**Computational Creativity in Slime Mould Simulations**

Jake Cooper

jc2088@kent.ac.uk

# Abstract

In this report, I will attempt to evaluate what it means for something other than humans to be creative through the creation of a simulation of the behaviour of slime mould. The simulation will exhibit behaviour similar to that shown by slime mould organisms and have an interactive layer for an end user to influence and interrupt the simulation. The report will go through the design decisions that were made throughout the creation of the system, and what computation creativity theories these decisions drew from. Through screenshots leading to its completion, it’ll go through technical explanations of how the simulation works followed by results produced by said system. This report will conclude with an evaluation of the system and its results, comparing it against various models of computational creativity, followed by conclusions and reflections on the project as a whole.

# Introduction

The goal of computational creativity is generally to create a system that mimics the behaviour and decision-making process of the human mind in a way that us humans would deem creative.The mimicking and analysing of creative processes that beings outside of humans express is a mostly unexplored area and by doing this project, I want to delve into what it could mean for something other than humans to be creative and if our current ways of evaluating computation creatively apply to this new space. I hope to achieve this by building a program that can emulate the behavioural patterns of a slime mould, with interactive elements that will allow the user to interact with and influence the decisions of this visually fascinating organism. The program will then be evaluated with both a simplified version of Ritchie’s Criteria and Colton’s Creative Tripod. Although both these methods are rarely used with more focus being on the SPECS model, I think that they will give a good enough understanding of current measures of computational creativity. Finally, the inspiration of this project comes from the work of Sebastian Lague’s project on Ant and slime simulations, the sources and papers of which I used frequently in being able to complete this work.

# Background Reading

To begin asking this question of what it means for something other than a human to be creative, it is necessary to first take a look into a few current definitions of creativity. It makes sense here to first look at Alan Turing’s test of computational intelligence, the Turing Test. It doesn’t take long for us to start disputing the Turing Test as it specifically refers to the imitation of human behaviour where a machine is considered intelligent if it can fool an unbiased observer into believing its output is from a human. We can however expand this definition to cover our new area of interest and say that if the machine can mimic the behaviour of whatever it is trying to compete with, to a degree that an unbiased observer would not be able to distinguish the two. This similar, but expanded definitions lets us attempt to recreate any system that exhibits complex behaviour and by comparing it to the original, evaluate whether it shows the same creativity and intelligence. This of course has the same flaws as the original statement, being that it encourages the development of convincing front ends with no guarantee of the creative process being the same, however this is hard to avoid if we want to look at the problem solving and creative explorations of emergent behaviour systems such as the Slime Mould that I will be looking to emulate. This is because emergent systems, inherently work on very little information on the individual level, not understanding the complexities and optimisations that they are making on the way, and it is only through them working in unison that these behaviours come to surface meaning we can only judge them by their output.

Following from the Turing Test, Ritchie’s Criteria also focuses on the output of the creative system and what proportion of that output is novel, atypical and valuable. This way of evaluating creativity becomes particularly important when you think of what the possible problems that could occur with this project in particular. Since my system will work with individual, relatively naïve agents that work together to great a complex behaviour, it is perfectly reasonable to believe that in some cases, there will be no emergent creative behaviour. This is one of the reasons I chose Ritchie’s Criteria to evaluate my system as the proportion of valuable output and irrelevant output is a large problem that could occur when working with emergent creative behaviour rather than a single creative process.

The final background reading I did was Colton’s paper where he introduces The Creative Tripod. In this paper, Colton mentions that it is common for the programmer to be attributed with the creativity rather than the computer itself. This is a view point that I’ve struggled to find a definite answer to whilst doing Creative Computing as I find it easy to attribute the creativity of a system such as ClueGen to the implementation by the authors rather than the system. This problem becomes particularly exaggerated when looking from our context of creativity displayed by emergent behaviour, as slight parameters that control options such as sensitivity to other agents can change the behaviour exhibited drastically. Does this mean whoever is settings the parameters and observing the results should be given credit for the creativity of the output? I hope that by creating this system I will be somewhere closer to answering a question like this. The Creative Tripod is unique to the other two methods talked about in that it attempts to define creativity by the process that the system takes rather than simple the output. Although I don’t expect my final system to be able to satisfy all the points of this creative evaluation, I think using it will present an insight into how creative systems such as the one I hope to create can be evaluated.

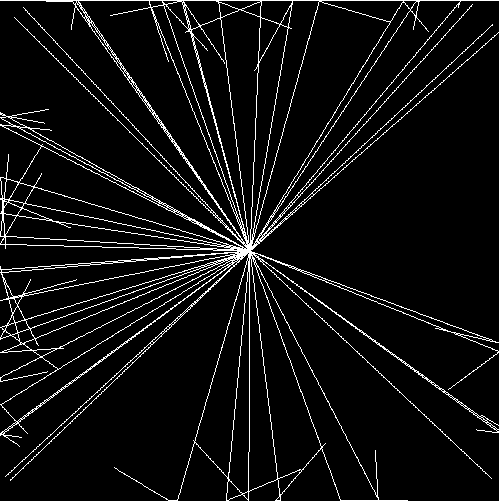
# Design and Implementation

The basis of my system is setting up a simulation of the behaviour I want to recreate. Since Sebastian Lague’s project was my inspiration, I used the same paper that he used to recreate a simulation of a slime mould known as Physarum. This paper is by Jeff Jones and will be the template I use to create this system. I decided to use Processing 3 throughout this project as it’s a flexible, java-based IDE and programming language that makes working with visual elements incredibly easy and removes a lot of the back-end work that working with pure java or something even lower level such as C++ would provide. The main downside of using Processing however, is that it is a language that lends itself to mostly being ran on the CPU. There are ways to code for the GPU through shaders on a different version of the language however I don’t know how to use this. This is relevant as, with a lot of simulation computations, there are a lot of very similar, small mathematical equations being ran every frame. This will be explained a bit more for each feature but if you can imagine 50,000 simulated agents can be optimised to run in parallel on the GPU much more efficiently than on the CPU. Due to this limitation, some of the features had to be toned down a little to achieve acceptable runtime but this does not affect the performance of the system creatively in any way.

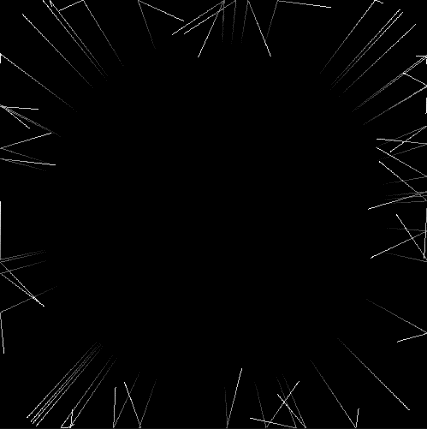
To begin with, I had to create the individual agents that would interact with each other. Each agent individually, is very simple and when put into a class, have simply their position and rotation as values that they hold. The starting life cycle of these agents is very simple and is taken from Jones’ paper. Every iteration, each agent will attempt to move forwards in the direction it is facing and if it is unable to move in that direction, a new random direction is chosen at random and the agent will attempt to move forwards again. After it has moved, it will colour the pixel is it currently over. This is of course repeated for all agents that have been spawned in. This gives us a solid foundation to work with. When initiated in the centre of the screen with random rotations, the simulation looks as shown in *Figure 1*.

The trails that each agent leaves behind, obviously should not linger forever. Therefore, the next thing that needs to be implemented is the fading of these trails overtime to imitate the evaporation of deposited components of the mould. To implement this, we simply need to loop through every pixel and reduce its colour value by a given amount, decided by the variable *evaporateSpeed*. This gives the effect that each agent is leaving a fading path behind it and prevents them from simply filling in the entire canvas one colour over time, as shown in *Figure 2*.

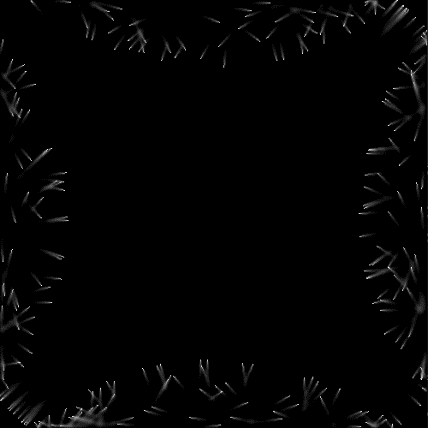
Next, it is necessary that the paths left behind by each agent diffuse over time into the surrounding area. This process mimics the natural diffusions of the chemicals released by the mould and therefore is an important feature of the simulation. There is no easy way to do this other than to conduct a smoothing or blur of the entire canvas every frame. To do this, each pixel is looped through, and replaced by the average colour value of the pixels around it in a 3x3 mask. This part of the simulation is where I first experienced the effects of being limited to only working on the CPU, as a blur like this looping through every pixel multiple times to gain an average is very inefficient. This meant that the resolution of the canvas I was working on would make a huge impact on how long each frame would take, with higher resolutions such as 1000x1000 only managing to output around 5 frames per second. Due to this, I decided on a canvas resolution of 500x500 to stick with throughout the rest of the project, as this hit a nice balance between acceptable performance but also a satistfying level of detail. The effects of the full screen blur can be seen in *Figure 3*



*Figure 1*

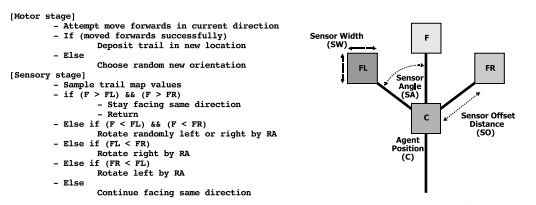


*Figure 2*



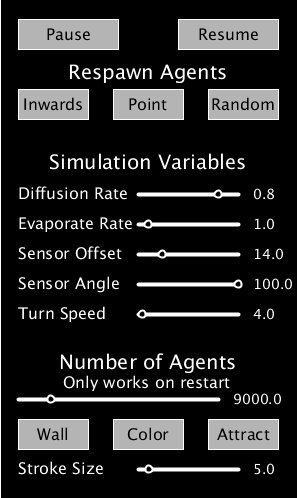
*Figure 3*

The final step of setting up the simulation would be implementing the sensory behaviour of each agent. As outlined in *Figure 4*, each agent would look at its three sensors (the size of which I set to 3x3 each) and depending on which sensor has the sum of all its components the highest, would be chosen as the direction to turn. This behaviour was fairly simple to implement thanks to the simple description provided by Jones’ and it leaves many variables to play with in order to have the simulation behave differently.



*Figure 4 Screenshot from Jones’ paper describing the behaviour of the slime mould*

With the simulation now completed, I wanted to add an easy to control user interface and other features so that the someone running the program could interact and influence the simulation. This would be the final thing to add in order to more easily attempt to see if the system could be behaving creatively. Once the base components were set up such as buttons and sliders for the first time, adding new ones for each setting came with relative ease and I decided to just place them all on the right of the simulation with a slider for each variable that could be changed. In order to interact with the simulation, I added 2 features. First, was the option to draw in and remove indestructable walls. This could block the path of the mould and force it to discover a new way around and could possibly even force it to navigate a maze. The second feature, labelled attract, was something that would leave a trail behind your mouse cursor that would act as an encoragement for the agents to move towards it. This meant that you could effectively draw with the simulation and I thought would be an interesting feature to add.



*Figure 5 Settings Menu*

# Results

As far as what I wanted to achieve in the program I created, I am very happy with what has been produced. The program is incredibly fun to play around with and being able to mess with, influence and interrupt the flow of this incredible looking simulation. I can picture quite clearly using an application like this in order to make artwork that portrays an important message, such as having the almost infectious looking simulation work its way through a drawn out map of the UK but something like that would be for someone more creative than mine. On certain settings, the simulation also shows forms of path finding, navigation and exploration of areas which is definitely interesting to see. As far as the program itself goes I don’t think there is anything that I would do differently other than attempting to learn to have it run more efficiently on the GPU like others such as Sebastian Lague have before. This would greatly improve the capabilities and possibilities of this kind of project and would perhaps lead to many more avenues to explore

# Evaluation

The first creative evaluation method I would like to use is a simplified version of Ritchie’s Criteria. This method looks heavily at the proportion of valuable output that a system presents and especially when it comes to something with this many user settings and an essentially random output depending on the infinite possible starting configurations, I think will give a valuable insight into the computational creativity. Although it is difficult to pick out the exact criteria (*Shown in Figure 6*) that my system satisfies, I will still attempt to take the concept of these criteria and evaluate against them. As far as the typicality of my output, I think that all outputs produced are suitably typical as, when given similar or the same conditions you can fairly easily predict the style of movement and behaviour that the simulation will exhibit. There is very little randomness in the behavioural system itself (only in the selecting of the starting conditions) and therefore when given the same input, a typical output is always produced. When it comes to the value of the output, it is very dependent on the conditions you give it. In good starting conditions, the behaviour of the system produces valuable outputs throughout but when the starting conditions are misaligned and not set up correctly then the output of the system is simply incomprehensible and uninterpretable. Finally, the novelty of the output is fairly low. Due to the lack of randomness implemented through the behaviour of each agent, if the chance is given for the simulation to run under the exact same conditions twice in a row, it will behave almost if not exactly the same as it did in the previous run. This means that despite there being novelty in the solutions to problems that the emergent behaviour of the system shows, it will act exactly the same if given the chance to. To conclude the evaluation using Ritchie’s Criteria, I think it is clear that this method of testing creativity is beneficial for a different type of creative system. It showed that there is nothing new that my system is coming up with and that the output is almost always typical and valuable but I cant help but feel that these criteria benefit a single output program rather than a continuous stream of information that a simulation provides.

*Figure 6 Simplified Ritchie’s Criteria*

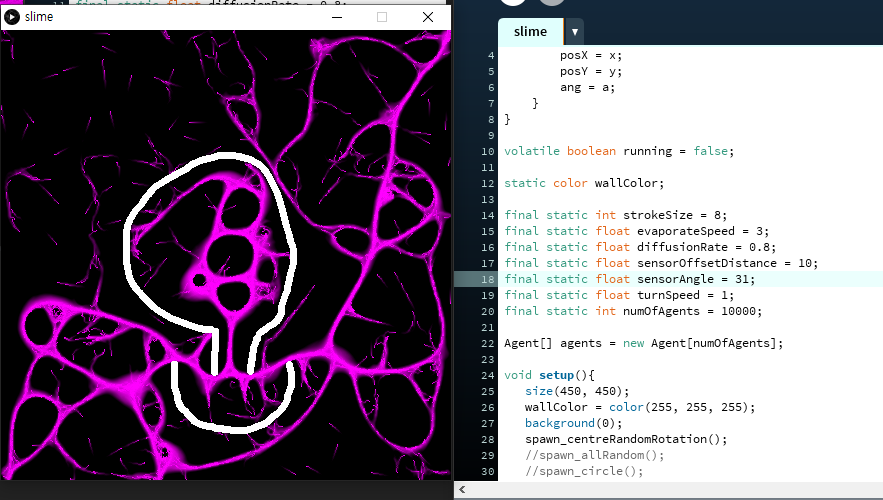
Evaluating a system by the standards of Colton’s creative tripod is very easy. I simply have to assess three main points of my system. Skill: the system’s ability to show its mastery of the area its working on. Appreciation: the system’s ability to critically assess the value of what it is producing. Imagination: the system’s potential to innovate its process for creating outputs. For skill, I think that it can be said that my system displays an ability and mastery in the area of creating an output that mimics slime mould behaviour. Some results from the system have a striking resemblance to the shapes and methods that the mould exhibits and I think this is plenty for us to be able to say that the skill leg has been satisfied. Appreciation however cannot be satisfied. Since each agent is working individually and sees a very small part of the entire picture, there is no capability for the system as a whole to assess and put a value to the output it is producing. For a system like mine, who’s behaviour is expressed as a culmination of many small, simpler behaviours it is simply impossible for any appreciation for the entire image to be implemented. This simply doesn’t apply at all to emergent behaviour, and I think should be one of the focusing points when we think about how to evaluate the creativity of systems like this. Finally, in terms of imagination I think you can argue to some extent that the behaviour of the system shows an ability to innovate, explore and come up with solutions to problems put in front of it. An example of this can be seen when a wall is used to split a section of the simulation in two, and over time the behaviours of each agent will discover and figure out how to traverse around this obstacle. It’s definitely a stretch as there is no way for the system to perhaps adapt its settings per agent to deal with different types of problems but I think this point can be partially satisfied.

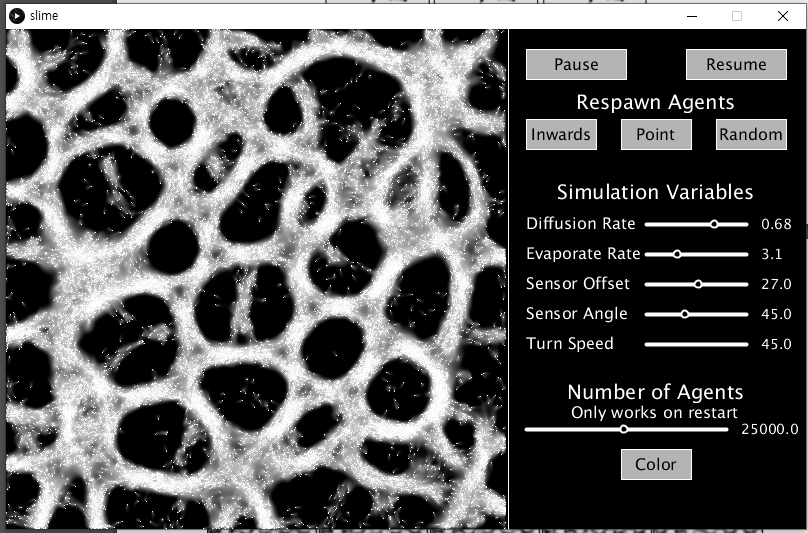
# Conclusions

Although valuable in assessing certain types of computationally creative systems, I think that current evaluative models struggle to account for creativity shown by systems that differ from the concept of human creativity. I don’t think this applies only to the emergent behaviour that I’ve created but to the forms of creativity that other species or even mammals exhibit as well. Of course, context often plays a large role in whether someone considers artwork or ideas creative or not, which plays to the benefit to systems such as mine which have the context of being how a slime mould would behave, but without this context, the focus of these models is often on the system being able to critically assess themselves which does not and sometimes can not apply to other types of creativity.

Reflecting on this project, I would have liked to further explore the boundary between a system being creative or the user interacting with the system being creative. There was a slight introduction to this through the features that I added to my program such as the changing of settings and ability to add obstacles, but this was no the focus of the evaluation. I would have also liked to improve the computational efficiency of the program more as I feel this limited the scope of what I could do with this concept.

# *Appendix*

*Interesting behaviours shown by certain settings*

**

**

# *Sources*

Sebastian Lague’s Video on his implementation of Slime Mould Simulations [ <https://www.youtube.com/watch?v=X-iSQQgOd1A> ]

Jeff Jones’ “*Characteristics of Pattern Formation and Evolution in Approximations of Physarum Transport Networks”* which was the template for my simulation [ <https://uwe-repository.worktribe.com/output/980579> ]

Simon Colton’s “*Creativity Versus the Perception of Creativity in Computational Systems”* where he introduces the Creative Tripod [ <https://www.aaai.org/Papers/Symposia/Spring/2008/SS-08-03/SS08-03-003.pdf> ]

Sage Jenson online blog on his implementation of Jones’ paper [ <https://cargocollective.com/sagejenson/physarum> ]