Extending the Growing Point Language to Self-Organise Patterns in Three Dimensions

Alyssa Morgan
University of the West Indies (Mona)
Kingston, Jamaica
amorgan.edu@gmail.com

Daniel Coore
University of the West Indies (Mona)
Kingston, Jamaica
daniel.coore@uwimona.edu.jm

ABSTRACT

The Growing Point Language (GPL) is used to engineer the emergent behaviour of an amorphous computer. GPL patterns are topologically one-dimensional objects, regardless of the dimension of the space in which the system exists. A crude length measure in GPL means that GPL patterns also have a geometric character to them. One of the constructs defined in GPL (*diatropisim*), directs a growing point to propagate tangentially to the level curve of a spatial distribution called a pheromone. In 2-dimensions, tangent spaces are 1-dimensional and therefore diatropism is reasonably well defined. However, in 3-dimensions (and higher) *diatropism*is no longer confined to 1-dimension, which means that some programs whose behaviour was well understood in 2-dimensional systems, become less so in higher dimensions. We argue that the predictability of the geometric properties of a GPL program in 3-dimensions can be completely recovered. We support this argument with the presentation of a program that given a centre point, a direction, and a radius will generate a circular path in the plane containing the centre, that is normal to the given direction. We provide quantitative data from a single run to illustrate how well the geometric objectives can be achieved.

Categories and Subject Descriptors

C.2.4 [**Distributed systems**]: Distributed applications

General Terms

Algorithms, Design, Experimentation, Languages

Keywords

Amorphous computing; programming languages; self-organising; spatial computing; swarm intelligence

1. INTRODUCTION

An amorphous computer consists of myriad nodes, which are identically programmed, operate asynchronously and are irregularly located within a distributed network. Each node is able to communicate with its neighbour, store some local state and do simple calculations[1]. Many tools have been developed to engineer an amorphous computer [2, 3, 5]. One such tool is the Growing Point Language (GPL) [4]. Work done with GPL has, so far, been in two-dimensions; we discuss in this paper one method for extending the language to controlling behaviour in 3-space.

Central to GPL, is the growing point: a locus of computation that changes the local state of a processor and moves from one node to the next in response to virtual stimuli in the amorphous system. These virtual stimuli (pheromones) are secreted by the same or other growing point(s). Computationally, they are results of diffusions that produce radially symmetric, monotonically decreasing values around the source. The growing point's movement relative to a pheromone's level curve is defined by a tropism. The specified tropism can cause the growing point's movement to be orthogonal (*ortho*), tangential (*dia*) or oblique (*plagio*) to the level curves of pheromones.

A growing point resides in one node at a given time, as it moves from one node to another, its trajectory forms a pattern. As such, GPL programs produce patterns made up of one-dimensional paths. When translating GPL to 3-D, the *ortho* tropism maintains its definition as it produces one-dimensional structures regardless of what dimension the system runs in. However, growing points using tropisms that are tangential to a level curve (*dia* and *plagio*) propagate to neighbours that are constrained to a dimension one less than that in which the sytem runs, and lose the determinism they had in 2-space.

As such, our problem is to regain control in drawing circular structures in 3-dimensions. We present a GPL program that self-organises a circular path that is specified by its centre, c (provided as a node), its radius, r, and the plane in which it lies, specified by a normal, $\mathbf{n}$ (provided as a node, whose direction is interpreted relative to the point representing the centre).

2. IMPLEMENTATION

We derive two spheres whose centres are co-linear with the intended centre of the circle Figure 1. Pheromones secreted from each centre, define a pair of spherically symmetric concentration gradients, whose plane of intersection contains the desired circle we desire to construct. We then measure the required radius from the given centre within this plane to find a point on the circumference of the required circle. Finally, a growing point moves tangentially to the pheromone levels of both spheres at the same time, which causes it to outline the desired circle.

Growing points working to complete a task can be grouped into networks. Networks take as inputs places where growing points can be invoked, and produce the desired termination

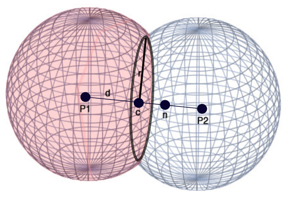

Figure 1: We specify a circle of centre, c, radius, r and be in the plane normal to n created by the intersection of two spheres (p_1 and p_2)

locations that need to be used outside of the network. To construct a curve in 3D we cascaded two networks. The first to locate the two centres of the spheres and the second to draw the circumference.

2.1 Locating the two centres of spheres

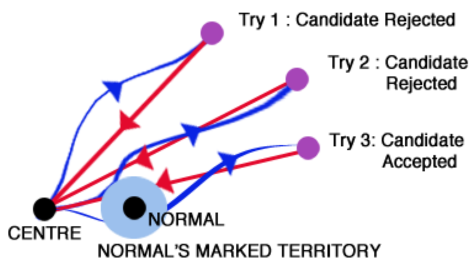
Try 1 : Candidate Rejected
Try 2 : Candidate Rejected
Try 3: Candidate Accepted
CENTRE
NORMAL
NORMAL'S MARKED TERRITORY

Figure 2: User provides centre, c, and normal, n. Candidates are produced and one is chosen if it passes through c and near n

The first network, accepts, c and **n**. We then construct two points co-linear with c and **n**, located a user defined d hops on either side of them Figure 1). To do this, we produce a ray from the c through **n**, and for the opposite direction from **n** through c. Figure 2 shows this for one side.

Rays are produced by self-repelling growing points, which meander rather than produce straight lines (blue lines in Figure 2). To ensure that a candidate, at the end of a ray, is in the correct direction, we start a growing point that moves from the candidate to the centre in a straight line, (red lines in Figure 2). If the growing point intersects the normal or any nodes in its territory on its way back to the centre point, we know the candidate is in the correct otherwise a search for a new candidate is done (try 1 and 2 in Figure 2). Once two co-linear points are found, they are passed as inputs to the second network for drawing the circumference.

2.2 Drawing the circumference

The two centres, each secrete for at least, $(r+d)$ hops, so their intersection will be large enough to accommodate a circle of radius, r. To draw the circle, we first locate a point on the circumference. From the center, a growing point, which propagates *ortho-* to both sphere pheromones for r hops is produced. This yields a point, s, on the circumference. The desired circumference is defined by the set of nodes whose pheromone concentrations are the same as the starting point, s. The growing point responsible for drawing the circle is diatropic to both sphere pheromones.

3. RESULTS

A node has a diameter of 1 unit.The simulation run has approximately 45,000 nodes in a volume of 100x100x100 units. The radius of communication used was 6 units.

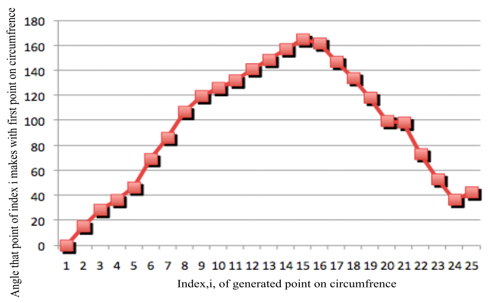

Figure 3: The graph shows the angle that a point (of index i) makes with the first generated point

The angles that each point created with s are shown in the graph above. We see a steady increase and then decrease during the trace. Table 1 shows some statistics of the run. In particular the points are approximately equi-distant from the centre (standard deviation of 0.5) therefore indicating a circular shape.

Table 1: Analysis data for run

Best fit Normal ($\mathbf{n}_b$)	(1.6, -0.4, -1)
Given Normal ($\mathbf{n}_u$)	(65.8, 51.4, 50.2)
$p_1 - p_2 = \mathbf{n}_v$	(-29.53, 5.22, 11.47)
$\mathbf{n}_b \cdot \mathbf{n}_u$	0.195861958
$\mathbf{n}_b \cdot \mathbf{n}_v$	-0.982245164
Standard deviation of radii	0.5

4. CONCLUSIONS

Using this method, a user can draw one-dimensional structures using *dia* or *plagio* by combining the geometrical properties of secretions, in a way that restricts the space.

5. REFERENCES

[1] Harold Abelson et al. Amorphous computing. *Communications of the ACM*, 43(5):74–82, 2000.

[2] Jonathan Bachrach and Jacob Beal. Programming a sensor network as an amorphous medium. *MIT-CSAIL-TR-2006-069*, 2006.

[3] Arnab Bhattacharyya. Morphogenesis as an amorphous computation. In *Proceedings of the 3rd Conference on Computing Frontiers*, pages 53–64. ACM, 2006.

[4] Daniel N Coore. *Botanical computing: a developmental approach to generating interconnect topologies on an amorphous computer*. PhD thesis, Massachusetts Institute of Technology, 1999.

[5] Radhika Nagpal. Programmable self-assembly using biologically-inspired multiagent control. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 418–425. ACM, 2002.