Summary of Lenia:

Lenia is a system of continuous cellular automata. It was derived from Conway's Game of Life by making everything smooth, continuous and generalized. It is a two-dimensional cellular automaton with continuous space-time-state and generalized local rule. It can support a greater diversity of complex autonomous patterns, "life forms", that differ from other cellular automata patterns in being geometric, matemeric, fuzzy, resilient, adaptive, and rule-generic.

The colab tutorial showing the progression from Conway's Game of Life to Lenia is available online:
https://colab.research.google.com/github/OpenLenia/Lenia-Tutorial/blob/main/Tutorial_From_Conway_to_Lenia.ipynb#scrollTo=F1NF30u4JTG4

This tutorial begins by implementing the original Game of Life cellular automata. It generalizes this by using convolution and a growth function. This way, the cellular automata uses a convolution with a kernel instead of counting the neighbors. The growth function replaces the conditional update in the original automata. Once the Game of Life has been generalized in this manner, it is ready to be extended to continuous cases. Larger-than-Life extends it to a continuous space. Primordia extends it to continuous states, and Lenia combines these and extends it to continuous states-space-time.

Larger-than-Life enlarges the kernel we created above to a given radius creating a continuous space.

Primordia allows for multiple states. The growth function is scaled up for the states and the neighbor sum and initial conditions both take these states into account. It helps to normalize the various properties, states, kernel, and growth ranges to restrict their values making further generalizations much simpler. This also effectively makes the states continuous.

We can add to this continuous state cellular automata, Primordia, by defining an update frequency that scales the updates making time continuous as well.

Once we have continuous states and time, we can add in the continuous space from Larger-than-Life. This is where Lenia starts to form. Once the kernel is smoothed using a bell-shaped function and the growth function is also smoothed by the same type of function, we finally arrive at Lenia.

Since the creation of Lenia, many extensions have been discovered. For example, for more efficient calculations, the naive convolution calculation can be replaced by fast Fourier transform. Another extension involves extending the kernel into multiple rings instead of just a single ring. An even more ambitious extension was to extend it to multiple kernels and multiple growth functions making the kernels, neighbor sums, and growth values into lists. Another extension was to extend the world into multiple channels, such as RGB.

Extensions aren't the only way that Lenia has been experimented with. Variants of Lenia have also come about. One such variant involves an asymptotic update changing the growth function into a target function which calculates a target state that the cell should approach. In multi-channel Lenia, growth or target function can be used for different channels, creating even more complex, interesting organisms. The last extension mentioned here involves changing the soft clip function to hard clip which creates smoother patterns and can exhibit long-range sensing.

Some initial experimentation with initial patterns and kernels in Lenia has led to the following observations.

Multiple initial patterns have been discovered in the Lenia framework. When these patterns are paired with a specific kernel, they perform interesting, life-like actions. These initial patterns and kernels are very sensitive to one another. If a stable pattern and kernel are found, deviating slightly from the initialization or kernel can cause the cellular automata to exhibit random noise or crash completely to nothing. There seem to be very few of these stable states compared to unstable ones. An interesting direction for future research could be experimenting with the discovery of the stable, versus unstable, conditions of Lenia.

Several kernels have been applied to the Lenia cellular automata. All of the kernels in the demonstration provided at https://chakazul.github.io/Lenia/JavaScript/Lenia.html, involve unimodal, bimodal, trimodal, and tetramodal kernels of circular rings. If one applies a square ring kernel, the resulting cellular automaton behaves significantly differently. This work was unable to discover a stable state for the square ring kernel experimented with, but it is worth noting that the only technique applied was manual perturbation of the initial state. The square kernel, however, also goes against the "smooth" nature of Lenia, so this could also be a contributing factor. An interesting research direction building upon this could be to use various functions to generate differently shaped / oriented kernels to explore further in this regard. Another interesting direction that has been touched on recently is 3-dimensional Lenia with 3-dimensional states, kernels, and growth functions.

Lenia is a recent extension of cellular automata that has an immense amount of potential for experimentation and discovery and has already started to see attention from researchers. There is still much to be discovered in regards to almost every aspect of Lenia's composition and theoretical implications. This work only begins to present the possibilities that Lenia offers.

try different initial patterns
-- generate results

Multiple initial patterns have been discovered in the Lenia framework that, when paired with a specific kernel, perform interestingly and in many cases resemble digital life.

These initial states, when paired with the correct kernel, perform similar to digital life, but are very sensitive to perturbations or changes.

There are very few stable states compared to the number of unstable states. How do we find these stable states? We could possibly do an exhaustive search. Maybe we could plot the trajectories and find stable states similar to how we did with lotka-volterra model with the

try different kernels
-- generate results

All of the kernels that I can find in the Lenia demo involve 1, 2, 3, or 4 kernel separators, and every single one of them has a circular shaped kernel.

One of the things I tried was to utilize a square kernel. This ended with a totally different pattern emergence than the circular kernel. When applied to the initial patterns discovered for Lenia, it does not produce interesting results and usually crashes to zero pixels. I believe this is at least partially due to the fact that the square ring kernel I am using has strict boundaries instead of curved smooth ones. This square kernel also exhibits a bit of a chaotic or unstable environment with random pixels whereas the circular kernel with the bell formula calculation generates a much more stable environment with random initialization.

What if we tried using a different function besides a bell curve to create the kernels?

Maybe the bell curve and circles in the kernel is a part of what leads to the stable states that resemble artificial life. We may not be able to find this same type of stability with differently shaped or differently calculated kernels.

What if we had a 3D kernel and had lenia in a 3d space?

-- Try to observe anything that consistently changes.

After playing with some kernels and initializations, it appears that there are certain kernels that will cause more stable states, especially those kernels that are more simplistic. It tends to be much easier to find a stable state with simplistic kernels. It is much more difficult to find stable states with more complex kernels. It is possible, but the initializations much be more specific and refined to do so. My intuition says that the reason behind this stability and, if you want to call it self-organization, is the tendency of the kernel to cause convergence toward something other than nothing or chaos. Otherwise the simulation just reduces to what resembles random noise or, in contrast, no pixels activated at all. The system is trying to find equilibrium. If it can find it without becoming random noise or degenerating to no activations at all, then it will and this is a stable state that can exhibit certain behaviors. I believe also that if this stable state is an oscillating or moving state, that the simulation may exhibit more complex behaviors. It's almost as if we are searching for the most complex state that also is somehow in stability or near stability.

One of the consistencies that I'm starting to observe is that if there are multiple kernels, it matters which one is strongest and that there is at least one that is relatively strong. If not, it may be hard to find a stable state. That being said, I also believe that these kernels that are much harder to find stable states for, may end up having stable states that exhibit higher degrees of complex behaviors since they have to in order to exist in stability or near-stability.