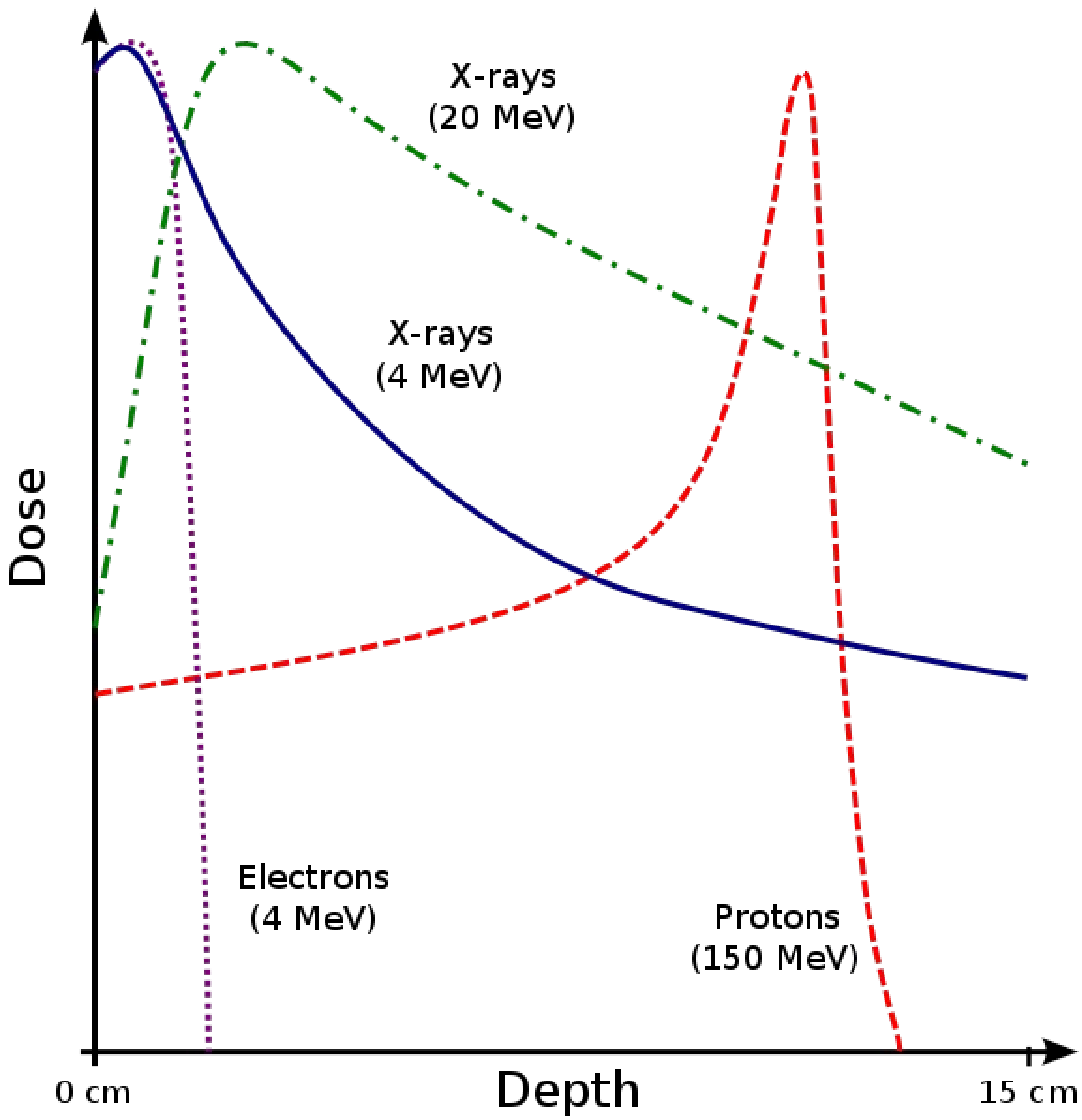
$$a = \frac{F}{m}$$

Basic Calorimetry

$$Q = \Delta T \times m \times C$$



Bragg's Peak



Img courtesy of

Wikipedia. <en.wikipedia.org/wiki/particle_therapy>

Appendix B: NetLogo Code

; Units of measure

```
; Charge: elementary charge (e); proton has charge 1e, electron has charge -1e ; Distance: millimeter (mm) ; Time: picosecond (ps) ; Mass: electron rest mass (me) ; Force: model force (mf); synthetic force expressed in (me)(mm)/(ps)^2 ; Energy: model energy (mod-e); 1 mod-e = 1.0977E+13 joules
; Conversions ; 1.60217646E-19 C = 1 e ; 1 m = 1E+3 mm ; 1 s = 1E+12 ps ; 1 me = 9.1093821545E-31 kg = 9.1093821545E-28 g ; 1 N = 1.09776929E+09 mf ; 1 cell = 1.0E-09 g = 1E-9/9.1093821545E-28 me = 1.09776929E+18 me ;;  ;
; Constants ; Speed of light (c) = 2.99792458E+08 m/s = 2.99792458E-01 mm/ps ; Coulomb's constant = 8.9875517873681764E+09 Nm2/C2 = 2.53263834E-13 (mf)(mm)^2/e^2
; Model scales ; 1 patch = 2.5 mm ; 1 tick = 1 ps
globals [   coulombs-constant   guide-patches   collector-patches   field-patches   electron-mass   speed-of-light   electron-charge   electron-display-size   electron-path-color   cell-display-size   cell-size   cell-mass   charge-slider-scale   patch-scale   tick-scale   speed-scale   save-draw-paths?   collisions   dead ]
 patches-own [   charge   field-x   field-y ]
breed [electrons electron] breed [cells cell]
turtles-own [   speed-x   speed-y ]
electrons-own [   saved-speed-squared   color-e ]
cells-own [   heatF   heatC   tempK   incremental-energy ]
to startup   clear-patches   clear-turtles   reset-ticks   setup-globals   setup-shapes   import-world "electron-world.csv" end
to setup   clear-patches   clear-turtles   reset-ticks   setup-globals   setup-shapes   setup-patches   export-world "electron-world.csv" end
to reset   clear-turtles   reset-ticks   setup-globals   setup-shapes   clear-drawing   set-current-plot "deadcellsovertime"   clear-plot  end
to setup-shapes   set-default-shape electrons "electron"   set-default-shape cells "circle" end
to setup-globals   let charged-set no-patches   set guide-patches      (patches with [(pycor = max-pycor) or (pycor = min-pycor)])
```

```
set collector-patches (patches with [(pxcor = max-pxcor) and (not member? self guide-patches)]) set charged-set (patch-set guide-patches collector-patches) set field-patches (patches with [not member? self charged-set]) set electron-mass 1 set electron-charge -1 set coulombs-constant 2.53263834E-13 set speed-of-light 2.99792458E-01 set charge-slider-scale 1E+09 set electron-display-size 1 set electron-path-color green set cell-display-size 3 set cell-size 0.015 set cell-mass 1.09776929E+18 set patch-scale 1 set tick-scale 0.1 set speed-scale (tick-scale / patch-scale) set collisions 0 set dead 0 set save-draw-paths? draw-paths? end
to setup-patches let min-magnitude 0 let max-magnitude 0 let field-magnitudes [] ask guide-patches [ set charge (charge-slider-scale * guide-charge * electron-charge) set pcolor blue ] ask collector-patches [ set charge (- charge-slider-scale * guide-charge * electron-charge) set pcolor red ] ask field-patches [ set field-x ( (1 - insulation-gradient) * coulombs-constant * (sum [patch-scale * ([pxcor] of myself - pxcor) * charge / ((patch-scale * distance myself) ^ 3)] of guide-patches + sum [patch-scale * ([pxcor] of myself - pxcor) * charge / ((patch-scale * distance myself) ^ 3)] of collector-patches) ) set field-y ( (1 - insulation-gradient) * coulombs-constant * (sum [patch-scale * ([pycor] of myself - pycor) * charge / ((patch-scale * distance myself) ^ 3)] of guide-patches + sum [patch-scale * ([pycor] of myself - pycor) * charge / ((patch-scale * distance myself) ^ 3)] of collector-patches) ) ] set field-magnitudes (sqrt ([field-x ^ 2 + field-y ^ 2]) of field-patches) set min-magnitude (sqrt min field-magnitudes) set max-magnitude (sqrt max field-magnitudes) if (min-magnitude != max-magnitude) [ ask field-patches [ set pcolor ( scale-color violet (sqrt (field-x ^ 2 + field-y ^ 2)) (min-magnitude - (max-magnitude - min-magnitude) / 10) (max-magnitude + (max-magnitude - min-magnitude) / 5) ) ] ] end
to shoot-electron create-electrons 1 [ set size 2 set heading (90 + (random-float angle-spread) - (random-float angle-spread)) set speed-x (entry-speed * speed-of-light * sin heading) set speed-y (entry-speed * speed-of-light * cos heading) if electron-color = "Rainbow" [ set color-e ("Rainbow") color-change ] if electron-color != "Rainbow" [ set color-e ("Static") set color electron-color ] if (draw-paths?) [ pendown ] ] end
to make-cells create-cells Cell-number [ set size cell-display-size let x-age (random-normal (min-pxcor / 3 + 2 * max-pxcor / 3) (world-width / 15)) let y-age (random-normal (0.5 * (min-pycor + max-pycor)) (world-height / 10)) setxy x-age y-age let current-cell-tempK (310.15) ] end
to move if (save-draw-paths? != draw-paths?) [ ifelse (draw-paths?) [ ask electrons [ pendown ] ] [ clear-drawing ask electrons [ penup ] ] set save-draw-paths? draw-paths? ] let shot-counter (ticks / ps-per-electron) let deviation (abs (shot-counter - round shot-counter)) if (2 * deviation * ps-per-electron < tick-scale) [ shoot-electron ] if (not any? cells) [ make-cells ask cells [ set color blue ] ] update-electron-velocities ;ask cells [ ; move-cell ;] ask electrons [ move-electron check-collision ] tick-advance tick-scale end
to color-change let electron-list (sort electrons) ask electrons [ ] foreach electron-list [ ask ? [ if electron-color = "Rainbow" [ set color-e ("Rainbow") ] if electron-color = "Rainbow" and color-e = "Rainbow" [ let color-counter (ticks / 200) let deviation (abs (color-counter - round color-counter)) if (2 * deviation * 200 < tick-scale) [ let color-value (random-float 140) set color color-value ] set color-e ("Rainbow") ] if electron-color != "Rainbow" or color-e != "Rainbow" [ set color-e ("Static") ] ] ] end
```

```
to update-electron-velocities   let electron-list (sort electrons)   ask electrons [     set saved-speed-squared ((speed-x ^ 2) + (speed-y ^ 2))   ]   foreach electron-list [     ask ? [       let acceleration-scale 0       let other-electrons (sort electrons with [who > [who] of myself])       let real-mass 0       foreach other-electrons [         let force-x 0         let force-y 0         let acceleration-x 0         let acceleration-y 0         let v-x 0         let v-y 0         let v-squared 0         let lorenz-contraction 0         let speed 0         let k 1         ask ? [           let distance-cubed ((patch-scale * distance myself) ^ 3)           set force-x (             (1 - insulation-gradient) * coulombs-constant * electron-charge ^ 2             * (patch-scale * ([xcor] of myself - xcor) / distance-cubed)           )           set force-y (             (1 - insulation-gradient) * coulombs-constant * electron-charge ^ 2             * (patch-scale * ([ycor] of myself - ycor) / distance-cubed)           )           set v-x (speed-x / speed-of-light) ; expressed as a fraction of c           set v-y (speed-y / speed-of-light) ; expressed as a fraction of c           set v-squared (v-x ^ 2 + v-y ^ 2)           set lorenz-contraction (sqrt (1 - v-squared))           set acceleration-x (lorenz-contraction / electron-mass * ((1 - v-x * v-x) * force-x - v-x * v-y * force-y))           set acceleration-y (lorenz-contraction / electron-mass * ((- v-x * v-y) * force-x + (1 - v-y * v-y) * force-y))           set speed-x (speed-of-light * v-x - tick-scale * acceleration-x)           set speed-y (speed-of-light * v-y - tick-scale * acceleration-y)           set speed (sqrt (speed-x ^ 2 + speed-y ^ 2))           set k (min list 1 0.99 * speed-of-light / speed)           set speed-x (speed-x * k)           set speed-y (speed-y * k)         ]         set v-x (speed-x / speed-of-light) ; expressed as a fraction of c         set v-y (speed-y / speed-of-light) ; expressed as a fraction of c         set v-squared (v-x ^ 2 + v-y ^ 2)         set lorenz-contraction (sqrt (1 - v-squared))         set acceleration-x (lorenz-contraction / electron-mass * ((1 - v-x * v-x) * force-x - v-                    x * v-y * force-y))         set acceleration-y (lorenz-contraction / electron-mass * ((- v-x * v-y) * force-x + (1 -          v-y * v-y) * force-y))         set speed-x (speed-of-light * v-x + tick-scale * acceleration-x)         set speed-y (speed-of-light * v-y + tick-scale * acceleration-y)         set speed (sqrt (speed-x ^ 2 + speed-y ^ 2))         set k (min list 1 0.99 * speed-of-light / speed)         set speed-x (speed-x * k)         set speed-y (speed-y * k)       ]     ]   ] end
to move-electron   let speed (sqrt (speed-x ^ 2 + speed-y ^ 2))         let k (min list 1 0.99 * speed-of-light / speed)         set speed-x (speed-x * k)         set speed-y (speed-y * k)   let speed-squared ((speed-x ^ 2) + (speed-y ^ 2))   let real-mass (1 / (sqrt (1 - speed-squared / (speed-of-light ^ 2))))   let acceleration-scale (tick-scale * electron-charge / real-mass)   let new-xcor 0   let new-ycor 0   set speed-x (speed-x + field-x * acceleration-scale)   set speed-y (speed-y + field-y * acceleration-scale)   set new-xcor (xcor + speed-x * speed-scale)   set new-ycor (ycor + speed-y * speed-scale)   ifelse (new-ycor > max-pycor) [     set new-ycor (2 * max-pycor - new-ycor)     set speed-y (- speed-y)   ]   [     if (new-ycor < min-pycor) [       set new-ycor (2 * min-pycor - new-ycor)       set speed-y (- speed-y)     ]   ]   ifelse ((new-xcor > max-pxcor - 2.5) or (new-xcor < min-pxcor)) [     die   ]   [     setxy new-xcor new-ycor     set heading (atan speed-x speed-y)   ] end   to check-collision   let collidees (cells in-radius (0.5 * cell-size / patch-scale))   if (any? collidees) [     let collidee (min-one-of collidees [distance myself])     let v-squared (speed-x ^ 2 + speed-y ^ 2)     let v (sqrt v-squared)     let relativistic-mass (electron-mass * speed-of-light / sqrt (speed-of-light ^ 2 - v-squared))     let delta-e (speed-of-light ^ 2 * (speed-of-light / (sqrt (speed-of-light ^ 2 - v-squared)) - 1))     let momentum-x (speed-x * relativistic-mass)     let momentum-y (speed-y * relativistic-mass)     ask collidee [       let current-cell-tempk (tempK)       set incremental-energy (incremental-energy + delta-e)       set speed-x (speed-x + momentum-x / cell-mass)       set speed-y (speed-y + momentum-y / cell-mass)       set tempK ((incremental-energy / (3.5 * 1E-9)) + (current-cell-tempk))       if (tempK >
```

```
638.15) [          let dead-new (dead + 1)          set dead (dead-new)          set-current-plot "deadcellsovertime"          plotxy
ticks dead          set-current-plot "Percentage of cells dead over time"          plotxy ticks (dead / cell-number)          ask
collidee [          die          ]          ]     ]          set collisions (collisions + 1)     die   ] end
to move-cell   let new-xcor (xcor + speed-x * speed-scale)   let new-ycor (ycor + speed-y * speed-scale)    ifelse (new-ycor >
max-pycor) [    set new-ycor (2 * max-pycor - new-ycor)     set speed-y (- speed-y)   ]   [     if (new-ycor < min-pycor) [      set
new-ycor (2 * min-pycor - new-ycor)        set speed-y (- speed-y)      ]   ]   ifelse (new-xcor > max-pxcor) [     set new-ycor (2 *
max-pxcor - new-ycor)     set speed-x (- speed-x)   ]   [     if (new-xcor < min-pxcor) [       set new-xcor (2 * min-pxcor - new-
xcor)       set speed-x (- speed-x)      ]   ]   setxy new-xcor new-ycor end
```

Appendix C: Works Cited

"ADENINE (6-AMINOPURINE)." *Chemicalland21.com*. Web. 07 Feb. 2010. <http://www.chemicalland21.com/lifescience/phar/adenine.htm>.

(particle accelerator) Breizman, B. N., P. Z. Cheboteav, A. M. Kudryavtsev, K. V. Lotov, and A. N. Skrinsky. *Self-Focused Particle Beam Drivers for Plasma Wakefield Accelerators*. Rep. Web. 11 Feb. 2010. http://peaches.ph.utexas.edu/ifs/ifsreports/Self-focused762.pdf.

(cytosine) "CYTOSINE (4-AMINO-2-HYDROXYPYRIMIDINE)." *Chemicalland21.com*. Web. 07 Feb. 2010. <http://chemicalland21.com/lifescience/phar/CYTOSINE.htm.>

Einstein, Albert. "Ist die TrägheiteinesKörpersvonseinemEnergieinhaltabhängig." *Annalen derPhysik* (1905). Print.

(to buy the Encyclopedia of Cancer) "Encyclopedia of Cancer." *Springer - International Publisher Science, Technology, Medicine*. Web. 11 Feb. 2010. <http://www.springer.com/biomed/cancer/book/978-3-540-36847-2>.

(principals of partical acceleration) "Field Precision LLC: download Principles of Charged Particle Acceleration." *Field Precision LLC: Finite-element Software for Electromagnetics*. Web. 11 Feb.

2010. <http://www.fieldp.com/cpa.html>.

(metastsis) "The FULL Cycle Of Cancer -- Normal Cell To Metastasis." *Oxygen - The Breath Of Life*. Web. 11 Feb. 2010. <http://www.oxygentimerelease.com/A/Disorders/p17.htm>.

(guanine) "GUANINE (2-AMINO-6-HYDROXYPURINE)." *Chemicalland21.com*. Web. 07 Feb. 2010. <http://www.chemicalland21.com/lifescience/phar/GUANINE.htm>.

Patel, S. G., and N. Kishore. "Thermodynamics of nucleic acid bases and nucleosides in water." *Journal of Solution Chemistry* 24.1 (1995): 23-58. Print.

(spacial relativity) "Theory: Special Relativity (SLACVVC)." *SLAC Public Website Server*. Web. 11 Feb. 2010. <http://www2.slac.stanford.edu/vvc/theory/relativity.html>.

(Thymine) "THYMINE (5-METHYLURACIL)." *Chemicalland21.com*. Web. 07 Feb. 2010. <http://www.chemicalland21.com/lifescience/phar/thymine.htm>.

"Understanding and Controlling Metastasis." *Suite101.com: Online Magazine and Writers' Network*. Web. 11 Feb. 2010. <http://www.suite101.com/article.cfm/new_cancer_treatments/94746/2>.

"URACIL (2,4-DIHYDROXYPYRIMIDINE)." *Chemicalland21.com*. Web. 11 Feb. 2010. <http://www.chemicalland21.com/lifescience/phar/uracil.htm>.