EG UK Theory and Practice of Computer Graphics (2010)
John Collomosse, Ian Grimstead (Editors)

Computer Graphics Education and the understanding of pixel plotting algorithms using Growth Aggregation models

Jonathan C. Roberts

School of Computer Science, Bangor University, UK

Abstract

It is sometimes difficult to teach fundamental aspects of computer graphics, especially pixel plotting techniques, as some students fail to engage with the material. In this paper we describe a constructionist approach to help students learn about fundamental computer graphics techniques. By getting the students to develop code that performs a growth aggregation model, principally using Diffusion Limited Aggregation techniques, reflect upon that code and make a critical analysis of their own work in a report we hope the students will learn the material. An evaluation of two years of students' work, their results and various indicators suggest that this approach has been successful and the students engaged with the material better.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation – Viewing algorithms

1. Introduction

Engaging with students to help them learn about various fundamental graphics principles such as rasterization, scan conversion, pixel filling (flood-filling) algorithms and other pixel based algorithms can be difficult. Often in a computer graphics class the students have a wide range of skills, abilities and interests. While some students implicitly engage with the material and some are merely keen to learn, others disengage with the course because they come with preconceived ideas such as they want the course to teach them how to re-create high level complex graphics scenes found in games and misunderstand the reason and usefulness of studying the fundamental aspects, or merely they may be put off by the mathematics of computer graphics.

We deliver a course on the "Fundamentals of Computer Graphics", which is compulsory for our second year Computer Science students. The course covers the fundamental concepts of plotting pixels. The course includes Bresenham's line drawing algorithm, scan conversion, anti-aliasing, pixel filling algorithms, shading models and basic rendering (where the principles of ray-tracing and z-buffer are covered). The students then take, in their third year, an Advanced Computer Graphics module.

Over the past two years we have being using a constructionist approach, by getting the students to develop a Diffusion Limited Aggregation simulation. We have several objectives for this approach.

- The students gain a deep understanding of the material by actively constructing an *artifact* (the DLA program). The students learn about the challenges of plotting pixels through a process of designing, making and reflecting.
- The students engage better with the course and also develop a deeper understanding of the material through making an exciting artifact.
- They will gain a broad understanding of some general computer science principles, such as coherence, speed/size and complexity of calculations.

This paper describes our approach. We briefly explain the background to the course and then describe the simple aggregation models that are explained to the students and some related work (section 2). Several different aggregation models are used in various fields of science and art, and we make reference to some of these related fields in this section. Section 3 presents the tasks that the students are asked to perform. In section 4 we present their results and explain what models they implemented, and describe what they wrote about in their report (and what they included in their critical evaluation). Finally, we discuss our analysis of their perfor-

DOI: 10.2312/LocalChapterEvents/TPCG/TPCG10/115-122

mance and some learning *indicators* (section 5) and make our conclusions (section 6).

2. Background & Related Work

Our second year computer graphics course includes four main areas of computer graphics: (1) graphics libraries, (2) rasterization, scan conversion, and pixel plotting algorithms (3) transformations and (4) surfaces and materials. OpenGL through JOGL is taught to the students, which they use to perform the aggregation model assessment. In this computer graphics module the students also complete another assessment, which is to develop a graph paper application that displays lines (pixelated by a rasterization algorithm, such as Bresenham's line drawing algorithm) and sine, cosine and tangent curves.

It is for the pixel plotting algorithms (flood and scan-line filling algorithms), rasterization and scan conversion part of the course that we specifically employ a constructionism approach. We follow the learning-by-making idea of Papert and Harel [PH91]. Such that by giving the students a task to perform, where they generate interesting pictures, they are enthused to do the assessment and learn some appropriate computer graphics principles on the way. For instance, by developing and thinking about aggregation models they should consider aggregation, neighbourhood, proximity and digitization issues, and general computer science concerns of space versus time along with speed and performance issues.

Structures in nature may (such as trees or lichens) on a first glance seem complex to model. However, there are several simple algorithms that can be used to simulate these complex forms and generate intricate results. In nature branches or leaves grow in order to to maximize the surface area and minimize the distance from the branches. This process thus balances the need of food production with the distance that the food travels [Her86]. In computer modelling many of these techniques use the principal of aggregation by *sticky pixels* [Rob01]. One or more seed pixels are placed in the environment. Subsequent pixels are introduced to the environment and move around the space; when they touch another pixel they stick together to form a larger cluster and an aggregation is formed. The pixels are said to be sticky as they stick to a pixel that is adjacent to them. The word pixel is used here as commonly the particles are pixels, but they could be larger than a pixel, and their size could change over time. An alternative way is to use grammar-based (L-grammar) models such as demonstrated by the trees of Briggs [JB98] or the plants by Měch and Prusinkiewicz [MP96]. But we focused on the point based aggregation models for our computer graphics course.

Various parameters of the model can be changed, which adapt the form of the aggregation. These include: (1) the behavior of the particles. How they enter the environment and move around. Whether the particles attract or repel other particles. (2) The container size, shape and area, and whether there is anything else in the container, such as liquid, other seed points. (3) Any external feature that may affect the particles. Such as heat, movement of the environment or gravity.

By changing several of these parameters scientists and artists have created different models that simulate different natural phenomena. These models are often intricate and self repeating (fractal patterns). Such techniques have generated images of plants [MP96], trees [PHL*09], leaves [RFL*05], lightening [EJM00], lichen [DGA04] and ice formation [KHL04]. Sanchez et al. investigate how close branching patterns in nature are to various aggregation simulations [SZC*03]. A few researchers have applied the concepts to non-natural structures. For instance, Chen and Lobo use structure of DLA to layout co-citation data [CL03]. Halsey writes "Recent insights from this well-studied model have led to many new applications from river networks to oil recovery, and from electrodeposition to string theory" [Hal00].

In our computer graphics class we explained three principal aggregation models (Eden, DLA and Ballistic) and also taught information about pixel adjacency and arrangement, which we briefly explain below.

2.1. Eden's model

Eden's model was published in 1961 [Ede61] and is thus one of the earliest computer growth models to be published [Her86]. In this model a seed point is started in the center of the grid with subsequent pixels being added randomly to bounding points. The cluster is compact and solid.

2.2. Diffusion Limited Aggregation

The traditional Diffusion Limited Aggregation model was originally developed by Witten and Sander [WS81]. The concept of diffusion is found in many chemical processes. Diffusion occurs as the particles in a solution move around before they aggregate together into a structure. The models are said to be 'diffusion-limited' because the position of the seed pixel is fixed to the center of the grid, it cannot move and the pixels stick together one at a time. First, a seed is set in the center of the lattice then the next particle is introduced far from the origin of the aggregation at a random point. This new particle moves at random (Brownian motion) and stops when it touches the initial seed or the aggregation.

It can take a long time for the point to randomly walk to the aggregate, and it is possible that the point may never stick to the aggregate. Consequently, a killing circle is often used to remove particles that are wandering away from the main aggregation. Alternatively, it is possible to allow the particle to move larger distances when it is further away from the main aggregate, and shorter distances when closer

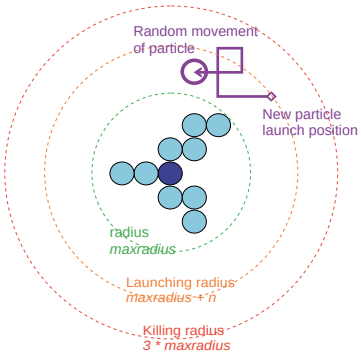


Figure 1: *Diffusion Limited Aggregation, after Witten and Sander [WS81]. Particles are added to the experiment far away and randomly move until they are adjacent to the current seed or aggregate.*

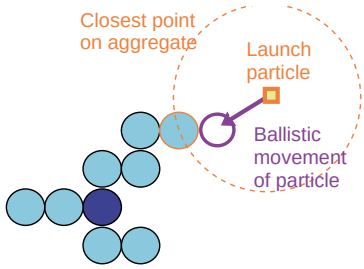


Figure 2: *Randrom Drop Ballistic Aggregation (RDBA), a ballistic aggregation approach that fires the new elements into the scene.*

to the aggregation. The speed of these models is something the students were asked to think about – especially in relation to the lattice, the movement of the pixels, the decision function that decides whether a particle sticks to a current position, and also storage of the aggregation.

Numerous adaptations can be made to this traditional algorithm that change the form of the aggregate. Methods include: having multiple starting points (a line of points generates a diffusion limited deposition model [Mea83]), changing the stickiness of the particle (i.e., the probability that a particle will aggregate), some particles may stimulate growth while others could destroy or infect part of the cluster, or trails could be left behind as the particle moves. Other variants of the algorithm include cluster-cluster methods where hundreds of particles start in a suspension, as they diffuse they form clusters when they collide, and subsequent clusters move and then form larger aggregates. Other models add repulsive or attractive energy such that their kinetics, mass and mass distribution are utilized [LLW*89].

2.3. Ballistic Aggregation

With ballistic aggregation the particles travel along straight lines. They are added to the aggregate whenever they touch the particle in the cluster. In 2001 we introduced a variant named Random Drop Ballistic Aggregation (RDBA) [Rob01] where the particles are dropped randomly and stick to the closest point in the current aggregate. This produces aggregation models that have longer thinner tendrils. Various dropping strategies could be used, such as dropping over a circle, making the subsequent pixels be biased by a random offset from the previous, or dropping using a mask to effect where the pixel is placed [Rob01].

2.4. Arrangement, Adjacency and Calculations

One important aspect of these models is to consider how they are arranged on the grid, and importantly to decide how they are considered connected. This was very relevant for the students of our class, because similar concepts of arrangement and adjacency are required in order to understand scan conversion, pixel filling (flood-filling) algorithms and other pixel based algorithms.

The Arrangement of how the data is stored on a grid is significant because by using different lattices the structure of the result can be altered. Commonly a square homogeneous lattice arrangement is used to hold the pixels. Thus the data may be readily stored and quickly accessed from an array structure. This also matches well with the other pixel algorithms in computer graphics that store the pixels in a regular lattice – such as a frame buffer. However, a regular grid creates an anisotropic result of the DLA structure [Bog02]. Other lattices may be hexagonal or triangular [Mea86] or unusual lattices such as the Penrose Lattice [LLT*94].

Adjacency is another fundamental part of the aggregation models, which is also important for the students to understand in this computer graphics module. In fact, defining what is considered to be connected as a neighborhood affects the formed structure. In two dimensions, four connectivity allows the edges of the squares to be adjacent, whereas 8-connectivity (a Moore neighborhood) adds the diagonals. Adjacency is obviously associated with the lattice arrangement, for instance 6-connectivity is possible on an hexagonal lattice and equally valid (but differently applied) on a three dimensional cube. In three dimensions 6-connected represents faces, 18-connected adds adjacency along edges, with 26-connected cubes include the corners.

In two dimensions, given a pixel p and a point (x,y) the 4-connected adjacency neighborhood is given by

$$N_4(p) = \{(x+1,y),(x-1,y),(x,y+1),(x,y-1)\}$$

The 8-connected neighborhood is given by

$$N_8(p) = N_4(p) \cup \{(x+1,y+1),(x-1,y-1),(x-1,y+1),(x+1,y-1)\}$$

When a series of pixels are each adjacent and 4-connected

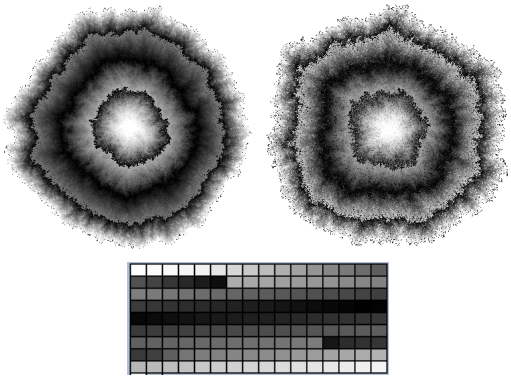


Figure 3: *By changing the adjacency criteria different aggregation forms can occur. Growth aggregation using Eden's model. Left, using 4-connectivity adjacency rule; Right, using 8-connectivity. Below, colourmap for the visualization, showing bands of white, grey and black.*

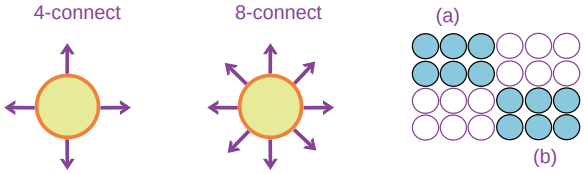


Figure 4: *Depending on whether the pixel area on the right is considered to be 4-connected or 8-connected the area of dark pixels is connected or separated into two regions.*

together they form a path from one to the other. Not only does connectedness affect the pixel form of growth aggregation models (as shown in Figure 3) but it is significant in pixel plotting algorithms such as flood-filling methods. For instance, the two dark regions (a) and (b) of pixels in Figure 4 are not 4-connected but are 8-connected. Thus depending on the connectivity of the boundary defined region the areas may be connected together or considered to be separate.

When considering connectedness and adjacency it is suitable to also consider the probability in which a particle will stick. Furthermore, by changing from 4-connect to 8-connect we are changing the state of the grid that defines whether a particle sticks in its current position or does not. Thus for example, cellular automaton can be used to change the outcome from step to step or a sticking coefficient could be used to determine the probability of that particle aggregating.

Finally, there are three different stratagies for storing and calculating the points. *Full-lattice* explains that the pixels are held and calculated on integer sized grids. *Semi-lattice*, enables calculations to be done in floating-point arithmetic and rounded into integers to be stored in an integer based lattice, similar in idea to the scan-conversion of a line, say. *Off-lattice*, both calculates the particles and stores their positions in floating point numbers.

3. The task

The task for the students was to develop a Java program that demonstrated a Growth Aggregation model. They were to use the Java Bindings to OpenGL to display the aggregation and suggested that they should consider their display as a petri dish experiment. They were also encouraged to think about pixels, their positions on a raster screen, and what connectivity and adjacency means.

They were given the following instructions and asked to provide the following functionality:

1. Your program will use an appropriate Java data-structure: Your Java program should have an appropriate data-structure to hold the points (e.g. it could be an array) and it should be clear whether you are using 4 or 8 connectivity. There are obviously different ways to implement this model, so you should think and plan of the best way to do it. Have a look at the papers that are placed on the course management system, as well as the slides on Growth Models and your notes from the lectures. Once the program is run it will allow numerous particles to move around and aggregate onto the (initial) seed position and/or any subsequent aggregated points. You may like to give the user the option of designating how many points will be added.
2. Your program will use JOGL to display the results. It should start off with a blank screen, with at least one seed point in the display, and should clearly show new particles appearing and aggregating. Think how you would do a re-draw event. Think how you may allow the window to be resized.
3. Your program should appear colourful. You should allocate a colour for each of the pixels. E.g., early points have a lighter colour, while later points are given a darker colour. You may like to think how a user could change the colour or appearance of the aggregated points.
4. Your program will include additional functionality (at least two extra features): There are many different features that could be added. Suggestions include: use of 3d, different sized particles, killing circles, zooming features in/out, different types of objects (e.g. circles, squares, hexagons), load/save model or save as a bitmap image, different attractors (circle, lines, areas etc), use of different connectedness (switch between 4-connection or 8-connection), displayed statistics e.g. number of particles currently or the centre of mass, radius of killing circle.
5. Your program will have an accompanying report The report (in Word or pdf format) will detail: (i) A description of what you have implemented and highlighting some key

features to your code. (ii) A *critical analysis* of what you have done/achieved. This is important, and should discuss any limitations with your code and how you could overcome those limitations. (iii) Detail of what additional functionality (task 4) you have added. (iv) At least, one screen shot of your results.

4. Results of their Work

We have run this assessment for two years. In 2009 we had 21 students make submissions, while in 2010 28 students submitted code. Each year the students get excited over seeing their programs develop interesting results, and some students even say that they *tinker* with the program after their submission.

Each student submitted a Java/JOGL program, with most of them using two classes: one to hold the main application class to initialize the program and create any buttons, and the other class as an event-listener class that is used to perform the drawing and re-scaling of the image. Some of the better solutions also added a `Point` class to hold details of the location and colour of each pixel.

Figure 10 includes some screen-shots from their solutions. The students added many features, choosing between models, switching models during mid processing, changing the size of the launching and killing circles, being able to zoom in and out, saving the results as an image.

Most students presented reports that detailed what they had achieved. Importantly a majority of the students presented a critical analysis of their work. They reported where their program was not fully functional or generating erroneous points. They also reported their wishes, such as wishing to include a zoom function but were not able to achieve it in the time available. Some of the students discussed the challenges of generating a zoom function that allowed each of the points to scale appropriately, while others discussed the nature of 4-connected over 8-connected models. Most of the students discussed the speed of their program, and some had obviously thought hard how to speed the program up as they explained the data-structure that they use. Overall their reports were comprehensive and demonstrated a very good understanding of the problem.

5. Analysis and Learning Indicators

We base our judgment using three methods. First, we analyzed the features of their tools, keeping the same order as in the task (section 3). Second, we looked at their reports and their critical analysis, and finally we present some results from a brief questionnaire that they answered.

It was encouraging to see that every student submitted a program with a basic DLA structure Figure 5 ; most used an array to hold the data (and many also discussed the limitations of doing so in their report), some of the better answers used linked lists or other dynamic container classes. In analyzing their answers for 2009 and 2010 respectively; 76% and 86% submitted a program that developed a correct aggregation method. However, 36% and 46% submitted a program with errors. Some of these errors were minor and functional (such as only walking in diagonal directions that created constrained diffusion aggregations) while other students provided code that would not compile.

Part 2 and Part 3 of the task involved considering how the information is updated. It is interesting that the students in 2010 created better answers to this part of the task, as seen in Figure 6. More students provided an interactive update where the points appear as they are calculated, other students animated the actual particle randomly walking. There may be several reasons for this; the students were better, we taught the material more clearly, or they collaborated and discussed the problem better together. It is our belief that the 2010 cohort gelled better together and they discussed the problem together. The laboratories that we gave were better attended in 2010 and the students asked many more questions than in 2009. Furthermore, it is clear the that students in 2010 incorporated many more features in their programs than the students in 2009, Figure 7. DLA was the most popular methodology to implement, followed by EDEN's model and RDBA, see Figure 8. One student each year also developed a particle suspension model where several particles move in the simulation at the same time.

A short questionnaire was given to the students.

a Explain whether you enjoyed this assessment?
b What did you learn from this exercise?

On a scale 1..5. (1) Strongly disagree; (2) Disagree, (3) Neither agree nor disagree, (4) Agree, and (5) Strongly agree.

c I understand rasterization and pixel plotting concepts better after this assessment.
d I wished to achieve a nice DLA design; it enthused me to do well at this assessment.
e I would recommend this assessment to be repeated for following years.

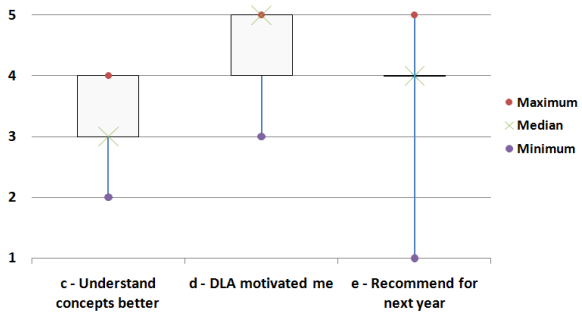


Figure 9: *Analysis of the questionnaire.*

In answer to the question about 'enjoyment' most of the

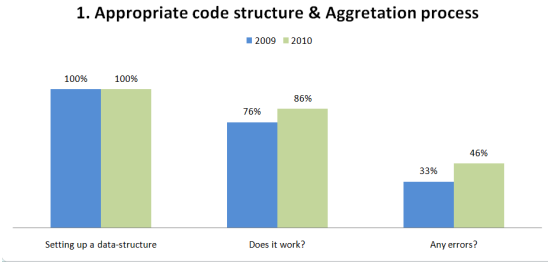


Figure 5: *Part 1: Your program will use an appropriate Java data-structure,*

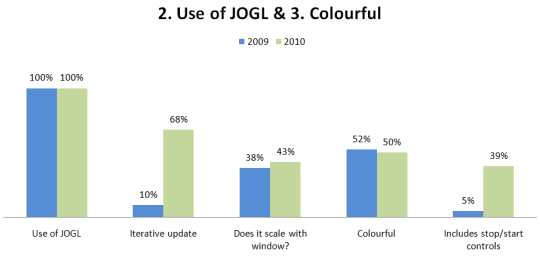


Figure 6: *Part 2 and 3: Your program will use JOGL.*

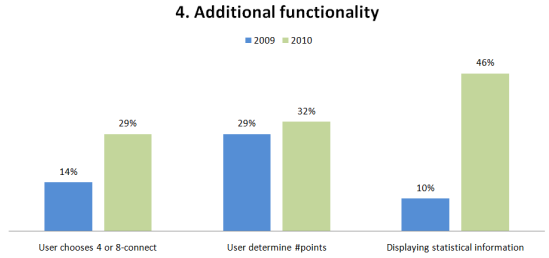


Figure 7: *Part 4: Additional functionality.*

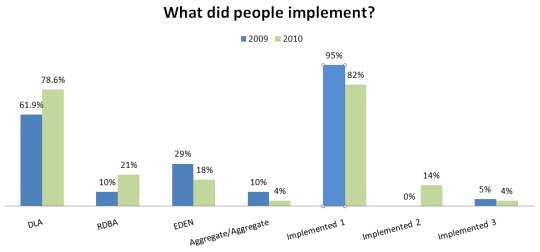


Figure 8: *Most students implemented one algorithm, which was a traditional DLA model. However, some of the better answers included several algorithms.*

students who answered the questionnaire said that they enjoyed the exercise. Quoting three different students: "I did enjoy the assignment, it was a more practical application of graphics than just rendering already defined things. Instead we were creating a mathematical model ourselves and thinking about how to display the output". "I enjoyed the DLA assignment once I got my head round it, I found it interesting researching and coding as I'd never even heard of the term DLA or seen anything on it before." "I quite enjoyed this assignment it was definitely more interesting than the average Java problems we get given, and it was also applicable to something other than the average business type problems we have to solve in the computing lab module". However, not every student enjoyed the experience or saw its relevance, one said "I unfortunately really disliked the DLA assignment and for a number of reasons did not complete it. I found (even though obviously there are lecture slides associated) that it was disconnected with the lectures and the labs".

For the question about what they 'learned', the students all agreed they learned about JOGL, JOGL interaction with GUI's and listeners, JOGL syntax and methods and types of aggregation. One student writes "I learned a whole bunch of JOGL stuff not to mention graphics strategies for speeding up a program and why mine was running slow etc.".

The results from the quantitative part of the questionnaire are shown in Figure 9. It demonstrates that most of the students who replied agreed that they had learned about the pixel plotting concepts through the assessment, and that the DLA design enthused them to do better, and that they would recommend the assessment to be repeated for the following years.

6. Discussion & Conclusions

The idea of using a constructionist approach for learning is not new; but it is a useful strategy to take especially for a computer graphics course. It is very easy for academics to think up ways to assess students, ways that are often looked at as being serious, but it is harder to find assessments that enable all the students (from whatever background) to engage with the material. We believe our use of the aggregation simulation was successful for this course. The majority of the students produced working code that modeled an aggregation successfully, some students went further than was required and added extra functionality, while some said that they would continue adding to their program after they had handed in their assessment.

It was explicit in the assessment that the students needed to learn how to operate JOGL effectively, such that the particles could be displayed and that they could animate and move. But it also required that the students thought about the placement, adjacency and movement of the pixels. Most students considered how this can be achieved on a full-lattice, while some students used a semi-lattice. By creating an artifact and then critiquing their work the students were forced to analyze what they had done, and think further of their

methodology. Even if the students had not been able to make the code perform correctly, they could reflect on their work and thus learn the concepts.

It is also suitable to consider that this assessment encouraged other good Computer Science methodologies, such that the students followed an Agile approach to their work. Resnick [Res03] writes about a suitable constructionist approach, the TREE strategy "Test Randomly, Evaluate, Elect (choose which direction to move)". We believe many of the students were following such a strategy because – along-side the lectures we ran a series of laboratory classes on JOGL – during these classes the students showed us different versions of their code, and asked for advice on their code.

We acknowledge that there are weaknesses to our study. For instance, the cohort sizes were small, not all the students provided answers to the questionnaires and it would have been interesting to analyze the student's work from previous years – who had not done this exercise. However, we have received several positive comments from the students saying that they had enjoyed the assessment and had learned from the experience and that we should repeat the assessment next year. In fact, even one of the students who was quite negative said "I must say I find it refreshing that a lecturer is doing this and I hope my feedback is not too harsh and helps you with a future approach."

We believe that we achieved our goals that (1) we have demonstrated that the students did gain a deep understanding of the material by actively constructing an *artifact* (the DLA program) and that the students learned about the challenges of plotting pixels through a process of designing, making and reflecting. (2) The students did seem to be better engaged with the course. (3) Their reports certainly demonstrated that they had thought about other aspects of computing such as speed and extensibility issues.

In conclusion, the constructionist approach worked well in this instance. We encourage other academics to create exciting assessments that engage the students through developing an artifact.

References

[Bog02] BOGOYAVLENSKIY V. A.: How to grow isotropic on-lattice diffusion-limited aggregates. *J.of Phys A: Mathematical and General 35*, 11 (2002), 2533.

[CL03] CHEN C., LOBO N.: Semantically modified diffusion limited aggregation for visualizing large-scale networks. In *Proc. 7th Conf. on Info.Visualisation, IV* (2003), pp. 576–581.

[DGA04] DESBENOIT B., GALIN E., AKKOUCHE S.: Simulating and modeling lichen growth. *Computer Graphics Forum 23*, 3 (2004), 341–350.

[Ede61] EDEN M.: A Two-dimensional Growth Process. In *Proc. 4th Berk. Symp. Math. Statist. and Prob.,* (1961), no. 4, pp. 223–239.

[EJM00] EVANS B., JONES H., MALLINDER H.: Lightning strikes: an installation. In *Proc. 18th Eurographics UK Conf.* (2000), pp. 9–16.

[Hal00] HALSEY T.: Diffusion-limited aggregation: A model for pattern formation. *Phys. Today 53*, 11 (2000), 36–41.

[Her86] HERRMANN H. J.: Growth: an introduction. In *On Growth and Form: Fractal and non-fractal Patterns inPhysics*, Stanley H., Ostrowsky N., (Eds.). NATO ASI Series, 1986, pp. 3–20.

[JB98] JONES H., BRIGGS P.: Modelling the growth process of trees. In *Eurographics UK Conference Proceedings* (March 1998), pp. 97–106.

[KHL04] KIM T., HENSON M., LIN M. C.: A hybrid algorithm for modeling ice formation. In *SCA '04: Proc. ACM SIGGRAPH/Eurographics Symp. on Comp. anim.* (2004), pp. 305–314.

[LLT*94] LIU Z., LIU Y., TIAN D., XIA H., JIANG Q.: Diffusion-limited aggregation on penrose lattices. *Journal of Physics: Condensed Matter 6*, 22 (1994), 4037.

[LLW*89] LIN M. Y., LINDSAY H. M., WEITZ D. A., BALL R. C., KLEIN R., MEAKIN P.: Universality of fractal aggregates as probed by lightscattering. In *Fractals in the Natural Sciences*, M.Fleischmann, D.J.Tildesley, R.C.Ball, (Eds.). Princeton Univ. Press, 1989, pp. 71–87.

[Mea83] MEAKIN P.: Diffusion-controlled deposition on fibers and surfaces. *Phys. Rev. A 27*, 5 (May 1983), 2616–2623.

[Mea86] MEAKIN P.: Universality, nonuniversality, and the effects of anisotropy on diffusion-limited aggregation. *Phys. Rev. A 33*, 5 (1986), 3371–3382.

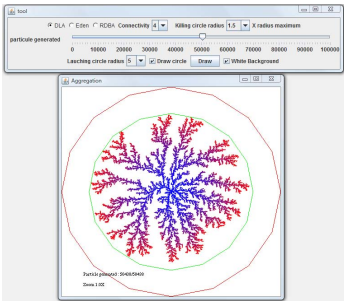
[MP96] MĚCH R., PRUSINKIEWICZ P.: Visual models of plants interacting with their environment. *Computer Graphics 30*, Annual Conference Series (1996), 397–410.

[PH91] PAPERT S., HAREL I.: *Situating Constructionism, Chapter 1 of Constructionism*. Ablex Publishing Corporation, 1991.

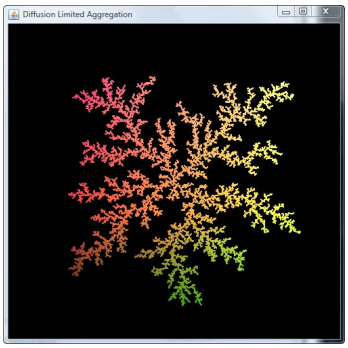
[PHL*09] PALUBICKI W., HOREL K., LONGAY S., RUNIONS A., LANE B., MĚCH R., PRUSINKIEWICZ P.: Self-organizing tree models for image synthesis. *ACM Trans. Graph. 28*, 3 (2009), 1–10.

[Res03] RESNICK M.: Thinking like a tree (and other forms of ecological thinking). *International Journal of Computers for Mathematical Learning 8* (2003), 43–62.

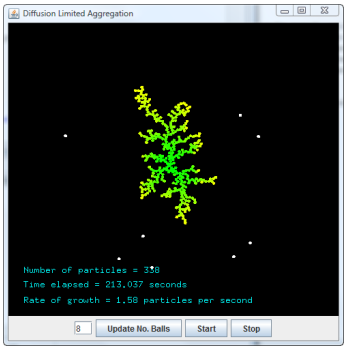
[RFL*05] RUNIONS A., FUHRER M., LANE B., FEDERL P., ROLLAND-LAGAN A.-G., PRUSINKIEWICZ P.: Modeling and visualization of leaf venation patterns. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), ACM, pp. 702–711.



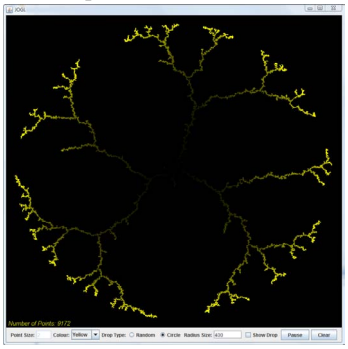
DLA model using a 4-connect model, showing the kill circle and comprehensive interface.



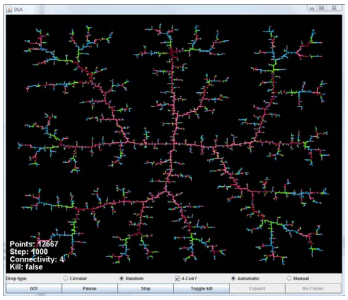
DLA model using an 8-connect model.



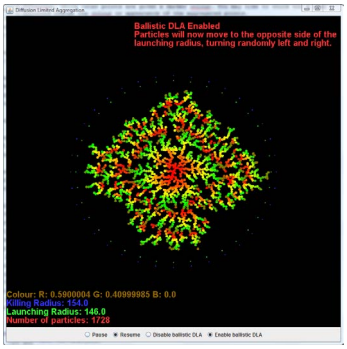
DLA model using an 8-connect model and tracing multiple particles.



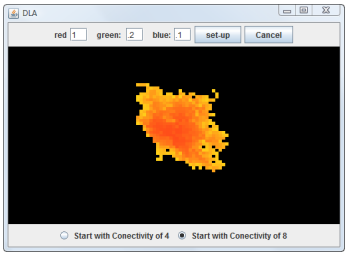
RDBA model with the particles being launched from a circle.



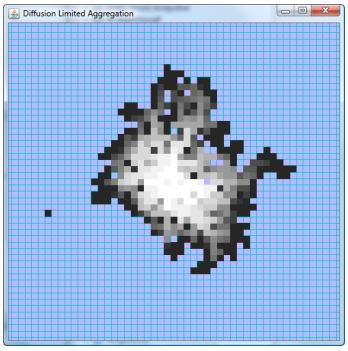
RDBA model with 4-connect pixels, with particles dropped anywhere in the area. Statistical information is also shown.



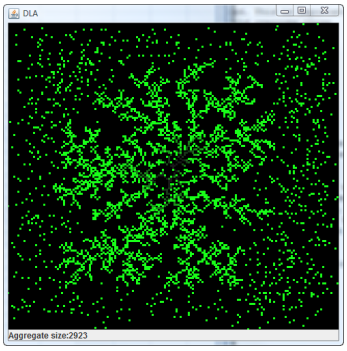
An animated RDBA model where the particles are seen to move.



An 8-connect Eden model.



A simple black and white Eden's model implementation.



An diffusion model with multiple particles moving and aggregating at the same time.

Figure 10: *Examples of students' work*

[Rob01] ROBERTS J. C.: Sticky pixels: Evolutionary growth by random drop ballistic aggregation. In *Proc. of Eurographics UK 2001* (2001), pp. 149–155.

[SZC*03] SANCHEZ J. A., ZENG W., COLUCI V. R., SIMPSON C., LASKER H. R.: How similar are branching networks in nature? a view from the ocean: Caribbean gorgonian corals. *J. Theoretical Bio. 222*, 1 (2003), 135 – 138.

[WS81] WITTEN T. A., SANDER L. M.: Diffusion-limited aggregation, a kinetic critical phenomenon. *Phys. Rev. Lett. 47* (1981), 1400–1403.