DNA Self-Assembly of Nanoelectronic Devices – The Nanodynamics Simulator

Ben Macadangdang
Electrical and Computer Engineering

Advisor:
Dr. Chris Dwyer

Submitted:
April 24, 2007

Background

The structure of nucleic acids lends itself to polymerization of multiple nucleic acids together to form a long strand of DNA. Coupled with the well known complementary base pairings of those nucleic acids, DNA is often found in a double helix. These base pairings utilize hydrogen bonds to keep the two strands of DNA close, with Adenine (A) almost always binding with Thymine (T) and Cytosine (C) almost always binding with Guanine (G). The AT and GC bonding gives DNA a predictable behavior when in solution and allows for design of single stranded DNA to form more complex structures, as proposed by Seeman as early as 1982.[1] These structures make use of the self-assembly process whereby single strands of DNA are mixed together in solution and the solution is heated to break any hybridizations and then cooled to allow the single strands of DNA to hybridize back together. By controlling the sequence of those single strands of DNA and thus the complementary base parings, only the desired structure is formed in large amounts (see Fig. 1).

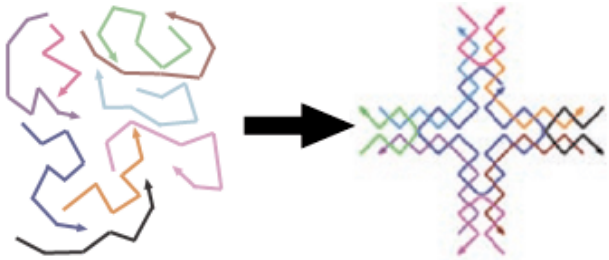


Figure 1 – The DNA self-assembly process. These nine strands of single stranded DNA are designed such that the only way complementary base pairs can come together is when the tile structure is formed.

In Figure 1, the nine strands of single stranded DNA form a tile object when subjected to the self-assembly process. Of the nine strands of single stranded DNA, one of the strands is called the core strand (dark blue strand in the center), four of the strands are called the shell strands (orange, brown, purple, and light blue strands) and four are called the arm strands (black, pink, green, and magenta strands). However, the "tile arm" is the set of DNA strands that protrude up, down, left, and right from the cruciform tile structure in Figure 1 and will henceforth just be referred to as the arm.

The tiles can then be extended to form more complex structures by connecting several tiles with each other. This is accomplished through the use of sticky ends, which are unpaired bases at the ends of the arms. Because each arm is essentially made of two double strands of DNA side by side, each arm has two sticky ends. Figure 2 below shows the sticky ends of the arms.

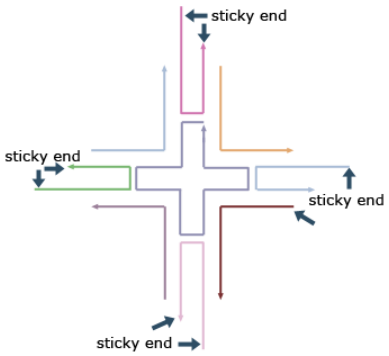


Figure 2 – The sticky ends of the tile arms. Each arm has two single strands of DNA at the ends that are used to hybridize with another set of two sticky ends, thereby connecting two tiles together.

The sequence of the sticky ends can then be designed such that the sticky ends will hybridize together. This will bring two tiles next to each other, forming a larger structure. Again the self-assembly procedure is employed, where the individual tiles are mixed, heated, and cooled together in solution and the only way the sticky ends can all hybridize together is to form the desired structure (see Figure 3). Complex scaffolds have been created using this strategy including two dimensional repeated patterns of grids[2], fully addressable patterns of grids[3], and three-dimensional polyhedra.[4]

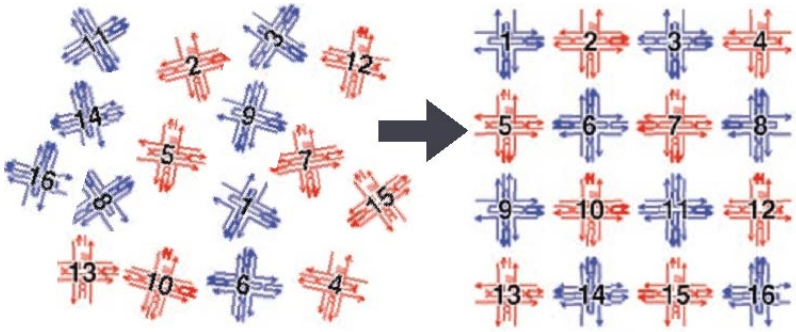


Figure 3 – Grid formation. The self-assembly process is used with tiles to form stable grids, such as the 4x4 grid shown here.

Tile formation usually results in a good yield of tiles due to the high release of Gibbs free energy when the long single strands hybridize. However, the grid formation process has not yet been perfected and does not produce as good of a yield. There are a couple of reasons for this. First, the sticky ends are only five base pairs long and therefore their hybridization is not as favorable thermodynamically. Second, the sticky ends are now constrained to the orientation of the tile and cannot rotate as freely as a single strand of DNA can.

The goal of this project is to capture the formation of the grids in order to better understand their behavior in solution. Different tools were explored in order to accomplish this task; however no current tools were found to do so. The use of molecular dynamic simulators was discarded because they give too much detail and get inaccurate after a certain period of time has elapsed. Higher level simulators did not give enough detail. Therefore, an intermediate level simulator was written in order to capture grid formation by modeling tile objects diffusing around in

solution. The simulator is written in C with a visualization portion written in C++ and utilizing OpenGL. The goal of the simulator is to not only capture the grid formation, but to learn about the kinetics and thermodynamics of that process. It can use that information to predict future behavior of the grid formation, hopefully to possibly extending grids to a more useful structure, such as a three dimensional structure.

Diffusion

When tiles are put together in solution, no external forces ever act on them to cause them to move. Therefore, the driving force behind the diffusion of the tiles is Brownian motion, which is the random movement of particles through collisions with the solvent and is named after Robert Brown, a botanist who observed this phenomenon in 1827.[5] Even though this movement is random, it is still a form of diffusion and thus over long periods of time, the average displacement of the particles can be mathematically quantified.

This mathematical relationship was first described by Einstein in 1905 in his doctorate thesis concerning Brownian motion.[6] In his paper, Einstein demonstrates that a small spherical particle will diffuse around in solution according to the following equation:

$$D = \frac{k_B T}{f} \tag{1}$$

where D is the diffusion constant, k_B is Boltzmann's constant, T is temperature, and f is the frictional coefficient. Since k_B and T can be thought of as constants for a given solution, the only difference between how different shaped particles will diffuse is due to the frictional coefficient, which is dependent on the shape of the particle. Specifically Einstein dealt with spherical particles, so the frictional coefficient can be given by Stoke's Law to be $6\pi\eta r$, where η is the viscosity of the fluid and r is the radius of the sphere.[7] This leads to the well known Stokes-Einstein relation of:

$$D = \frac{k_B T}{6\pi\eta r} \tag{2}$$

Several papers have now verified the Stokes-Einstein relation and have applied it to other systems.[8,9] The diffusion coefficient can then be used to predict the displacement of the particle after a certain period of time. This relationship is given by:

$$< \vec{R}_k > = 2kDt \tag{3}$$

where R is the root mean square displacement, k is the dimension, D is the diffusion coefficient, and t is time.[10] It should be noted that, for a general particle, there is often a translational diffusion coefficient as well as a rotational diffusion coefficient.

Verification of Spherical Diffusion

The diffusion of spheres is a simplified process due to the symmetry of the spheres. This means that the translational coefficient is the same for the sphere in all directions and also that the rotational diffusion coefficient does not matter. In order to simulate the movement of spheres through the solution, a Gaussian random function is used. In one dimension, the probability density function for the displacement, x, can be given by

$$P(x) = \sqrt{\frac{1}{2\pi\sigma^2}} e^{-x^2/2\sigma^2} \tag{4}$$

where $\sigma = \sqrt{2D\tau}$ and τ is the time step between the measurements of position and D is the diffusion coefficient for the sphere.[10] (Equation 4 can be applied to any type of particle, not just spheres as long as the correct diffusion coefficient is used.) This means that to simulate a step that a sphere takes, a Gaussian random number is picked that has a mean of 0 and a variance of $2D\tau$ for each of the three mutually orthogonal directions in space. The three Gaussian numbers are then combined together to form a three dimensional vector and that vector represents the displacement of the sphere for that time step.

In order to verify that spheres could be successfully simulated, a setup was made similar to the setup found in Nakroshis *et al.* paper, *Measuring Boltzmann's Constant Using Video Microscopy of Brownian Motion*.[10] Polystyrene spheres of radius 0.51 μm where stepped in two dimensions for 30 μs and the results were plotted on a graph of time and displacement. Equations 2 and 3 were combined to get

$$< \vec{R}_2 > = \frac{4k_B T\tau}{6\pi\eta r} \tag{5}$$

Solving for Boltzmann's constant, equation 5 becomes

$$k_B = \frac{6\pi\eta r}{4T} * \frac{< \vec{R}_2 >}{\tau} = \frac{6\pi\eta r}{4T} s \tag{6}$$

where s is the slope of the graph. Therefore, the slope of the displacement vs. time graph should be proportional to Boltzmann's constant if the spheres are being simulated correctly. Figure 4 shows the results of the simulations.

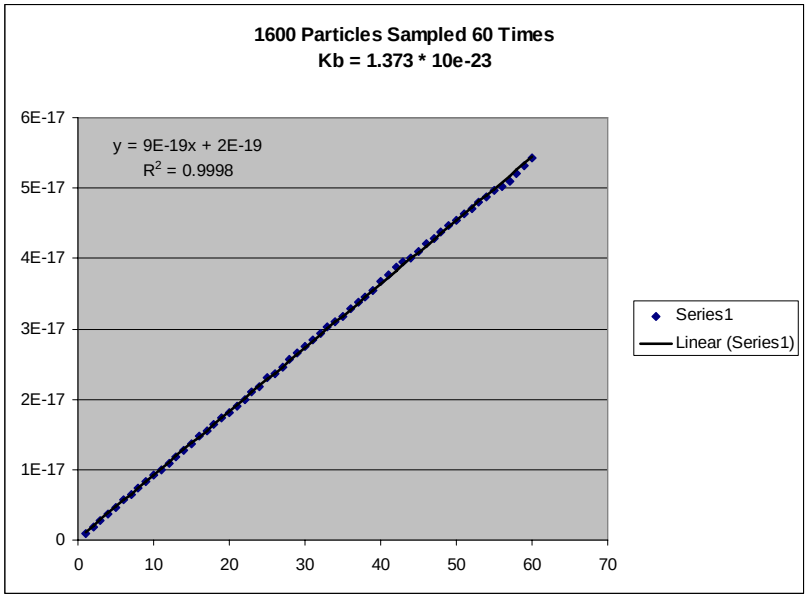


Figure 4 – 1600 spheres were simulated and stepped for 30μs. The slope of the displacement vs. time graph is proportional to a factor of Boltzmann's constant. Here a value of $1.373 * 10^{-23}$ was found when the real value is $1.381 * 10^{-23}$

The percent error of this measurement was approximately 0.5%. It was therefore concluded that spheres could be successfully simulated in two dimensional space.

Diffusion of DNA

The diffusion of DNA is slightly more complicated than the diffusion of spherical particles. Tirado *et al.* have done a lot of theoretical work concerning the diffusion of DNA in solution and have come up with some interesting results.[11,12,13] They have found that DNA will diffuse around as if it were a right circular cylinder, which makes sense since the DNA can be circumscribed by a cylinder. However, what is not intuitive at first is that there are actually two translational diffusion coefficients, one for movement parallel to its long axis and one for movement perpendicular to its axis.[11,13] These diffusion coefficients are given by

$$D_{\parallel} = \frac{(\ln p + v_{\parallel})k_B T}{2\pi\eta L} \tag{7}$$

$$D_{\perp} = \frac{(\ln p + v_{\perp})k_B T}{4\pi\eta L} \tag{8}$$

where p is the ratio of length to diameter, L is the length, and v is the end effect correction. Because there is a different diffusion coefficient along the parallel and perpendicular axes of the cylinder, the orientation of the cylinder needs to considered. The rotational diffusion coefficient of a cylinder was also found by Tirado *et al.*[12,13] Its equation is given by

$$D_r = \frac{(\ln p + v_r)3k_B T}{\pi\eta L^3} \tag{9}$$

In order to simulate the translation of the DNA, the Gaussian random function given by Equation 4 is used again, with the mean still 0 and $\sigma = \sqrt{2D\tau}$, but this time with the correct diffusion coefficient for the parallel and perpendicular axes. The orientation of the DNA is stored so that the parallel axis and two mutually orthogonal perpendicular axes are known at all times. Then one Gaussian is picked for the parallel axis and two are picked for each of the perpendicular axes. The composite vector of all three mutually orthogonal vectors gives the translational step.

It is not intuitive what the rotational diffusion constant represents since it is dimensionless. However, Hijazi and Zoaeter show that the angle of rotation about a particular axis has a variance of $2D_r\tau$, just like the translational counterpart.[14] Since the cylinder is assumed to be radially symmetric, rotation around the parallel axis does not affect the DNA. Therefore, only two Gaussians are chosen to represent the rotation about each of the axes orthogonal to the parallel axis. The cylinder is then rotated to its new orientation.

Quaternions and Rotations in Three Dimensional Space

The most common way to represent an orientation in three dimensional space is to use rotations about three orthogonal axes of a reference frame, such as a universal X, Y, and Z axis. This method of spatial orientation is called Euler angles. The advantages of using Euler angles is that the representation is easy to use and visualize and involves a well documented rotation matrix that is easy to find. The major drawback of Euler angles is that they suffer from a condition called Gimbal lock.[15] During Gimbal lock, two of the three axes are aligned together, causing one of the axes to be cancelled, and thus a loss of information.

A solution to this problem is the use of a four-vector called a quaternion, first discovered by William Hamilton in 1843.[16,17] In a quaternion, a scalar and three complex numbers representing three orthogonal imaginary axes are stored. The mathematics of quaternions is different from real number mathematics, including a non-commutative multiplication.[18] Rotations involving quaternions can be represented in the following way:

$$\mathbf{q} = [\cos(\frac{\theta}{2}), v_x \sin(\frac{\theta}{2}), v_y \sin(\frac{\theta}{2}), v_z(\frac{\theta}{2})] \tag{10}$$

where θ is the angle of rotation, v is the axis of rotation, and |v|=1. Therefore any axis can be chosen and any angle can be chosen to be rotated about that axis, and the quaternion can encompass it. This is called an axis-angle rotation. If the quaternion, **q**, is represented in the notation of $\mathbf{q} = [q_0, \mathbf{v}]$, where **v** is the imaginary vector, then the complement of **q**, denoted by $\bar{\mathbf{q}}$, is $\bar{\mathbf{q}} = [q_0, -\mathbf{v}]$.[18]

Rotation about an arbitrary point, (x,y,z), is easily found if first a quaternion is composed using the point in such a way as $\mathbf{p} = [0, x, y, z]$. Then, to do an axis-angle rotation, simply perform the following calculation

$$\mathbf{p_{new}} = [0, x_{new}, y_{new}, z_{new}] = \mathbf{q} \cdot \mathbf{p} \cdot \bar{\mathbf{q}} \tag{11}$$

where the quaternion multiplication is used and (x_{new}, y_{new}, z_{new}) represents the coordinates of the newly rotated point.

The use of quaternions is extremely useful to calculate rotations of DNA in solution because the rotational diffusion coefficient helps to determine the random Gaussian angle to rotate about the two orthogonal axes. The calculations of quaternions are also very fast and involve less overall computations than the rotational matrix multiplication.

Verification of Correct Cylinder Stepping

Just like the movement of the sphere needed to be verified, the stepping of the cylinders also needed to be verified, especially since two different translational diffusion coefficients were used, as well as a rotational diffusion coefficient. This time, the cylinder was stepped 1,000,000 times with a time step of 8.8 ps ($8.8*10^{-12}$ seconds) and the displacement was compared to the overall theoretical displacement provided in a formula by Tirado *et al.*[11,13] Figure 5 shows the results of this simulation.

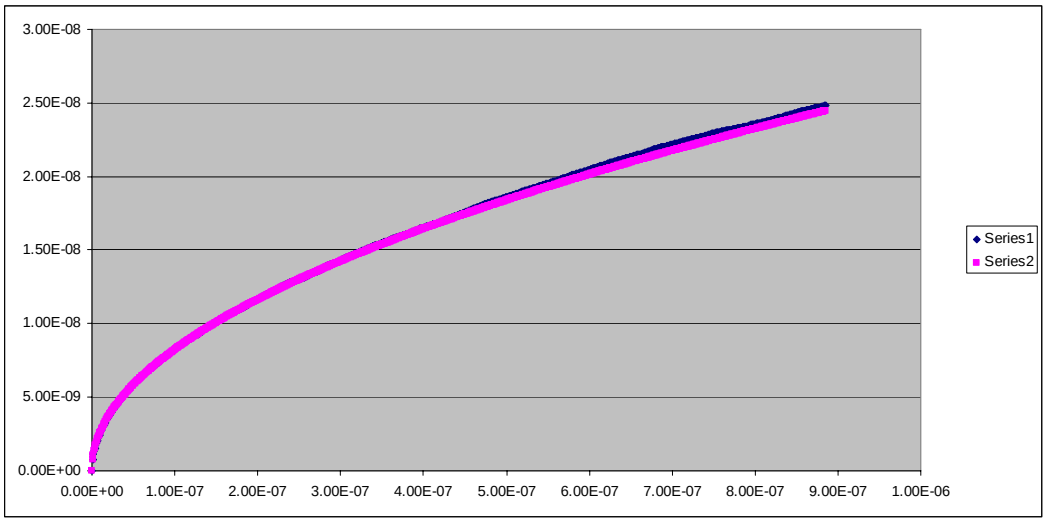


Figure 5 – Simulation of a cylinder being stepped 1,000,000 times with a time step of $8.8*10^{-12}$ seconds. The blue line is the theoretical displacement and the pink line is the actual displacement.

From Figure 5, it is easy to see that the theoretical displacement is almost perfectly paralleled by the simulated displacement. Equation 3 was used to find the theoretical displacement with the correct diffusion coefficient provided by Tirado *et al.*[11,13]

Structural Modeling and Simulator Objects

The shape of the tile needs to be represented in a three dimensional structure that has to take into account a balance between being physically accurate and the ability to be put through collision and intersection tests easily. For example, since DNA diffuses around solution as if it were a right circular cylinder, it would be physically accurate to model a strand of DNA as that. However, testing for the intersection of right circular cylinders is computational expensive because it involves the solution to a differential equation.[19] On the other hand, computations for the intersection of spheres is extremely fast, but a sphere does not physically represent a strand

of DNA, so it would be worthless to even test for a collision in the first place. Therefore, to find a mixture of the two solutions, a capsule model was introduced to model a double strand of DNA. A capsule is defined as all the points equidistant from a finite line segment, as illustrated in Figure 6.

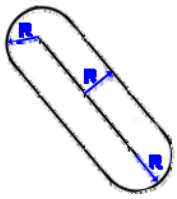


Figure 6 – A capsule. All the points are the same distance away from a finite line segment.

To test for an intersection of a capsule, the distance between two line segments was extended.[20] This distance algorithm is an O(1) algorithm and not very computationally expensive. Once the distance between the two line segments is known, it is a simple calculation to see if the capsules intersect by checking if that distance is less than the sum of the two radii of the capsules. This method of capsule intersection was tested thoroughly both visually and by checking boundary conditions.

Since the sticky ends of the arms are single stranded DNA, the capsule model was not used. Instead, a Lorentz cone, or second order cone, was used, which is shaped like an ice cream cone. This model was chosen because it represents the movement of the single strand of DNA at the end of the arm. Since one end is tied to the end of the arm, it is not free to rotate. However, the other end is free to move around but because the sticky end is only five nucleotides long, it is relatively stiff. Therefore, the shape that it sweeps when moving in three dimensional space is a Lorentz cone, as illustrated in Figure 7.

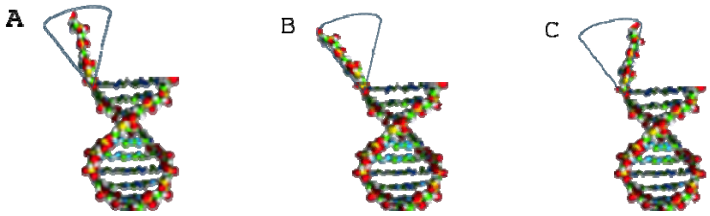


Figure 7 – The Lorentz cone. The double stranded DNA represents an end of half an arm (since an arm is made of two double strands of DNA). The single stranded DNA is the sticky end, which has a constrained rotation. The shape the it fills in three dimensional space is the Lorentz cone.

The tile is then made by putting together eight capsules and eight cones as shown in Figure 8.

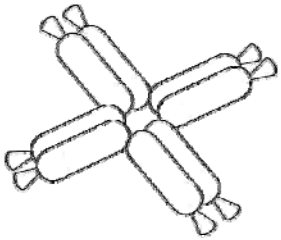


Figure 8 – The tile model made of eight capsules and eight cones.

Each of the capsules is kept close to the adjacent capsules in what is known as a constraint. The constraint gives a maximum and minimum distance that two objects can get from each other.

However, an object within the simulator is not constrained to just tiles. A tile is just one of many objects that can be represented in the simulator. A "simulator object" can be made of any number of four primitive objects. These four primitive objects are capsules, cones, slabs, and spheres, as illustrated in Figure 9.

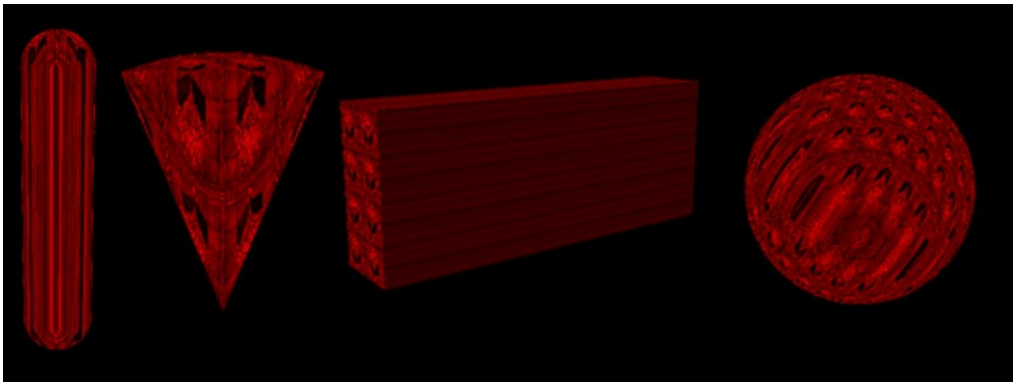


Figure 9 – The four primitive shapes in the simulator objects. From left: Capsule, Cone, Slab, and Sphere.

It just so happens that the tile object only utilizes two of the four possible objects – the capsule and cone. However, it can easily be imagined that these objects can be extended to other systems, such as biological systems. For example, both a phospholipid and a transmembrane protein can be modeled using different numbers of spheres and capsules, as shown in Figure 10.

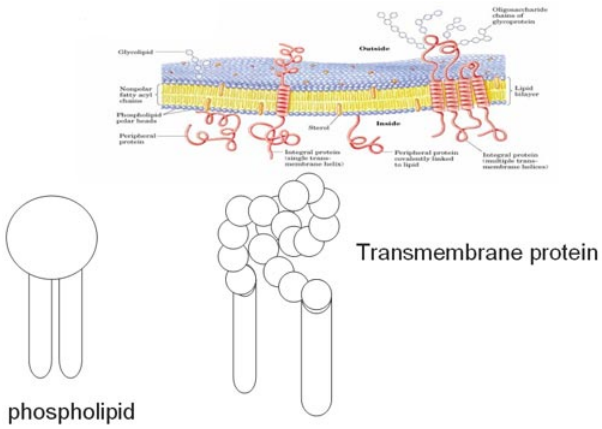


Figure 10 – Two examples of biological structures that can be modeled in the simulator using only the primitive objects. Here a phospholipid and a transmembrane protein use different numbers of capsules and spheres.

It is common to model very flexible structures using spheres like the transmembrane protein does.

The Constraint Model and Validation of it

It was a concern that the constraint model listed above for the tiles will prevent the tile from diffusing around in solution like it should in real life. Therefore, the constraint model was verified through two different tests.

In the first test, four capsules were tethered together as shown in Figure 11. These smaller capsules should then diffuse around like a larger capsule with the same surface area. A time step value of 8.859 picoseconds was used and the constraints were picked so that the maximum distance that the objects could get away from each other was 10 times the distance an individual capsule could move in one time step.

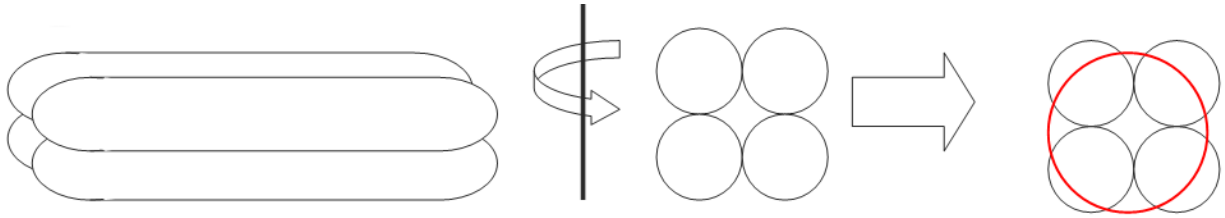


Figure 11 – Four tethered capsules. The four small black capsules should diffuse in solution like the big red capsule does, given that they have the same surface area.

The results of the simulation are shown in Figure 12 where 400 tethered capsule objects were simulated and each small capsule was stepped 1,000,000 times with diffusion coefficients of the small capsule.

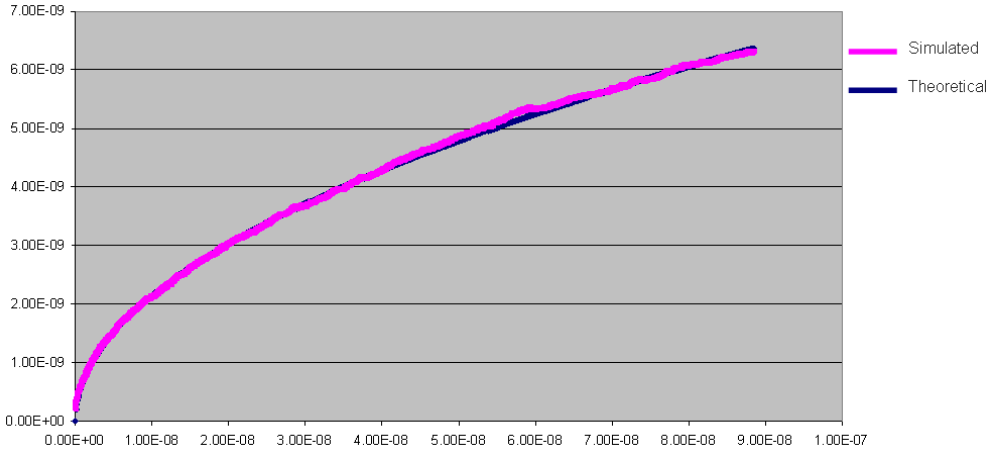


Figure 12 – The four tethered capsule simulations.

However, it should be noted that the simulated distances were multiplied by a constant factor of 1.64 in order to get them to line up directly with the theoretical diffusion distance. Therefore, it was hypothesized that there is a correspondence between the maximum constraint distance and the time step chosen.

Therefore, as second constraint test was setup. This time, tiny spheres were positioned into rings and stacked on top of each other in order to model a cylinder. Rings of six spheres per ring, 12

spheres per ring, and 24 spheres per ring were simulated as shown in Figure 13. Because to total length of the cylinder needed to stay the same, and when there are less spheres per ring, the radius of each sphere is bigger, there are different numbers of rings for each system.

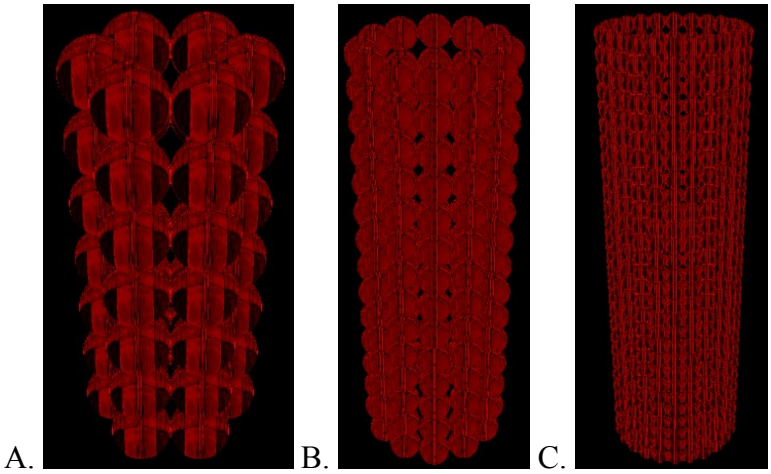


Figure 13 – A) Six spheres per ring, 7 rings. B) 12 spheres per ring, 14 rings. C) 24 spheres per ring, 28 rings

The spheres would diffuse with diffusion coefficients of a sphere, but would also be constrained. A matrix of time step values to maximum constraint values were simulated, as shown in Figure 14.

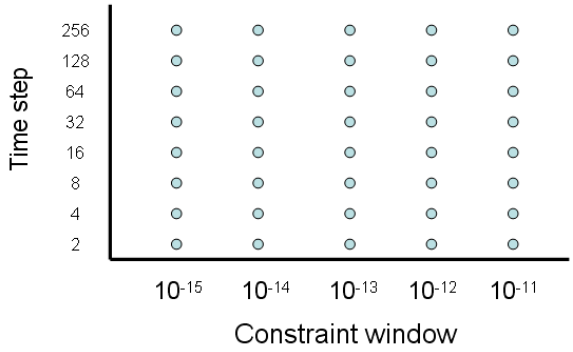


Figure 14 – The matrix of time step values and constraint window values used in the simulations.

The constraint window was defined as the maximum constraint minus the minimum constraint. These systems were stepped 100,000 times with 400 cylinder objects. For brevity, the results will not be shown. The conclusion made was that there is a definite correspondence between the constraint window and the time step, but that values for each can be chosen that give the correct theoretical values for a system. Therefore, the constraint system is a valid system to model multiple objects diffusing in solution together.

Diffusion Limited Aggregation

The theoretical model behind the formation of grids in the simulator is called diffusion limited aggregation (DLA). The DLA model was developed in 1981 by Witten and Sander.[21] The

original application of the DLA model was to demonstrate how tiny metal particle aggregation can be simulated by a computer.[21,22,23] In the model, spheres are assumed to bind to each other irreversibly when they come into contact. There are four steps to the model:

1. Start with a seed particle in the center of the environment
2. At some random distance, R, from the seed, introduce another particle
3. Randomly walk this particle. There are two situations possible:
    3a. If this particle moves into an adjacent space to an existing particle, stop the first particle's movement and go to step 2.
    3b. If the particle gets to a distance, R', from the center particle where R' > R, then remove the particle and go to step 2.
4. Repeat this process for a certain number of particles added or a certain amount of time.

The results of the simulations of metal particles look like Figure 15.

Figure 15 – The results from Witten and Sander.[21]

The final product is very wispy and was in good agreement from experimental observations as evidenced by similar fractal dimensions.[22] This model of aggregation was applied to other dilute systems where the limiting restraint on formation was the contact of the particles.[22] Therefore, this model seems to fit well with grid formation because tiles are in a dilute solution.

Modified DLA and Sticky End Interaction

The DLA model was modified slightly for the purposes of the nanodynamics simulator. First, whole tiles are used instead of spherical particles. Second the tiles no longer stick irreversibly with each other and third, they do not automatically bind when coming into contact. Instead, the interactions of the sticky ends are taken into consideration. These interactions depend on three variables – the temperature of the solution, the sequences of the sticky ends, and the amount of volume overlap of the cones.

The temperature of the solution needs to be taken into consideration because double stranded DNA has a critical temperature called a melting temperature, where 50% of the DNA is single stranded and 50% is double stranded.[24] This is due to the fact that double stranded DNA is kept

together by relatively weak hydrogen bonding when compared to covalent bonding. A theoretical melting curve for DNA is shown in Figure 16.

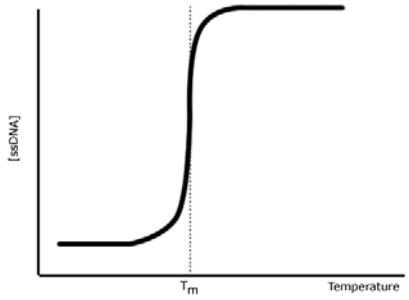


Figure 16 – The melting temperature, T_m, is the temperature at which 50% of the DNA is single stranded and 50% is double stranded. The curve is a very sharp curve.

Any temperature above the melting temperature and most of the DNA in solution will be single stranded and any temperature below the melting temperature, most of the DNA will be double stranded. This is due to the sharpness of the curve. The melting temperature is also dependent of the sequence of the DNA. As pointed out earlier, G will bind with C through three hydrogen bonds and A will bind with T through two hydrogen bonds. Therefore, the more GC content a sequence has, the stronger its interaction and the higher the melting temperature will be. This melting temperature can be calculated by a nearest neighbor interaction, as outlined by Santa Lucia.[25] Therefore, the sequence and temperature of the solution are closely related and will influence how well the sticky ends will interact with each other.

The third influence in the sticky end interaction is the volume of overlap of the cones. The more the cones overlap each other, the stronger their interaction should be because more of the DNA of one of the sticky ends has a chance to hybridize with the sticky end. The intersection volume of two Lorentz cones is not a closed form solution and therefore a unique algorithm was invented to find this volume using a binary search.

The binary search algorithm first found the two points corresponding to the greatest overlap between the cones. This was done by first finding the distance, D, between two points on the long axis of the cone as illustrated in Figure 17-A. Next, the distance from one point to the edge of its cone was measured as D1 and the same was done for the other point and stored as D2 as shown in Figure 17-B. Then if the quantity D – (D1 + D2) was negative, then the cones overlapped and that value was stored. Then the binary search was used to pick four more points, two on each axis, as illustrated by Figure 17-C. The same method was used to find the distances and if the quantity D – (D1 + D2) was more negative than previously stored, it was stored, along with the points on the axis.

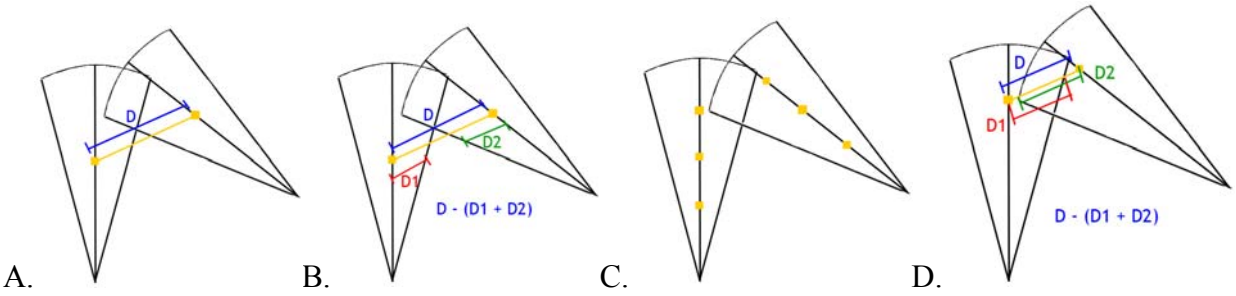


Figure 17 – The binary search algorithm for finding the maximum overlap of cones for intersection volume.

Then once the two points of most overlap was found, the two points on the cone edges were found as well as the midpoint between those two points, as illustrated by Figure 18-A. Then a box was found by extending out from the midpoint in two mutually orthogonal directions illustrated by the blue line and a line pointing in and out of the page in Figure 18-B. These lines gave the dimensions of the box and the volume of the box was used to estimate the intersection volume.

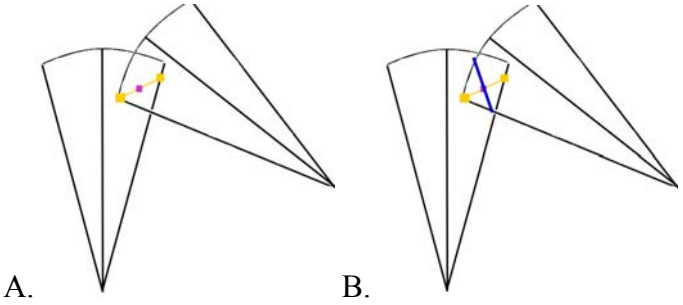


Figure 18 – Finding the overlap box

To verify that the accuracy of the binary search algorithm, the volumes obtained by the binary search were compared to a brute force parser. The brute force parser is able to take a point in space and figure out if that point is within a cone or not. It will do this for sequential points in space. For instance, if the step size is 0.1, then it will first check the point (0,0,0), then (0.1, 0, 0), then (0.2, 0, 0), etc. When it reaches the maximum x value, it will move in the y direction and check (0, 0.1, 0), then (0.1, 0.1, 0), etc until the entire 3D space if filled. If a point is within both cones, then the volume of the box created by the step size is added to the total volume of the intersection. For example, if the step size was 0.1, and point was found within both cones, then $0.1^3 = 0.001$ was added to the intersection volume. It was found that the volumes found by the brute force parser and the binary search method differed by less than 1% of the normalized volume of the cones.

Arrhenius Equation

In order to quantify the three different variables for the sticky end interaction, the Arrhenius equation is used. The Arrhenius equation is given by Equation 12:

$$k = Ae^{\frac{-\Delta E}{RT}} \tag{12}$$

where ΔE is given by

$$\Delta E = \frac{V_{new} - V_{old}}{V_{\max}} \Delta G \tag{13}$$

Substituting Equation 13 into equation 12, the Arrhenius equation becomes

$$k = Ae^{\frac{-(V_{new} - V_{old})}{RTV_{\max}} \Delta G} \tag{14}$$

where A is the reciprocal of the area of the Arrhenius integral of V_{new} from 0 to V_{max} for a givenV_{old}. Integrating the Arrhenius equation gives the probability distribution function, which can then be used in a coin flip. If the tile tries to make a step that changes the intersection volume of the cones, then the new intersection volume, V_{new} is plugged into the distribution function to get a number between 0 and 1. Then a uniformly distributed random number is chosen between 0 and 1 and if the random number is less than the Arrhenius value, the step by the tile is not taken and the tile returns to its original position. Otherwise the step is taken. Therefore, when the sequences are complementary, the temperature is low enough, and the volume overlap is close to max, the distribution function should give a number close to one, meaning there is a high probability that a step is not taken. This makes sense theoretically, since when those conditions are satisfied, the sticky ends are at their optimum hybridization.

Optimization of Collision Detection

Collision detection is simply checking whether given the current positions of two objects, if they are colliding. Most of the collision detection algorithms between the various primitive shapes in the simulator have been modified by the algorithms given by David Eberly.[26] However, there is a way to optimize collision detection by smartly picking which objects to check with the current object being stepped. For instance, it is not efficient to move a sphere by stepping it and then check to see if it has collided with each of the objects in the environment, especially if there are lots of objects.

Looking at Figure 19 below, if just the x-axis is viewed, then there are two overlaps – green & red and blue & red. Looking at the y-axis, there are four overlaps – green & blue, green & red, green & black, and red & black. However, when both the x and y axes are combined, the only two colors that overlap on both axes are green & red, and therefore those two shapes are colliding.

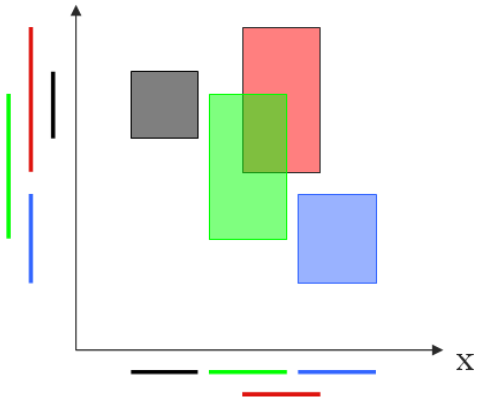


Figure 19 – The only objects the overlap in 2D are the red and green. Source from http://www.cs.wisc.edu/graphics/Courses/638-f2001/lectures/cs638-23.ppt.

The efficiency of the algorithm is seen when three parallel arrays are kept, one for each dimension – x, y, and z. Those arrays are then sorted by each objects one dimensional position. For example, the x array sorts all the objects based on their minimum x coordinate, the y array sorts all the objects based on their minimum y coordinate, and the z array sorts for all objects based on their minimum z coordinate. Also stored is each of the objects' maximum coordinate in each dimension, giving that object a window. Then when an object is stepped, a bubble sort is used to sort the arrays by the minimum value in each dimension again. The bubble sort is extremely efficient for nearly sorted arrays, which is the case here since stepping one object will not result in a lot of reshuffling of positions. Then the window of the stepped object is searched in each of the three arrays to see if it overlaps with any window of another object. If another object overlaps the stepped object's window in all three arrays, those two objects are checked for a collision. This method eliminates checking for collisions of objects that are really far away.

Future Work

Currently the simulations validating the constraint model are still being finalized using the computer cluster owned by the Computer Science Department at Duke. The next step of work needs to be done to look at the sticky end interactions. Preliminary simulations have shown that the maximum intersection of two Lorentz cones does not line them up head to head, but rather at a 135 degree angle from each other, as shown in Figure 20.

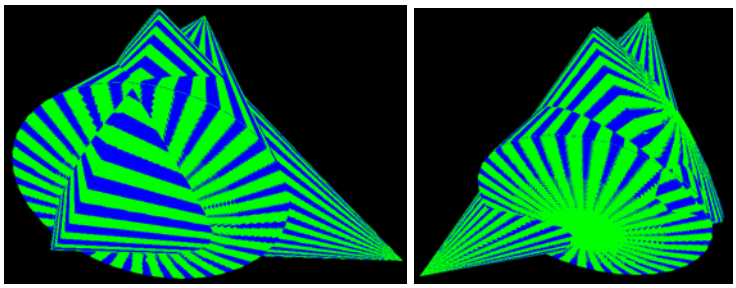


Figure 20 – Two Lorentz cones intersecting maximally

A new model as been proposed that sets the tip of the cone well within the capsule as shown in Figure 21.

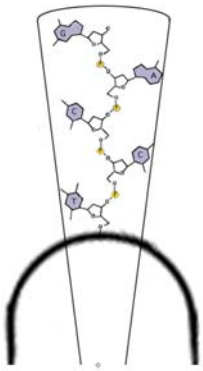
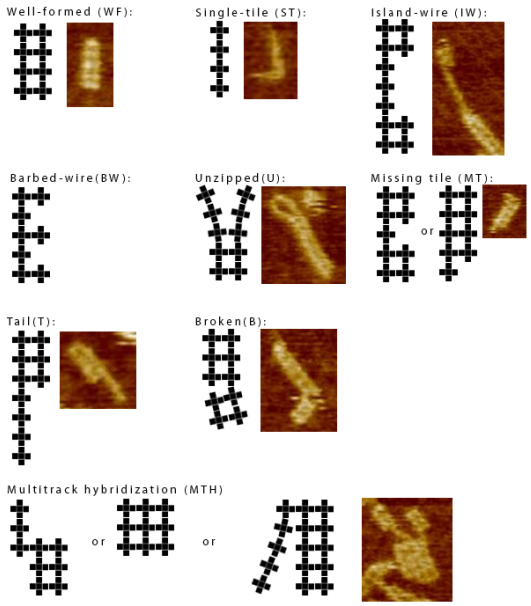


Figure 21 – A new model for the sticky ends.

Work needs to be done to investigate whether this new proposed method will produce maximum intersection volumes for the cones that line them up head to head, like they should be.

Following the sticky end interactions, simulations can begin using the tiles and the DLA model. A result analyzer needs to be written that can easily interpret the results of each simulation and classify the grids that are formed. These results will then be compared to the experimental work done last spring. In the experimental work, a group of four tiles was designed so that they would create long tracks, two tiles wide, called nanotracks. The results of the nanotrack experiments were classified qualitatively and quantitatively. Figure 22 shows the qualitative classifications and Figure 23 shows an image from the nanotrack experiment using the AFM.



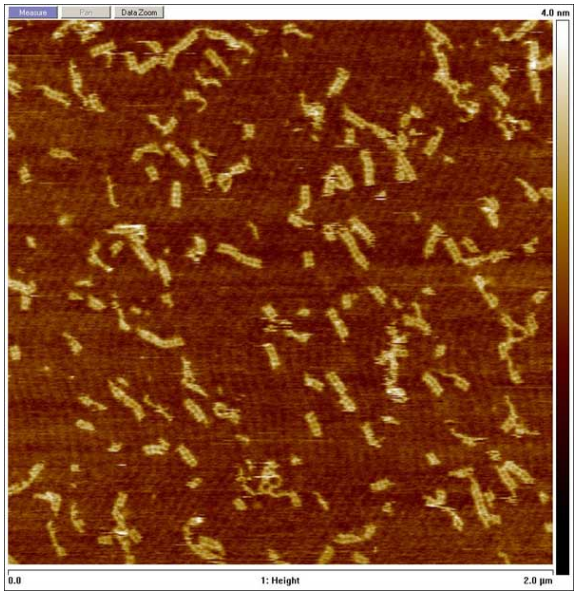


Figure 22 – Different classifications of the nanotracks.

Figure 23 – An AFM image of the nanotracks.

The results of the simulations will be compared to the results of the nanotrack experiments, using the same parameters in the simulation as was done in the experiment. This would be a verification of the simulator as a whole. Parameters would be tweaked if the results were slightly different in order to get more accurate results.

The simulator would then be used to create more interesting structures such as three dimensional grids and structures that are more useful. If a combination of tiles was found that gave good results in the simulator, they could then be tested in real life.

References

1. Seeman, Nadrian. 1982. Nucleic Acid Junctions and Lattices. *Journal of Theoretical Biology*. 99: 237-247.
2. Park, Sung Ha, *et al.* 2005. Programmable DNA Self-Assemblies for Nanoscale Organization of Ligands and Proteins. *Nano Letters*. 5(4): 729-733.
3. Park, Sung Ha, *et al.* 2005. Finite-Size, Fully Addressable DNA Tile Lattices Formed by Hierarchial Assembly Procedures. *Angewandte Chemie*. 45(5): 735-739.
4. Chen, Junghuei and N.C. Seeman. 1991. Synthesis from DNA of a molecule with the connectivity of a cube. *Nature*. 350: 631-633.
5. Brown, Robert. 1866. The miscellaneous botanical works of Robert Brown: Volume 1.
6. Einstein, Albert. 1906. A New Determination of Molecular Dimensions. *Annalen der Physik*. 4(19): 289-306.
7. http://www.britannica.com/eb/article-9069781/Stokess-law. Accessed: April 12, 2007.
8. Kivelson, Daniel, S.J. Knak Jensen, and M. Ahn. 1972. Molecular theory of translational Stokes-Einstein relation. *The Journal of Chemical Physics*. 58(2): 428-433.
9. Liu, Bin and J. Goree. 2006. Test of the Stokes-Einstein Relation in a Two-Dimensional Yukawa Liquid. *Physical Review Letters*. 96: 1-4
10. Nakroshis, Paul, *et al.* 2002. Measuring Boltzmann's constant using video microscopy of Brownian motion. *American Association of Physics Teachers*. 71(6): 568-573.
11. Tirado, Maria M. and J.G. de la Torre. 1979. Translational friction coefficients of rigid, symmetric top macromolecules. Application to circular cylinders. *Journal of Chemical Physics*. 71(6): 2581-2587.
12. Tirado, Maria M. and J.G. de la Torre. 1980. Rotational dynamics of rigid, symmetric top macromolecules. Application to circular cylinders. *Journal of Chemical Physics.* 73(4): 1986-1993.
13. Tirado Maria M., C.L. Martinez, and J.G. de la Torre. 1984. Comparison of theories for the translational and rotational diffusion coefficients of rod-like macromolecules. Application to short DNA fragments. *Journal of Chemical Physics.* 81(5): 2047-2052.
14. Hijazi, A. and M. Zoaeter. 2002. Brownian dynamics simulations for rod-like particles in dilute flowing solution. *European Polymer Journal*. 38: 2207-2211.
15. Jones, Eric M. and P. Fjeld. Gimbal Angles, Gimbal Lock, and a Fourth Gimbal for Christmas. Nasa. http://www.hq.nasa.gov/alsj/gimbals.html. Accessed: April 12, 2007.
16. Hamilton, William R. 1844. On a new species of imaginary quantities connected with a theory of quaternions. *Proceedings of the Royal Irish Academy*. 2: 424-434.
17. Hamilton, William R. 1847. On Quaternions. *Proceedings of the Royal Irish Academy.* 3: 1-16.
18. Karney, Charles FF. 2006. Quaternions in molecular modeling. *Journal of Molecular Graphics and Modelling*. 25: 595-604.
19. Eberly, David. 2000. Intersection of Cylinders. *Geometric Tools, Inc.*
20. Eberly, David. 1999. Distance Between Two Line Segments in 3D. *Geometric Tools, Inc.*
21. Witten, T.A. and Sander, L.M. 1981. Diffusion-Limited Aggregation, a Kinetic Critical Phenomenon. *Physical Review Letters*. 47(19): 1400-1403.
22. Witten, T.A. and L.M. Sander. 1983. Diffusion-Limited Aggregation. *Physical Review B.* 27: 5686-5697.

23. Meakin, Paul. 1983. Formation of Fractal Clusters and Networks by Irreversible Diffusion-Limited Aggregation. 51(13): 1119-1122.
24. Breslauer, Kenneth J., *et al.* 2007. Predicting DNA Duplex Stability from the Base Sequence. *Proceedings of the National Academy of the Sciences of the United States of America*. 83: 3746-3750.
25. SantaLucia, John. 1998. A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics. *Proceedings of the National Academy of the Sciences of the United States of America*. 95: 1460-1465.
26. Eberly, David. Geometric Tools. http://www.geometrictools.com/Documentation/Documentation.html. Accessed: April 22, 2007.
27. Chenney, Stephen. 2001. CS638: Computer Game Technology. http://www.cs.wisc.edu/graphics/Courses/638-f2001/lectures/cs638-23.ppt. Accessed: April 22, 2007.