Politecnico di Torino

Politecnico di Torino

Laurea magistrale in Ingegneria del Cinema e dei Mezzi di Comunicazione

A.a. 2025/2026

Sessione di laurea Marzo 2026

Lenia Mixer

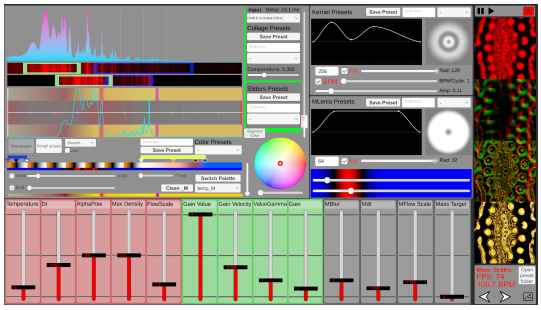
Tool per il mixing di un automa cellulare in real time

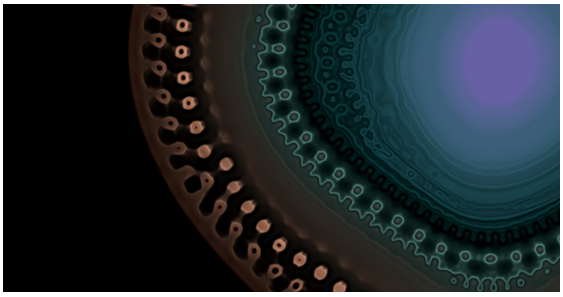
Relatore: Andrea Bottino

Candidato: Adriano Valentini

Abstract

Il presente lavoro di tesi descrive la progettazione e l'implementazione di un sistema per la modulazione in tempo reale dei parametri di un automa cellulare.
Definiremo a breve cos'è un automa cellulare e i suoi meccanismi ma semplicemente un automa cellulare è una simulazione che si basa sull'evoluzione dei valori di una griglia di celle (pixel) in base al valore dei pixel che hanno intorno.
Le regole che determinano questo nuovo valore sono altamente personalizzabili e da esse si possono far emergere comportamenti molto diversi, nello strumento da me sviluppato è possibile cambiare il valore di molti parametri che descrivono queste regole grazie a degli strumenti manuali e servendosi dello spettro di un segnale sonoro.
Questo tool è ideato per creare degli effetti visuali originali che rispondono reattivamente agli stimoli sonori musicali, perfetti per le installazioni grafiche della musica elettronica o live performance.
Il software sviluppato con il motore grafico Unity è un applicazione dual display dove il primo display contiene con tutti i controlli necessari per gestire i parametri di simulazione ordinati in una interfaccia compatta mentre il secondo visualizza il risultato a schermo intero.
Ho creato il pannello di controllo in modo da gestire tutti i parametri puramente numerici con degli slider come un mixer audio in cui slider e nobs servono per regolare i volumi. Inoltre queste variabili sono associabili a dei controller MIDI USB per poter usare una periferica esterna hardware per modularli simultaneamente con più praticità.
L'utilizzatore dovrà maturare una certa esperienza nella ricerca delle configurazioni più interessanti e più inerenti al contesto musicale e performativo e dev'essere in grado di cambiarle con padronanza, per questo la figura tecnica che gestisce questo strumento deve maturare delle abilità simili a quelle di un musicista e di un DJ.





Indice

Elenco delle figure v

1 L'arte generativa nella live performance 1
- 1.1 Arte generativa . . . 1
- 1.2 AI Generativa . . . 4
- 1.3 Live performance e Visual Jockey . . . 4
- 1.4 Generative AI nella Live Performance . . . 6
- 1.5 Automi Cellulari nella Generative Art . . . 6
- 1.6 Il mio strumento per il mixing di FlowLenia . . . 7

2 Celular Automata 8
- 2.1 Sistemi Emergenti . . . 8
  - 2.1.1 Cos'è l'emergenza . . . 8
  - 2.1.2 L'Universo come Sistema Emergente . . . 8
  - 2.1.3 Componenti caratteristici . . . 9
  - 2.1.4 Il Potere delle Esplosioni Combinatorie . . . 9
- 2.2 Cos'è un automa cellulare . . . 10
  - 2.2.1 L'origine storica . . . 10
  - 2.2.2 Automi cellulari 1D . . . 11
  - 2.2.3 Motivi della ricerca sugli automi cellulari e le sue applicazioni pratiche . . . 13
  - 2.2.4 Automi Cellulari 2D . . . 14
  - 2.2.5 Automi Cellulari Multistato . . . 14
- 2.3 Game of Life . . . 16
  - 2.3.1 Operazioni del ciclo di simulazione . . . 17
  - 2.3.2 Complessità emergente . . . 19
  - 2.3.3 Turing completeness . . . 20
  - 2.3.4 Personalizzare Game of Life . . . 21
- 2.4 Da Game of Life a Lenia . . . 22
  - 2.4.1 Gestione dello spazio e della profondità . . . 22

2.4.2 Approssimazione del continuo . . . 23
2.4.3 Kernel dimensione e struttura . . . 24
2.4.4 Funzione di Crescita. . . . 24
2.4.5 Convoluzione e accelerazione hardware GPU . . . 25
2.5 Conservazione della massa . . . 26
2.5.1 Fragilità di Lenia classica . . . 26
2.5.2 Matrice di affinità (U^t) . . . 27
2.5.3 Matrice di flusso (F^t) e α . . . 27
2.5.4 Reintegration Tracking (RT) . . . 29
2.5.5 Vantaggi Grafici . . . 30

3 Lenia Tool 31
3.1 Introduzione . . . 32
3.1.1 Menù . . . 32
3.1.2 Avvio e interfaccia complessiva . . . 33
3.1.3 Struttura generale . . . 34
3.2 Pipeline di Simulazione . . . 36
3.2.1 Due Simulazioni . . . 36
3.2.2 FFT . . . 37
3.2.3 Affinità e GrowthLUT . . . 37
3.2.4 Flusso di materia . . . 38
3.2.5 Combinazione del Macro e Micro Flusso . . . 38
3.2.6 Movimento della Massa e Reintegration Tracking . . . 38
3.2.7 Shader di Simulazione . . . 39
3.3 Pannello di Controllo . . . 40
3.3.1 Preset Manager . . . 41
3.3.2 Manipolazione dello spettro . . . 41
3.3.3 Creazione della palette colori . . . 44
3.3.4 Micro-Kernel Editor . . . 47
3.3.5 Macro-Sim Editor . . . 48
3.3.6 Pannello laterale: controlli e preview . . . 49
3.3.7 Image Mode . . . 50
3.3.8 Sliders . . . 52
3.4 Visualizzazione . . . 57
3.4.1 Spostamento, zoom, e disegno . . . 57
3.4.2 Paint mode . . . 57

4 Conclusioni 64
4.1 Usabilità . . . 64
4.2 Prestazioni . . . 65
4.2.1 Convoluzione e FFT . . . 65

4.2.2 Reintegration Tracking 65
4.2.3 Doppia pipeline e misura della massa 65
4.3 Sviluppi futuri 66
4.3.1 Lenia e l'universo espanso 66
4.3.2 Particle Life 67

Bibliografia 68

Elenco delle figure

1.3 Refik Anadol, Living Architecture: Gehry 5
1.4 Kappa futur festival, Torino. ph: Dan Reid 5
1.5 Generazione di creature 7

2.1 Stephen Wolfram (1983). Regola 110. 12
2.2 Conus textile, mollusco marino che presenta pattern molto simili alla regola 30 di Wolfram. 13
2.3 Diverse configurazioni di automi cellulari multistato. Simulazioni effettuate con Cellarium. https://benpm.github.io/cellarium/ https://benpm.github.io/blog/gol_2/ 15
2.4 Game of life: evoluzione di una struttura "glider". 16
2.5 Finestra dell' intorno. 17
2.6 Tabella delle specie. https://blog.xojo.com/2022/05/11/conways-game-of-life/
2.7 Computer basilare implementato in GOL. https://www.nicolasloizeau.com/gol-computer 20
2.8 GOL implementato in GOL.https://oimo.io/works/life/ https://www.youtube.com/watch?v=xP5-iIeKXE8 20
2.9 https://www.shadertoy.com/view/NlfGDr 22
2.10 La maschera limitata di GOL diventa un esteso kernel in Lenia e la funzione gradino diventa una funzione continua. 23
2.11 https://www.shadertoy.com/view/sdsyWN 24
2.12 Orbium Unicaudatus: $\mu = 0.15$, $\sigma 0.017$. https://www.shadertoy.com/view/7ls3z7 25
2.13 Mappa delle specie in Lenia.[1] 26
2.14 Simulazione Lenia / FlowLenia.[2] 27
2.15 Reintegration Tracking.[2] 29
2.16 Specie in FlowLenia.[2] 30

4.1 Expanded universe Lenia 66
4.2 Particle Life Simulator 67

1. Introduzione

L'arte generativa nella live performance

1.1 Arte generativa

"Il termine Arte generativa si riferisce al concetto di "Arte che genera arte" dove, l'opera artistica, è il prodotto di un sistema autonomo in grado di determinare le caratteristiche."[1]

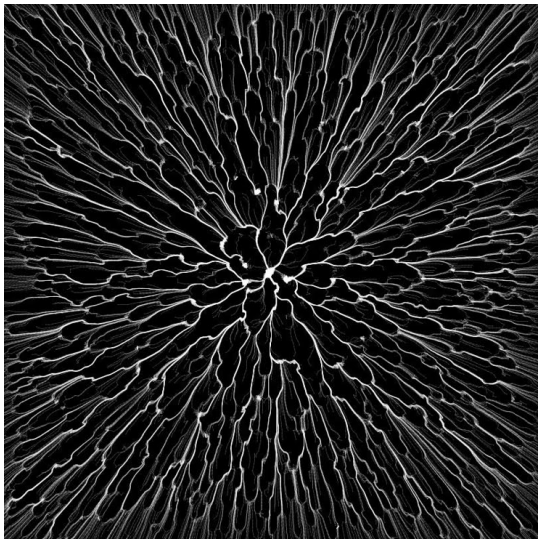
"Generative Art is the idea realized as a genetic code of artificial events, as the construction of dynamic complex systems able to generate endless variations following the Nature world. Each Generative Project is a concept-software that works producing unique and non-repeatable events, like music, images, or 3D Objects, as possible and manifold expressions of the generating idea identified by the designer as his/her subjective proposal of a possible aesthetic world."[2]

L'arte generativa è la pratica di definire un insieme di regole e lasciare che l'opera prenda forma attraverso l'esecuzione autonoma di tali regole. L'autore non compone direttamente ogni dettaglio ma gestisce lo spazio delle possibilità. In questo senso, la generazione non è un espediente tecnico, ma un modo di pensare il processo creativo.

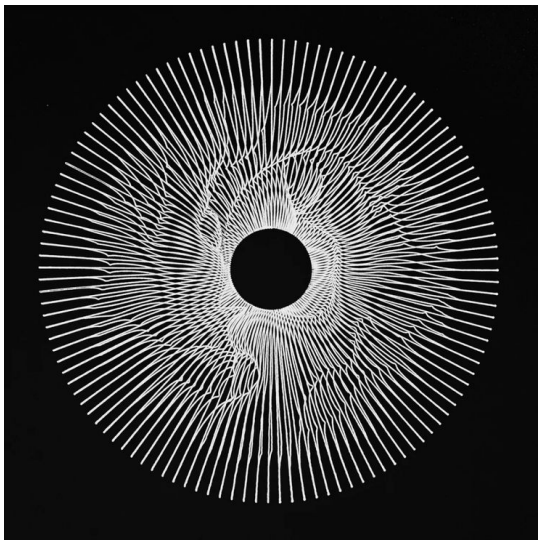
[1] `https://it.wikipedia.org/wiki/Arte_generativa`

[2] `https://soddu.it/`

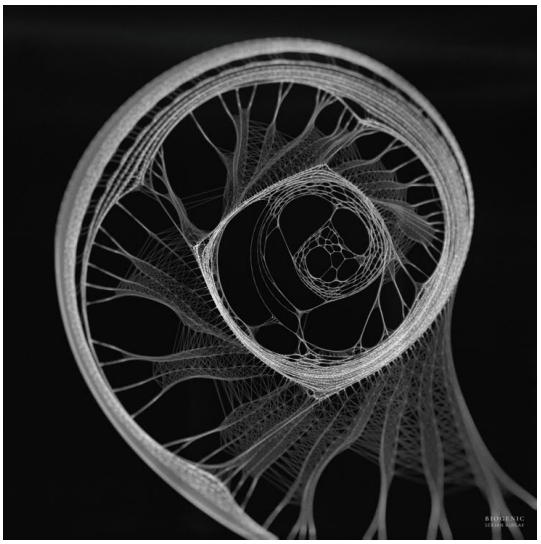
(a) Simon Alexander-Adams: Just Undulation, adding motion to a reaction diffusion variation.



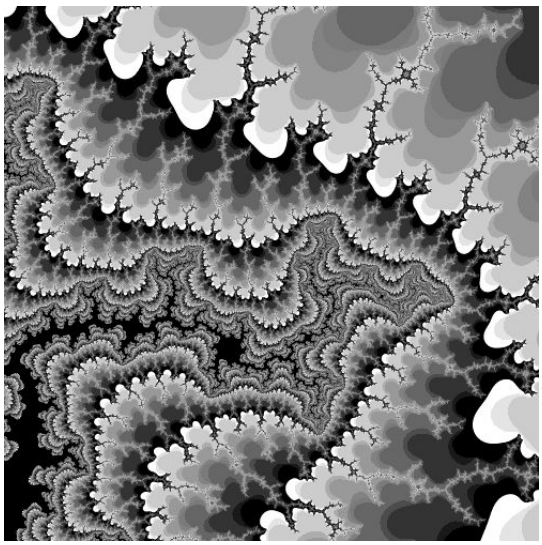
(b) Daniel Eden: Drawing With Numbers.



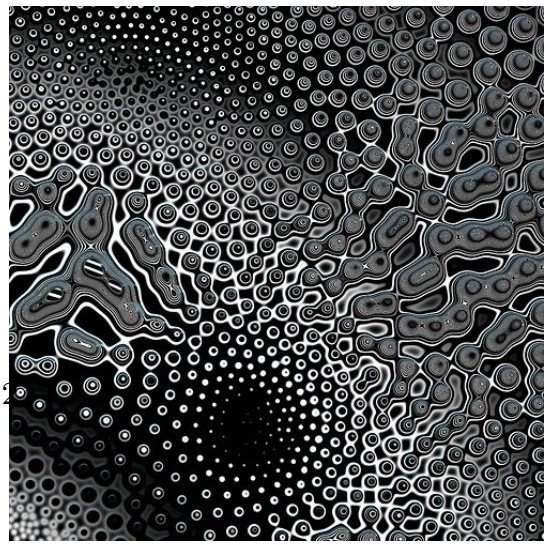
(c) Pierre Pasilier: Cinema4D for generative art.



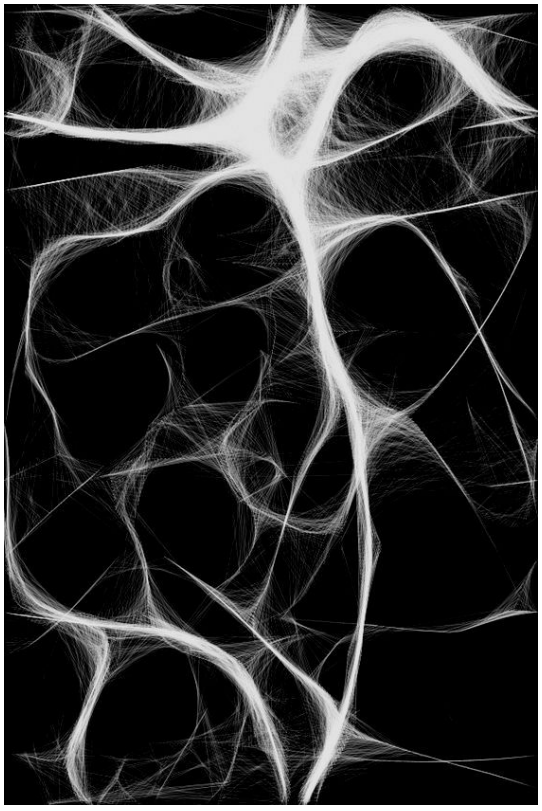
(d) Serjan Burlak: Geometry of Intelligence.



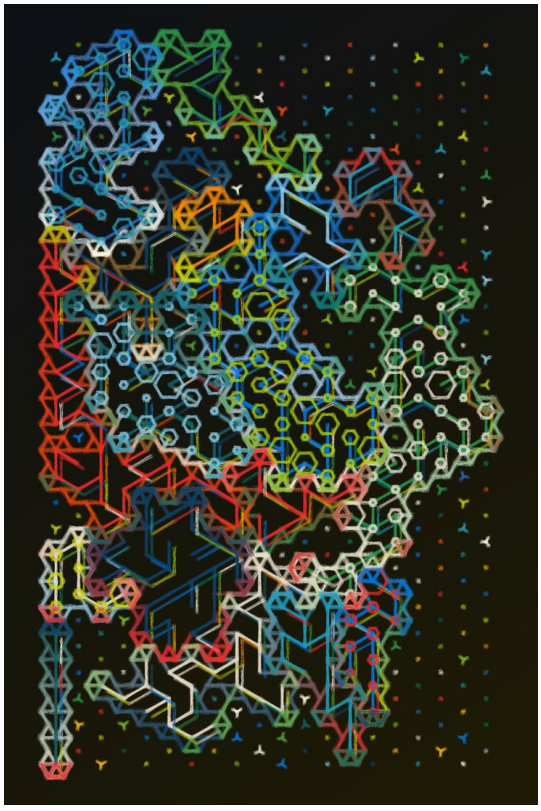
(e) Rednor: Animated Fractal.



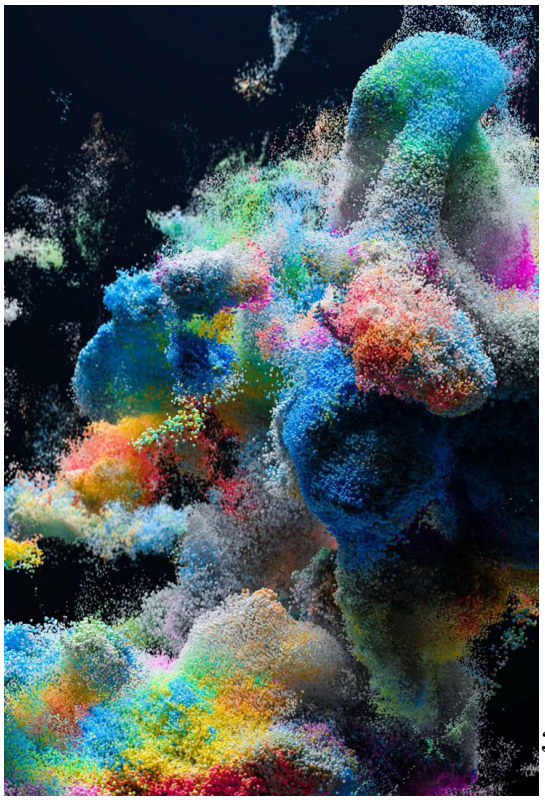
(f) Robert Hodgin: Visualization of forces generated by subatomic interaction.



(a) Gilles Deuleuze e Félix Guattari: Concept.



(b) Gorilla Sun: Generative Art with Javascript.



(c) leManoosh: Bubble rendering.

(d) leManoosh: Reflections.

1.2 AI Generativa

Negli ultimi anni si è diffusa rapidamente una nuova categoria di sistemi di intelligenza artificiale, chiamata Generative Artificial Intelligence. A differenza delle forme tradizionali di AI, progettate per riconoscere, classificare o prevedere dati, i modelli generativi sono in grado di creare nuovi contenuti: immagini, testi, suoni o video che non esistono nei dati di partenza. Questo è possibile perché questi modelli non si limitano a memorizzare esempi, ma apprendono le regole statistiche e strutturali che definiscono un insieme di dati e le utilizzano per produrre nuovi risultati che rispettano quelle stesse regole.

Tra le diverse tecniche di intelligenza artificiale generativa, i modelli di diffusione rappresentano oggi uno degli approcci più avanzati e diffusi. Il loro principio di funzionamento si basa su un processo di ricostruzione progressiva: partendo da un'immagine composta unicamente da rumore casuale, il modello impara a rimuovere gradualmente quel rumore fino a ottenere una figura coerente. Durante l'addestramento, la rete neurale osserva una grande quantità di immagini reali e impara a prevedere come esse vengono corrotte e ricostruite, in modo da poter invertire il processo.

Nel campo della generative art, questi modelli permettono di esplorare nuove modalità di creazione, in cui l'artista stabilisce solo alcune condizioni iniziali, come una forma, un colore o una descrizione testuale, e lascia che l'intelligenza artificiale completi l'opera secondo la propria logica interna.

1.3 Live performance e Visual Jockey

Visual Jockey (o VJ) è un'artista visivo che in tempo reale crea, manipola e mixa immagini in sincronia con la musica o altri contenuti performativi.
Il VJ nasce negli anni 80 90 come mixer di video clip, quindi contenuti non generativi, allo scopo di sincronizzarli alla musica. Ha un ruolo simile a quello del DJ che non crea il suono ma manipola e interpreta dei brani preesistenti.
Negli ultimi anni grazie alla potenza dei nuovi tool di programmazione visiva e alla incrementata potenza computazionale è possibile la creazione di contenuti di arte generativa in tempo reale. Ora il VJ spesso produce i propri contenuti che essendo prodotti al momento possono rispondere a dei input istantanei, in questo caso il VJ diventa un performer di arte generativa, suona e improvvisa uno strumento visuale.
Il VJ moderno sfrutta entrambi i mondi, usa contenuti preregistrati spesso raffiguranti scenari realistici e usa contenuti generati per ottenere un effetto astratto e complesso di simulazioni artificiali.

Figura 1.3: Refik Anadol, Living Architecture: Gehry

Figura 1.4: Kappa futur festival, Torino. ph: Dan Reid

1.4 Generative AI nella Live Performance

L'impiego delle intelligenze artificiali generative nelle performance dal vivo e nei live musicali rappresenta oggi un ambito di sperimentazione in rapida evoluzione, ma ancora caratterizzato da importanti limiti tecnici e concettuali. Nella maggior parte dei casi, i contenuti visivi generati dall'AI non vengono prodotti in tempo reale, ma sono materiali prerenderizzati, ovvero sequenze o clip create in precedenza attraverso modelli generativi come le reti neurali o i modelli di diffusione. Questi contenuti vengono poi integrati nella performance e sincronizzati con l'andamento generale del brano, ad esempio seguendo variazioni di intensità, passaggi strutturali o cambi di atmosfera, ma raramente riescono a reagire in modo diretto e preciso alla metrica musicale, al battito (BPM) o a singoli eventi sonori come una nota o un colpo di batteria.

La ragione principale di questa limitazione risiede nella complessità computazionale dei modelli generativi moderni, che richiedono tempi di calcolo troppo lunghi per garantire una risposta immediata. Anche nei casi in cui l'AI è impiegata in tempo reale, la sua funzione è spesso più decorativa che strutturale: genera variazioni visive, distorsioni o pattern psichedelici su materiali preesistenti, piuttosto che costruire l'immagine o la narrazione visiva da zero. Di conseguenza, il ruolo dell'intelligenza artificiale in queste performance è quello di amplificare e modulare l'esperienza sensoriale, introducendo un grado di imprevedibilità e organicità che arricchisce la componente visiva, ma che difficilmente si integra con precisione nei tempi e nelle dinamiche musicali.

In questo senso, l'AI generativa agisce più come un co-performer autonomo che come uno strumento pienamente controllabile: contribuisce a creare ambienti visivi in continua trasformazione, ma la sua interazione con la musica resta per ora mediata e parziale, più vicina a una sincronia percettiva o emozionale che a una sincronizzazione tecnica rigorosa.

1.5 Automi Cellulari nella Generative Art

Nel corso degli ultimi decenni, sistemi come gli automi cellulari hanno mostrato come regole locali semplici possano produrre comportamenti globali complessi.
Negli ultimi anni modelli continui come Lenia estendono questa intuizione, non è più un mosaico di pixel accesi o spenti, ma un fluido di materia che pulsa, si divide e si aggrega. È qui che l'arte generativa dialoga con l'estetica della simulazione: ciò che vediamo non è disegnato, è emergente.

1.6 Il mio strumento per il mixing di FlowLenia

Ho approcciato il mondo degli automi cellulari nel corso di quest'ultimo semestre universitario in cui ho frequentato il corso di GPU programming, io e il mio gruppo abbiamo deciso di programmare in CUDA una simulazione su griglia in cui più creature (blob) competevano per contendersi il cibo e si evolvevano tramite reinforcement learning.

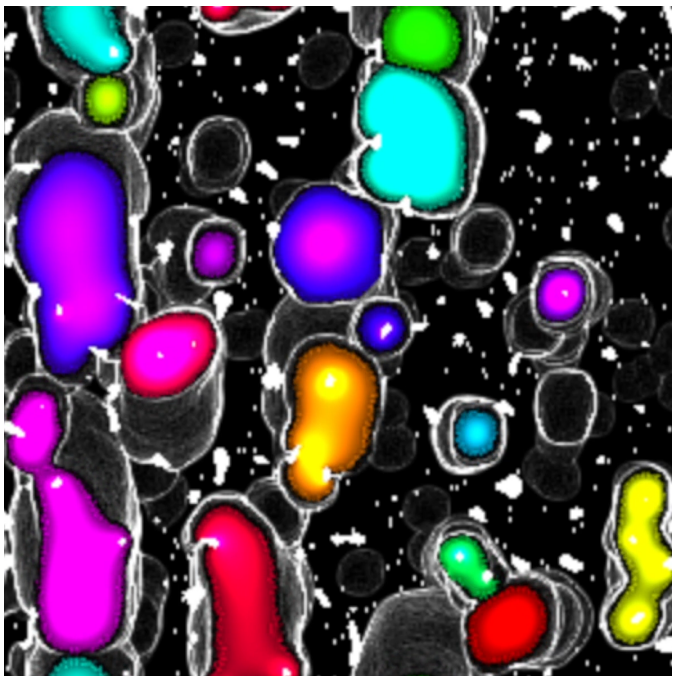


Figura 1.5: Generazione di creature

Questa esperienza mi ha dato consapevolezza sull'enorme potenzialità del calcolo parallelo e la capacità computazionale delle GPU. Inoltre mi ha incuriosito l'improbabilità delle simulazioni di automi cellulari che, diversamente da simulazioni fisiche che emulano fenomeni reali di cui siamo a conoscenza, creano mondi completamente alieni che rispondono a regole arbitrarie.
Ho deciso di fare uno studio estetico della fenomenologia degli AC e nello specifico di un automa cellulare chiamato "Lenia" di Bert Wang-Chak Chang[1] nella sua versione "Flow"[2].
Di seguito spiegherò in dettaglio il funzionamento di queste simulazioni e le loro caratteristiche, il mio Tool è una applicazione che usa le ricerche sugli automi cellulari, principalmente finalizzate alla scoperta di strutture capaci di sopravvivere ad ambienti simulati, allo scopo di creare effetti grafici interessanti e originali generati in tempo reale.

2. Stato dell'arte

Celular Automata

2.1 Sistemi Emergenti

2.1.1 Cos'è l'emergenza

L'emergenza si verifica quando elementi semplici si uniscono per formare strutture complesse che possiedono proprietà o comportamenti che i singoli elementi non hanno. Il collettivo diventa maggiore della somma delle sue parti.
Consideriamo l'acqua come esempio: una singola molecola d'acqua non mostra le proprietà che vediamo quando trilioni di molecole si combinano. Solo dalla loro interazione collettiva emergono fenomeni come i cristalli frattali dei fiocchi di neve o la tensione superficiale. Queste proprietà non esistono in una singola molecola, ma emergono esclusivamente dall'interazione di molte molecole.

- Le colonie di formiche che realizzano comportamenti complessi impossibili per una singola formica.
- I cervelli che producono pensieri che singoli neuroni non potrebbero generare.
- I linguaggi che emergono dalla combinazione di parole secondo regole sintattiche e semantiche

2.1.2 L'Universo come Sistema Emergente

Nature, it seems, is the popular name for milliards and milliards and milliards of particles playing their infinite game of billiards and billiards and billiards. -Piet Hein Il nostro universo è profondamente emergente: è pieno di elementi semplici che si combinano per creare strutture complesse. È un universo dove il tutto è maggiore della somma delle sue parti. Se lasciamo che gli atomi giochino la loro partita galattica di bigliardo per qualche miliardo di anni, alla fine si formano

stelle e fiocchi di neve, pianeti, persone e galassie. Strutture sorprendentemente complesse emergono da componenti relativamente semplici.

2.1.3 Componenti caratteristici

Tutti i sistemi emergenti condividono due componenti fondamentali:

1. **Gli elementi di costruzione:** sono gli elementi che possono essere combinati con altri elementi simili. Possono essere disposti, assemblati e organizzati insieme. Possono essere oggetti fisici (mattoncini Lego, atomi, particelle, formiche) o concetti astratti (parole, numeri, bit di informazione).

2. **Le Regole:** definiscono il comportamento degli elementi, specialmente quando interagiscono tra loro. Stabiliscono se gli elementi si attraggono o si respingono, se si connettono o si intersecano, se sono soggetti a gravità, momento o attrito.

2.1.4 Il Potere delle Esplosioni Combinatorie

Gli elementi di costruzione possono essere qualsiasi oggetto o concetto combinabile. Consideriamo l'esempio dei bit (codice binario). Ogni bit può essere 1 o 0, acceso o spento, bianco o nero. Combinando i bit in sequenze ordinate:

- Con 2 bit $\rightarrow$ 4 combinazioni possibili.
- Con 3 bit $\rightarrow$ 8 combinazioni.
- Con 10 bit $\rightarrow$ 1.024 combinazioni.
- Con 100 bit $\rightarrow$ oltre 1 nonilione di combinazioni.

Il numero di combinazioni cresce esponenzialmente rispetto al numero di elementi. Questa è un'esplosione combinatoria, il motore della complessità emergente. Esistono molte più combinazioni di elementi che elementi stessi. Questo fenomeno è quasi magico: otteniamo esponenzialmente più di quanto investiamo.
Le esplosioni combinatorie conferiscono un immenso potere creativo a qualsiasi sistema di elementi combinabili. Quando acquistiamo un set Lego, non otteniamo solo il modello previsto, ma infinite altre costruzioni possibili con gli stessi pezzi.
Il linguaggio è un esempio classico. Le parole sono mattoncini linguistici. Quasi ogni frase che pronunciamo, se supera poche parole, è probabilmente la prima volta nella storia che quella esatta sequenza viene espressa.

2.2 Cos'è un automa cellulare

Gli automi cellulari rappresentano una classe di modelli computazionali discreti in grado di descrivere sistemi complessi attraverso l'interazione di componenti semplici.
"Automi": Macchine auto-operanti il termine deriva dal concetto di automa in matematica e informatica, che indica una macchina astratta capace di operare autonomamente seguendo regole predefinite. In pratica, ogni elemento è un piccolo "calcolatore" automatico che segue regole meccaniche semplici, opera in modo autonomo senza controllo centrale e agisce ripetutamente nel tempo.
"Cellulari": Organizzazione in celle, si riferisce alla struttura spaziale del sistema composta da una griglia di celle (o cellule). Nel gioco Life di Conway questo è costituito da un reticolo cristallino di cellule dove ogni casella (cellula) costituisce, in termini matematici, un piccolo automa, vale a dire un piccolo calcolatore.
Ogni cella:

1. Occupa una posizione fissa nello spazio discreto.
2. Ha un proprio stato interno.
3. Interagisce solo con le celle vicine (vicinato locale).

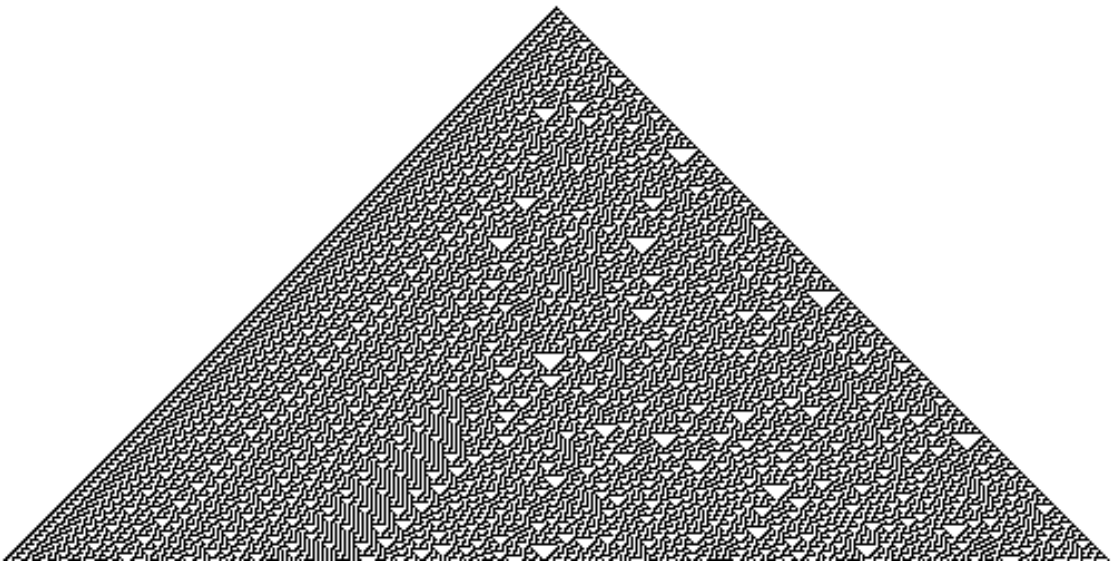
In pratica gli AC si basano su una griglia di celle, ciascuna delle quali assume uno stato appartenente a un insieme finito e aggiorna il proprio valore nel tempo in base a una regola locale. Tale regola, identica per tutte le celle, dipende esclusivamente dallo stato della cella stessa e dei suoi vicini, e viene applicata in modo sincrono a tutta la griglia, producendo un'evoluzione temporale deterministica.

2.2.1 L'origine storica

L'idea dell'automa cellulare è vecchia approssimativamente quanto il calcolatore elettronico digitale. Le prime ricerche furono condotte da John Von Neumann, con un importante contributo di Stanislaw Ulam, nei primi anni cinquanta. Von Neumann cercava di creare sistemi capaci di auto-riprodursi come gli organismi viventi, e la struttura "cellulare" ricordava proprio l'organizzazione biologica delle cellule.

2.2.2 Automi cellulari 1D

Gli automi cellulari unidimensionali, inventati da Stephen Wolfram, sono programmi semplici che seguono regole locali per modificare sequenze di bit. Con regole diverse otteniamo pattern diversi: alcune producono semplici linee, altre generano frattali come il triangolo di Sierpiński, altre ancora creano pattern complessi e apparentemente caotici, ma con struttura nascosta.



configurazione del vicinato	111	110	101	100	011	010	001	000
nuovo stato per la cella centrale	0	1	1	0	1	1	1	0

Tabella 2.1: Progressione temporale di un automa cellulare elementare a una dimensione (ECA) di Stephen Wolfram (1983). Regola 30

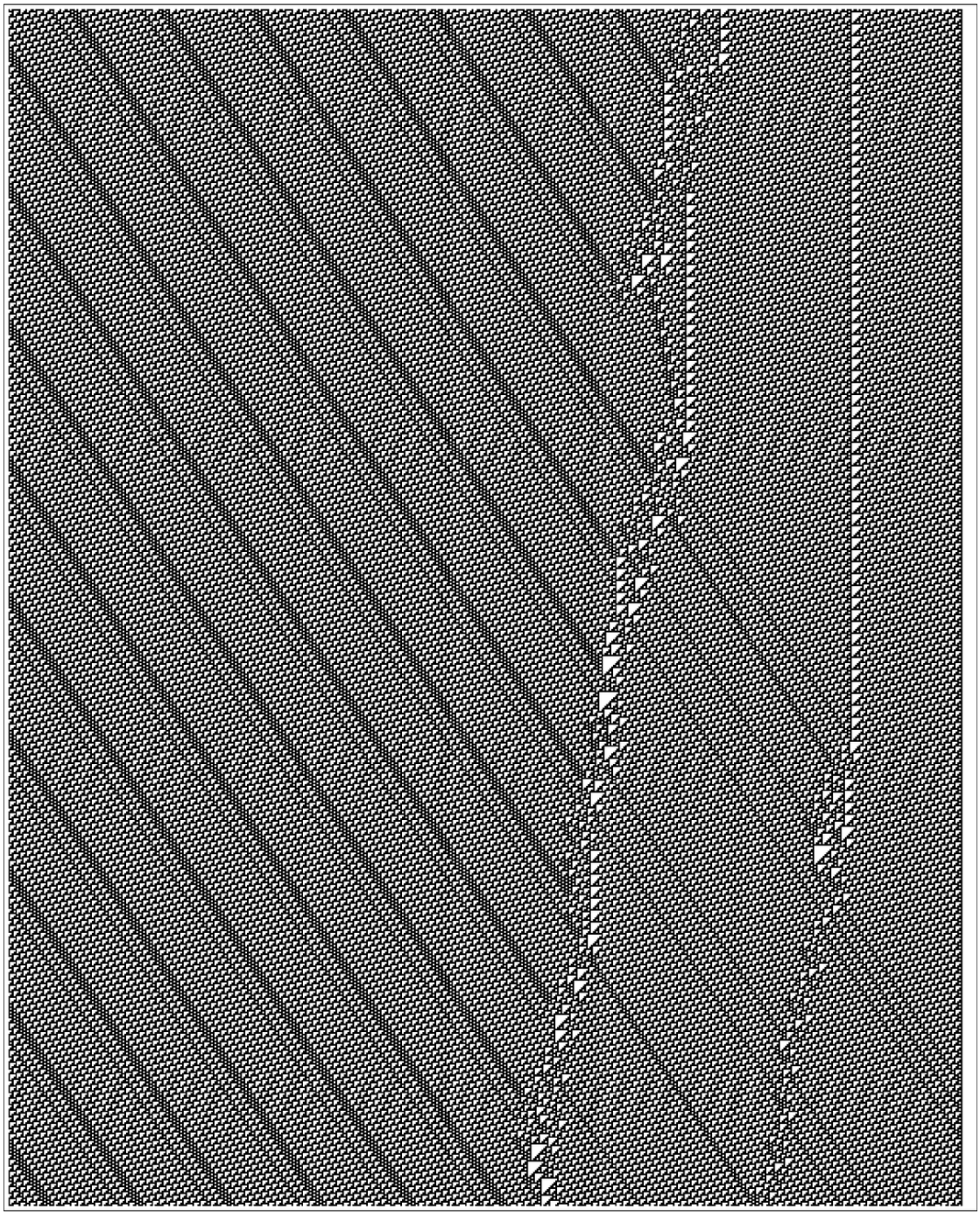


Figura 2.1: Stephen Wolfram (1983). Regola 110.

2.2.3 Motivi della ricerca sugli automi cellulari e le sue applicazioni pratiche

Le ricerche sugli automi cellulari sono motivate da ragioni sia teoriche sia applicative. Sul piano teorico, essi permettono di esplorare la natura della computazione, comprendere l'emergenza di strutture ordinate in sistemi dinamici e investigare le relazioni tra semplicità locale e complessità globale.
Sul piano applicativo, gli automi cellulari trovano impiego nella modellazione di fenomeni naturali come la diffusione di sostanze, la crescita di organismi, le transizioni di fase, la dinamica dei fluidi e le reazioni chimiche. Inoltre, sono alla base di numerosi sistemi di simulazione grafica, ambienti di modellazione biologica, procedure di generazione di pattern e tecniche avanzate di calcolo parallelo e distribuito.
In sintesi, gli automi cellulari rappresentano un ponte tra matematica, fisica, informatica teorica e scienze naturali, e aiutano a comprendere come regole semplici possano generare comportamenti complessi nel mondo naturale e artificiale.

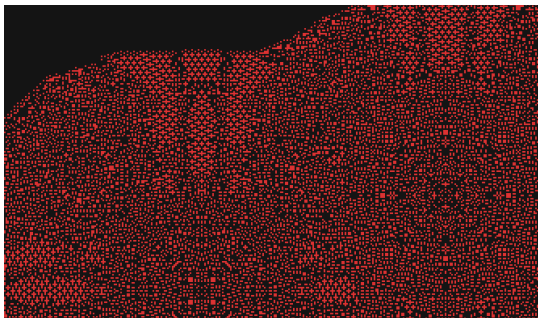
Figura 2.2: Conus textile, mollusco marino che presenta pattern molto simili alla regola 30 di Wolfram.

2.2.4 Automi Cellulari 2D

Un automa cellulare 2D è un modello matematico che permette di mutare una griglia di valori in un' altra seguendo una sequenza di operazioni cella per cella, la simulazione continua ripetendo il processo usando la nuova griglia come stato iniziale.
Il passaggio dalla dimensione unidimensionale a quella bidimensionale introduce una ricchezza di comportamenti e applicazioni significativamente maggiore. Nel caso bidimensionale i più famosi tipi di vicinato sono quelli di Moore, di Von Neumann e di Margolus.
L'intorno di Von Neumann comprende le 4 celle adiacenti ortogonalmente (nord, sud, est, ovest), mentre l'intorno di Moore include tutte le 8 celle circostanti la cella centrale.
Il modello 2D più celebre è indubbiamente il Game of Life di Conway, che con sole quattro regole semplici genera una straordinaria varietà di pattern complessi come alianti, oscillatori e cannoni.
Un percorso di questa ricerca riguarda i tentativi di costruire CA a lungo raggio o continui, e di cercare e studiare pattern autonomi auto-organizzanti, o solitoni. Questi tentativi includono CAPOW (Rucker, 1999), Larger-than-Life (Evans, 2001), RealLife (Pivato, 2007), SmoothLife (Rafler, 2011) e Lenia (Chan, 2019).

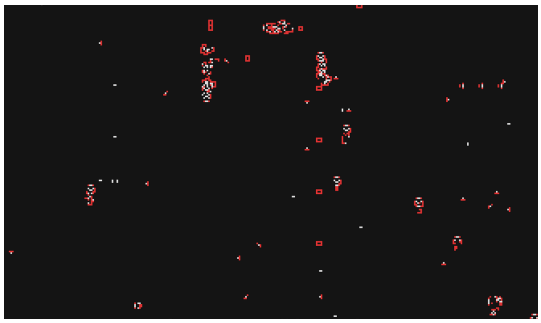
2.2.5 Automi Cellulari Multistato

Mentre gli automi cellulari elementari si limitano a due stati possibili per cella (tipicamente 0 e 1, o *vivo* e *morto*), gli automi cellulari multi-stato estendono significativamente le possibilità espressive permettendo a ciascuna cella di assumere tre o più stati distinti.
Questa generalizzazione consente di modellare fenomeni molto più complessi e realistici. Se k è il numero di stati per cella ed n è il numero di celle incluse nell'intorno, vi sono $((k)^k)^n$ possibili regole , il che evidenzia come l'aumento del numero di stati comporti un'esplosione combinatoria nello spazio delle possibili regole.
Gli automi multi-stato trovano applicazione in simulazioni ecologiche dove ogni cella può rappresentare diversi tipi di vegetazione o livelli di densità di popolazione, in modelli di reazioni chimiche dove gli stati codificano diverse concentrazioni di reagenti, e in applicazioni di computer graphics per la generazione procedurale di texture e pattern naturali.
La complessità computazionale aumenta con il numero di stati, ma anche il potere espressivo e la fedeltà nella rappresentazione di sistemi reali complessi.

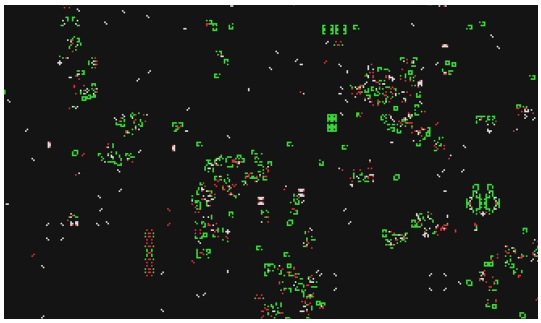


(a) Chaotic Growth

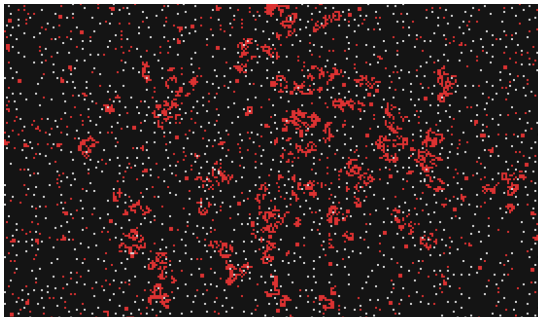
(b) Circuit City



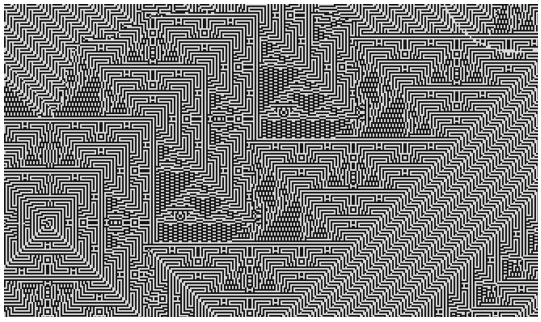
(c) Glider Builders



(d) Meta Stable



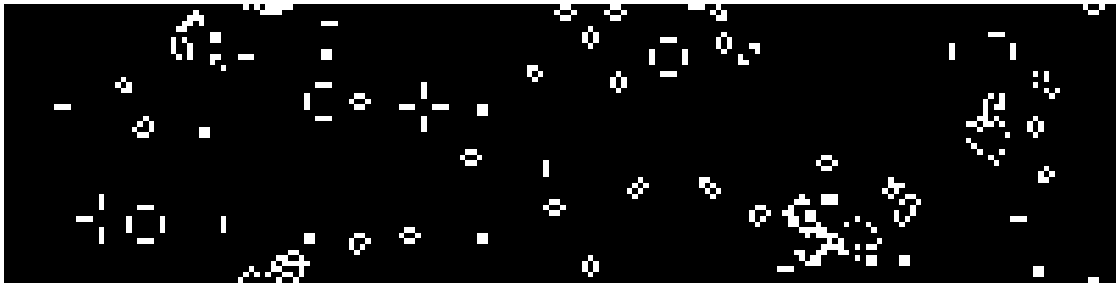
(e) Petri Dish



(f) Tri Circuit

Figura 2.3: Diverse configurazioni di automi cellulari multistato. Simulazioni effettuate con Cellarium. `https://benpm.github.io/cellarium/` `https://benpm.github.io/blog/gol_2/`

2.3 Game of Life



Per introdurre l'argomento consideriamo un automa cellulare semplice come Game of Life.

Game of Life, introdotto da John Conway nel 1970[3], è uno degli automi cellulari bidimensionali più celebri e significativi nello studio dei sistemi complessi. Si tratta di un modello discreto in cui ogni cella di una griglia regolare può trovarsi in uno stato binario può trovarsi in uno di due stati: viva (1) o morta (0) ed evolve nel tempo secondo una semplice regola locale basata sul numero di vicini vivi.
Nonostante la definizione estremamente minimale, Game of Life è capace di generare un'ampia varietà di comportamenti emergenti: strutture stazionarie, oscillatori, pattern in movimento e configurazioni in grado di interagire tra loro.

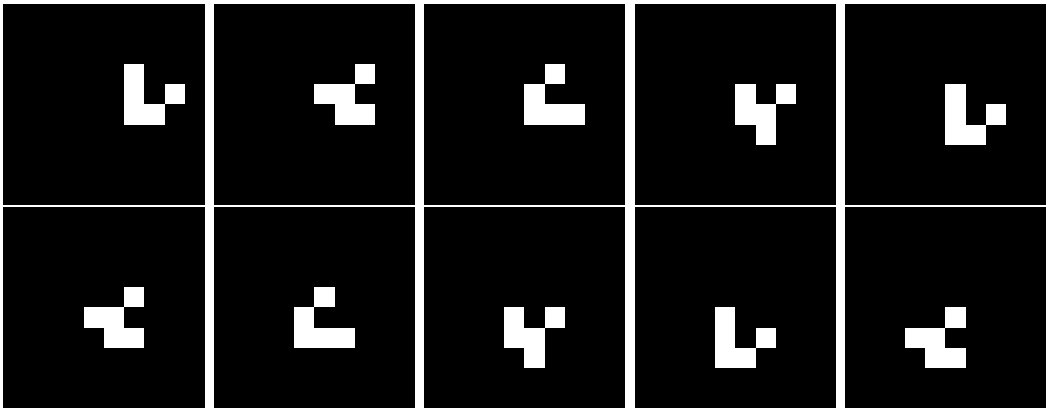


Figura 2.4: Game of life: evoluzione di una struttura "glider".

2.3.1 Operazioni del ciclo di simulazione

Per raggiungere lo stato del successivo passo temporale occorrerà eseguire per ogni cella le seguenti operazioni:

1. **Analisi dell'intorno:** trovare la sommatoria K dei valori intorno alla cella (numero di cellule vive).
2. **Funzione di attivazione:** mappare il risultato nella funzione di attivazione opportuna.

Analisi dell'intorno

L'analisi dell'intorno in Game of Life consiste nel conteggio delle cellule vive presenti nel vicinato di Moore (le otto celle circostanti) di ogni cella.
Questi valori indicano la densità di popolazione e si usano successivamente per individuare nella funzione di attivazione lo stato successivo.

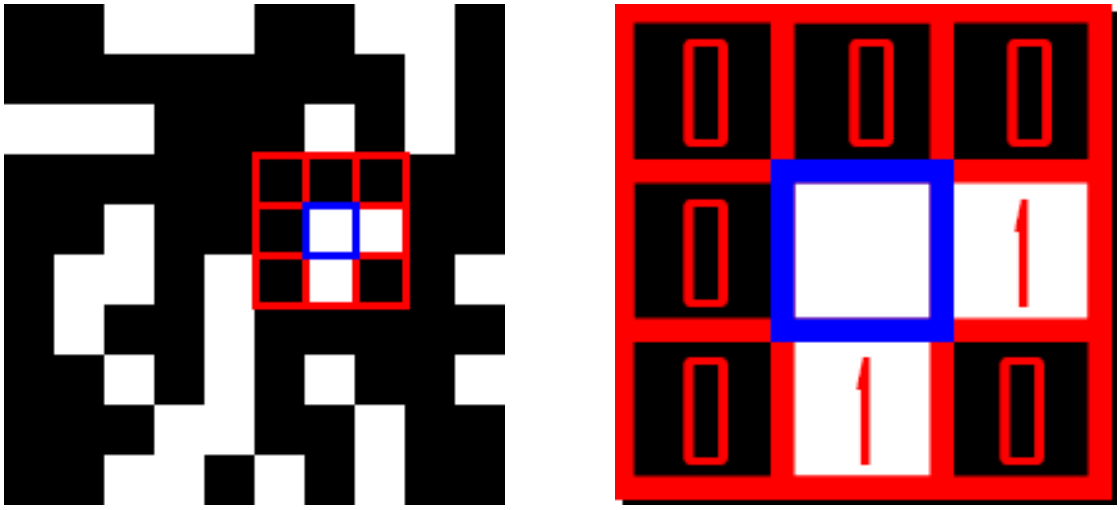
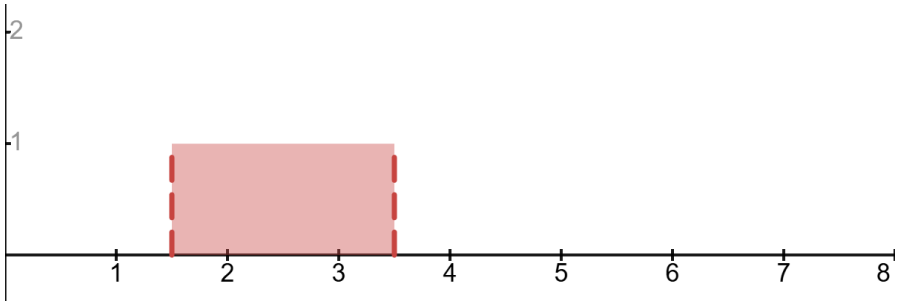


Figura 2.5: Finestra dell' intorno.

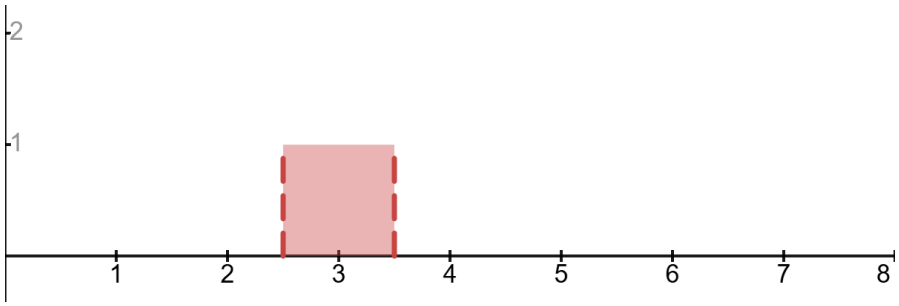
Funzione di attivazione

In Game of Life ci sono due funzioni di attivazione, una per le cellule vive e una per le cellule morte.

FA cellula viva



FA cellula morta



Queste funzioni gradino mappano il numero di vicini con lo stato successivo corrispondente, se una cellula è viva rimarrà viva solo se ha 2 o 3 vicini se no morirà, se una cellula è morta tornerà in vita solo se ha esattamente 3 vicini vivi. In game of life sono queste mappature che hanno lo scopo di simulare basilari principi di sopravvivenza, sovrapopolazione, isolamento e riproduzione.
Le funzioni di attivazione possono essere descritte discorsivamente da queste 4 regole:

1. **Sopravvivenza:** Una cella viva rimane viva se ha 2 o 3 vicine vive.
2. **Morte per isolamento:** Una cella viva con meno di 2 vicine vive muore.
3. **Morte per sovrappopolazione:** Una cella viva con più di 3 vicine vive muore.
4. **Nascita:** Una cella morta con esattamente 3 vicine vive diventa viva.

2.3.2 Complessità emergente

Possiamo notare dei pattern che riescono a stabilizzarsi e non morire. Questi pattern emergenti sono stati classificati in:

1. **Figure stazionarie:** (still lifes) Rimangono immutate nel tempo. Esempi: block, beehive, loaf.
2. **Oscillatori:** Alternano ciclicamente tra alcune configurazioni. Esempi: blinker, toad, beacon.
3. **Astronavi:** (spaceships) Pattern che si spostano nello spazio mantenendo la loro forma. Esempio: glider, lightweight spaceship.
4. **Pattern complessi:** Strutture che generano, distruggono o trasformano altri pattern. Esempi: gun, puffer, breeder.

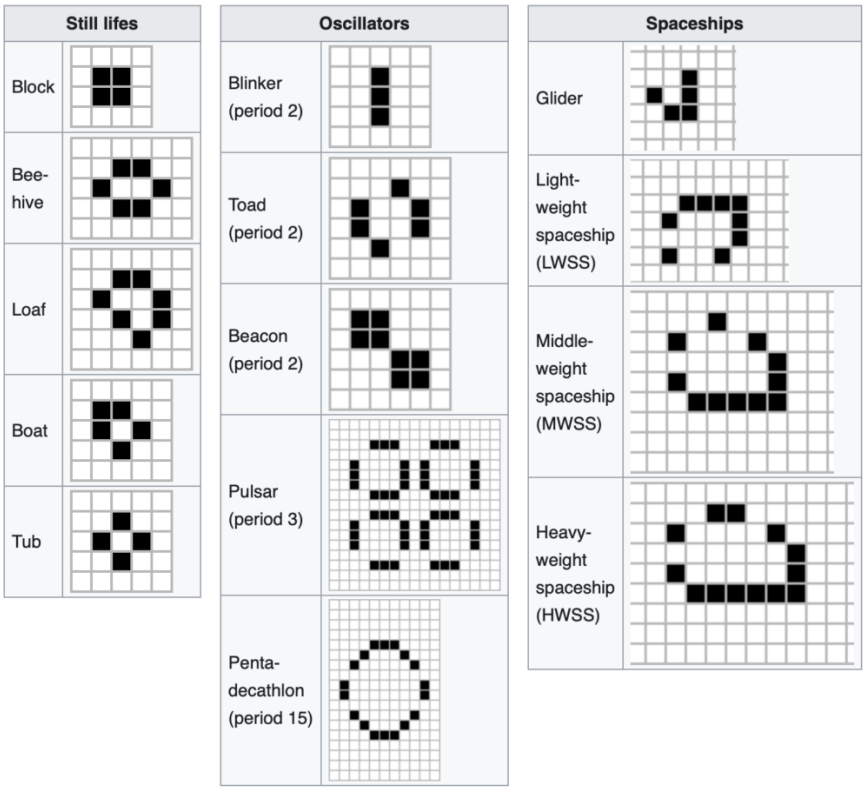


Figura 2.6: Tabella delle specie. `https://blog.xojo.com/2022/05/11/conways-game-of-life/`

2.3.3 Turing completeness

Nel 1982 è stato dimostrato che, pur essendo governato da sole quattro regole locali e da uno stato binario per cella, il Game of Life è in grado di simulare una macchina di Turing, e quindi di eseguire qualunque computazione algoritmica teoricamente possibile.
In particolare strutture come i glider e le glider gun possono essere interpretate come trasmissione di bit e sorgenti periodiche di informazione. Combinando collisioni controllate tra glider, oscillatori stabili e configurazioni opportunamente progettate, è possibile costruire porte logiche, memorie, canali di comunicazione, fino a veri e propri circuiti computazionali completi.

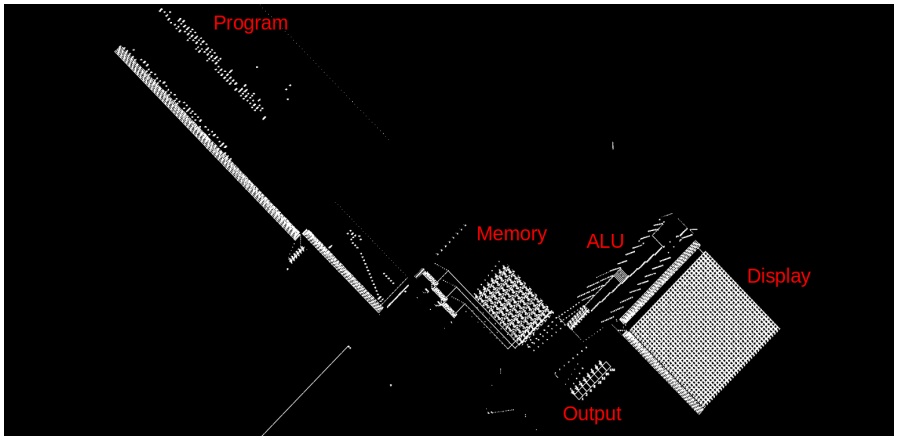


Figura 2.7: Computer basilare implementato in GOL. `https://www.nicolasloizeau.com/gol-computer`

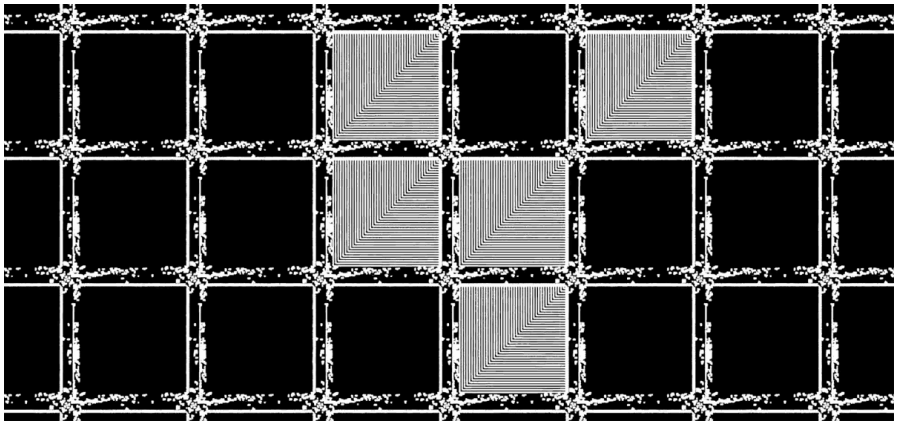


Figura 2.8: GOL implementato in GOL.`https://oimo.io/works/life/` `https://www.youtube.com/watch?v=xP5-iIeKXE8`

2.3.4 Personalizzare Game of Life

Fin'ora abbiamo presentato le strutture emergenti e le caratteristiche delle regole originali di GOL, è possibile ottenere mondi simulati diversi cambiando tre caratteristiche:

- Le operazioni di analisi dell'intorno.
- Funzione di attivazione.
- Le condizioni di partenza.

Cambiare l'analisi dell'intorno

Cambiare le operazioni di analisi dell'intorno può significare considerare più celle. Consideriamo infatti una matrice maschera dispari di valori che se centrata in ogni cella indica i pesi per ogni valore corrispondente.
Si calcola quindi la sommatoria di ogni valore per il peso corrispondente. Normalizzando la maschera si otterrà un valore da 0 a 1 che poi andrà scalato per coprire il dominio della funzione di attivazione.

Cambiare la funzione di attivazione

Cambiare la funzione di attivazione è come cambiare le regole della realtà simulata, è cambiando questa funzione che si determinano le relazioni tra le celle vicine.

Cambiare la configurazione iniziale

Infine anche la configurazione iniziale determina l'evolversi di determinate strutture a discapito di altre ma, in generale, se le regole del gioco favoriscono un determinato pattern, allora esso si manifesterà con più facilità da qualsiasi configurazione randomica.

2.4 Da Game of Life a Lenia

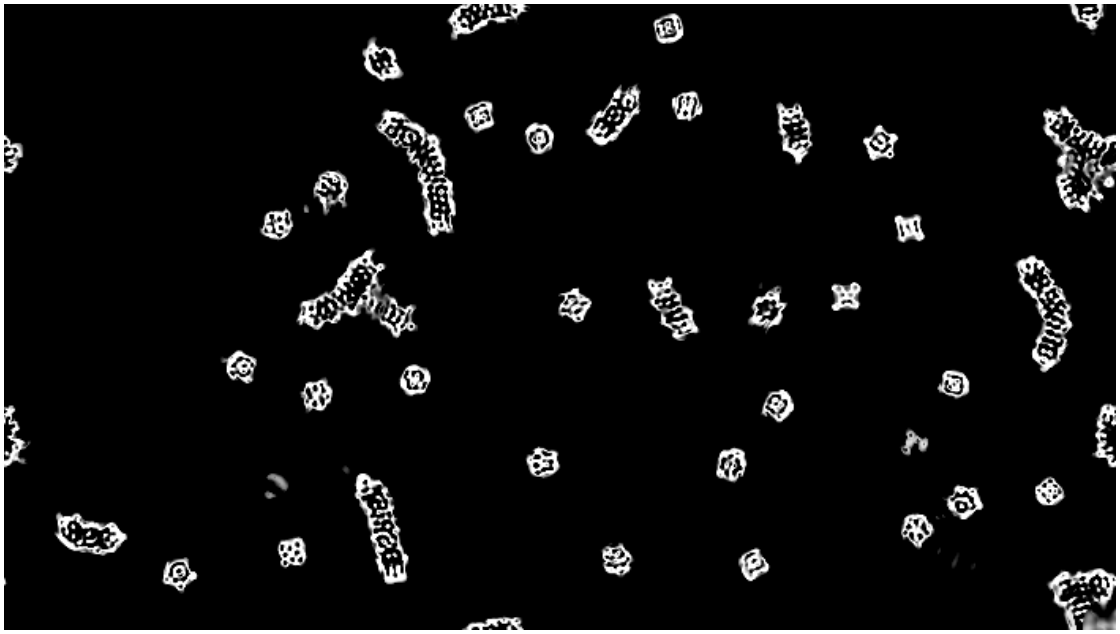


Figura 2.9: `https://www.shadertoy.com/view/NlfGDr`

2.4.1 Gestione dello spazio e della profondità

Game of Life è stato sviluppato sul finire degli anni sessanta, ora con l'avvento delle schede video moderne è possibile ampliare game of Life nella sua versione continua, Lenia.
Il suo creatore Bert Wang Chan la definisce così:
A two-dimensional cellular automaton with continuous spacetime-state and generalized local rule. Lenia is a system of continuous cellular automata, a form of artificial life. It was derived from Conway's Game of Life by making everything smooth, continuous and generalized. [1]
Lenia consiste nell' ampliazione dello spazio e nel tempo della simulazione: La maschera dei valori dei pesi diventa una distribuzione più complessa disegnata su di un kernel più grande. La funzione di attivazione diventa una funzione di crescita per delle evoluzioni più graduali.

2.4.2 Approssimazione del continuo

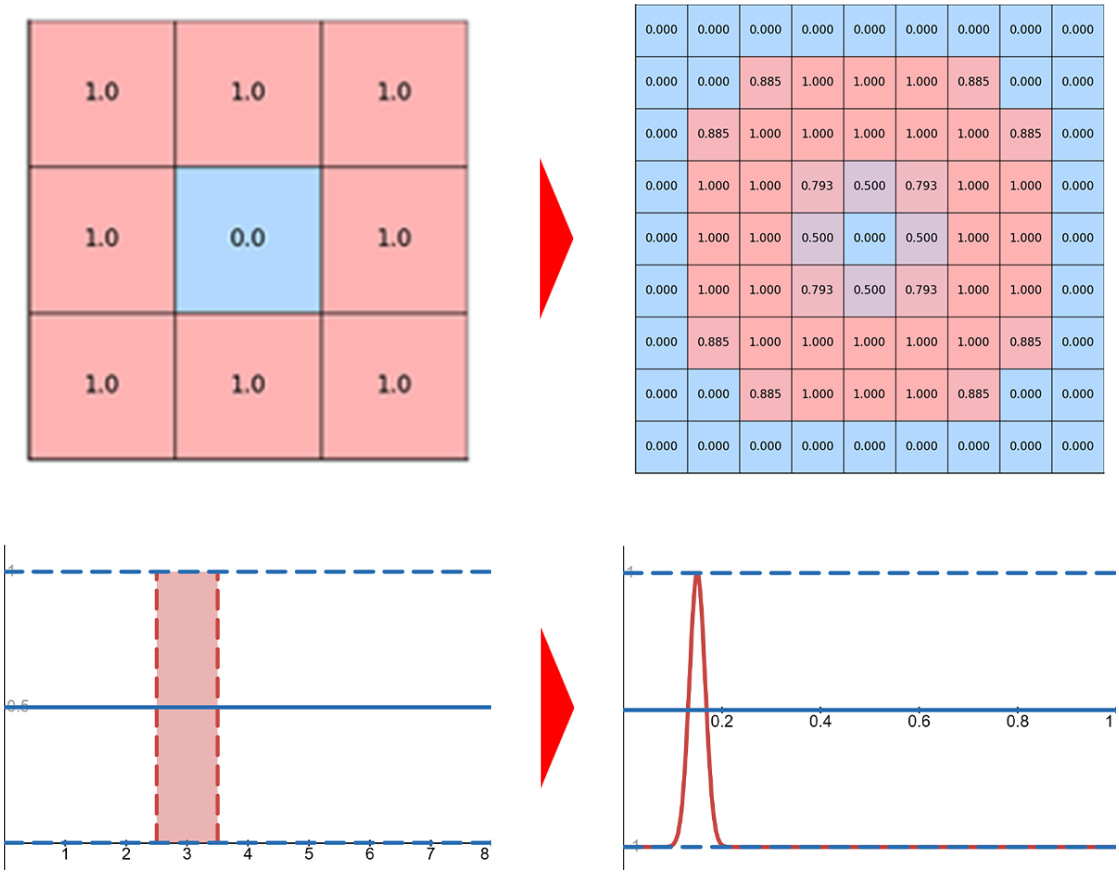


Figura 2.10: La maschera limitata di GOL diventa un esteso kernel in Lenia e la funzione gradino diventa una funzione continua.

Le operazioni diventano da:

1. **Analisi del' intorno.**
2. **Funzione di attivazione.**

A:

1. **Convoluzione:** calcolare la matrice risultato della convoluzione della griglia del mondo e il kernel maschera.
2. **Funzione di crescita:** incrementare o decrementare lo stato usando la funzione di crescita.

Il risultato è una simulazione più interessante da guardare, le strutture che si creano più organiche e fluide.

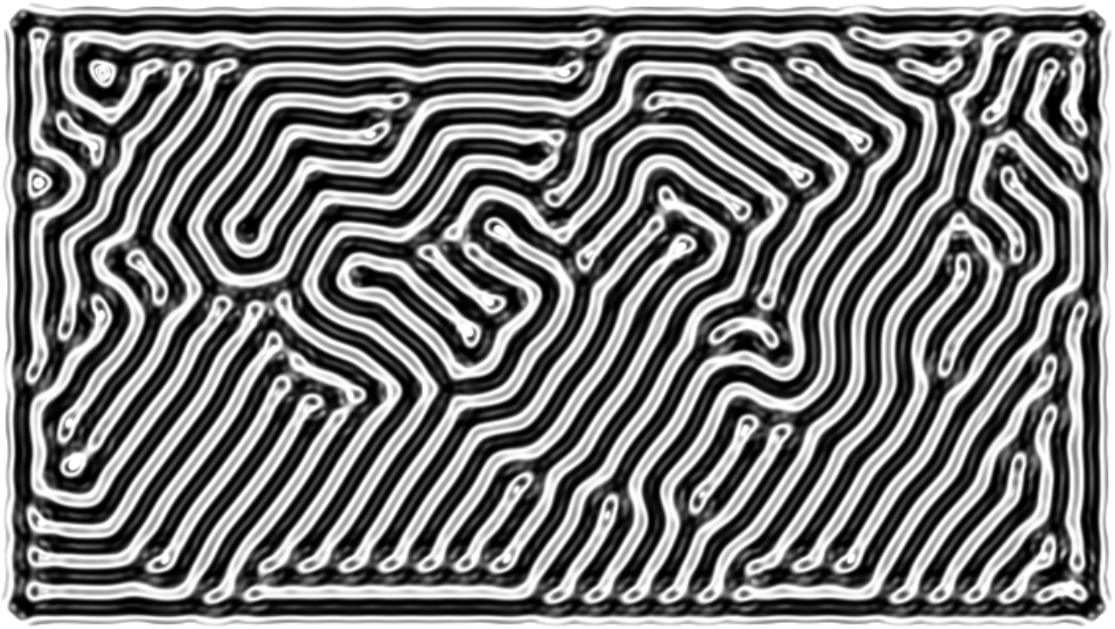


Figura 2.11: `https://www.shadertoy.com/view/sdsyWN`

2.4.3 Kernel dimensione e struttura

A differenza di Game of Life che usa una maschera a pesi 0 o 1, Lenia adotta un kernel convolutivo a valori reali di dimensione maggiore, che permette di modellare interazioni su una scala più ampia e graduale. Questa transizione da un'analisi discreta a una continua consente a Lenia di produrre strutture con movimento fluido in contrapposizione all'evoluzione a scatti tipica di GOL.
Il kernel può avere dimensioni e distribuzioni di pesi qualsiasi anche se distribuzioni sbilanciate portano a uno spostamento complessivo della materia costante che rende difficile l'emergere di comportamenti interessanti, è per questo che Bert Wang Chan[1] usa kernel formati da anelli concentrici nei suoi studi.

2.4.4 Funzione di Crescita.

$$G(u, \mu, \sigma) = 2 \exp\left(-\frac{(u-\mu)^2}{2\sigma^2}\right) - 1 \tag{2.1}$$

La funzione di crescita non indica lo stato finale ma dà un informazione di andamento, Il nuovo parametro dt viene usato per manifestare questo andamento.

Per trovare il prossimo stato occorre quindi sommare allo stato di partenza un incremento composto dalla funzione di crescita pesata dal parametro dt.

$$a^{t+dt} = [a^t + dt \cdot G(u, \mu, \sigma)]_0^1 \tag{2.2}$$

dove

$a \in [0,1]$	valore del pixel
$u \in [0,1]$	densità locale (valore risultato dell'analisi dell'intorno)
$\mu \in [0,1]$	posizione del picco della crescita
$\sigma > 0$	larghezza (deviazione standard)

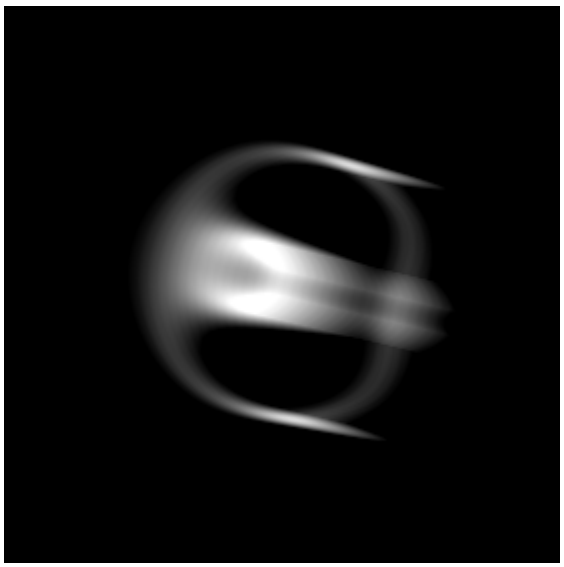


Figura 2.12: Orbium Unicaudatus: $\mu = 0.15$, $\sigma 0.017$.
`https://www.shadertoy.com/view/7ls3z7`

2.4.5 Convoluzione e accelerazione hardware GPU

L'operazione di calcolo dell'intorno per ogni punto dell'immagine è interpretabile come il calcolo di una convoluzione tra la matrice del mondo e il kernel maschera K. La matrice risultante è una matrice che per ogni pixel indica il suo valore d'intorno e quindi la funzione che determina il susseguirsi delle iterazioni di Lenia è:

$$A^{t+dt} = \Big[\, A^t + dt \cdot G\Big(\mathbf{K} * A^t\Big) \,\Big]_0^1 \tag{2.3}$$

dove

$A \in [0,1]$	matrice mondo
$\mathbf{K}$	kernel di convoluzione normalizzato.

La convoluzione è un processo facilmente parallelizzabile e quindi la GPU è capace di processare matrici e kernel grandi in tempo reale. Il processo può essere ulteriormente ottimizzato procedendo nel dominio delle frequenze, la convoluzione, infatti, diventa un prodotto tra matrici.

2.5 Conservazione della massa

2.5.1 Fragilità di Lenia classica

Nelle simulazioni precedenti un problema sussiste nella maggior parte dei casi. il mondo finisce dopo un numero finito di step per saturarsi di pattern o svuortarsi completamente. Bert Wang-Chak Chan ha catalogato le specie che riescono a mantenere una struttura che non cade in questi due fenomeni estremi[1].

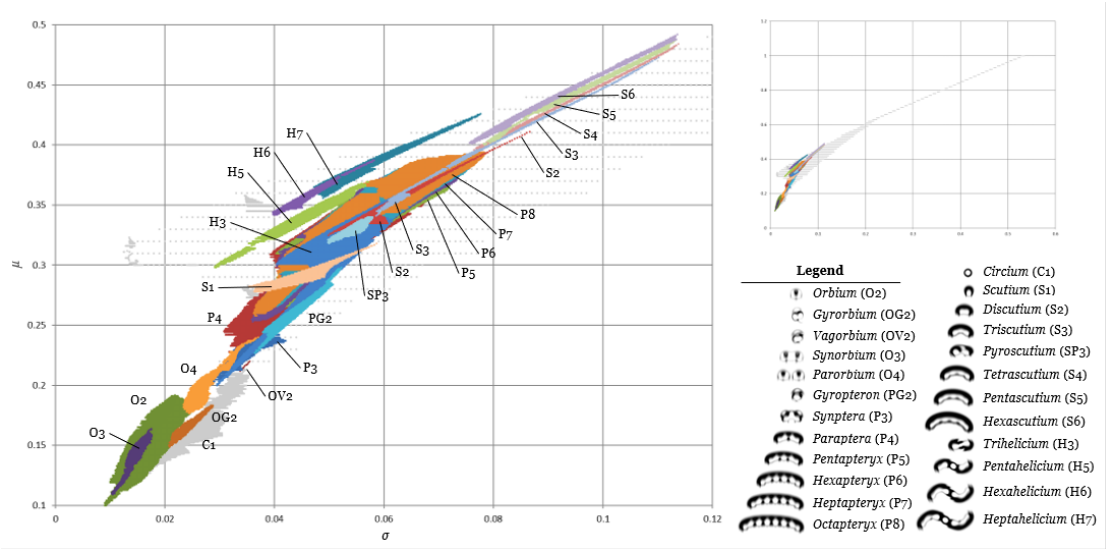


Figura 2.13: Mappa delle specie in Lenia.[1]

"Lenia, a family of cellular automata (CA) has attracted a lot of attention because of the wide diversity of self-organizing patterns it can generate. Among those, some spatially localized patterns (SLPs) resemble life-like artificial creatures and display complex behaviors. However, those creatures are found in only a small subspace of the Lenia parameter space and are not trivial to discover, necessitating advanced search algorithms."[2].
Un modo effettivo per evitare il collasso della simulazione è conservare la massa globale dell' ambiente di simulazione usando la convoluzione per creare una mappa di flusso che indica dove la massa si deve spostare ad ogni iterazione. Questo

procedimento viene approfondito nel paper Flow Lenia publicato nel 2023.
Il risultato della funzione di crescita della convoluzione che in lenia rappresentava la matrice degli incrementi viene usata in Flow Lenia per creare un campo vettoriale. Questo campo vettoriale viene modificato dalla matrice del mondo in modo da evitare un collasso, simulando quindi l'incomprimibilità della materia.

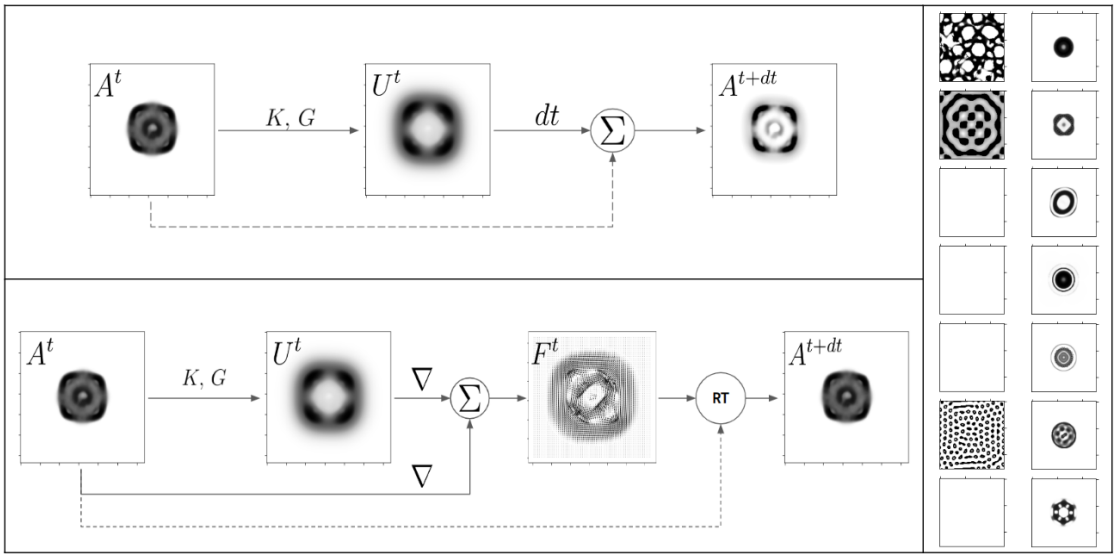


Figura 2.14: Simulazione Lenia / FlowLenia.[2]

2.5.2 Matrice di affinità (U^t)

$$U^t = \Big[\, dt \cdot G\Big(K * A^t\Big) \,\Big]_0^1 \tag{2.4}$$

dove

G	funzione di crescita.
K	kernel di convoluzione normalizzato.
$A^t \in [0,1]$	matrice mondo.

La matrice di affinità che in Lenia originale veniva direttamente usata per trovare lo stato successivo della simulazione ora viene interpretata come una matrice di "affinità" o "attrazione" le aree dal valore più elevato saranno le aree che avranno più attrattiva nello spostamento della materia.

2.5.3 Matrice di flusso (F^t) e α

$$F^t = (1 - \alpha^t)\nabla U^t - \alpha^t \nabla A^t \tag{2.5}$$

dove

α	peso per regolare l'importanza delle due forze.
∇U_i^t	gradiente della matrice di affinità.
∇A^t	gradiente della matrice mondo.

Per il calcolo del flusso vengono combinati due campi vettoriali, uno ottenuto facendo il gradiente della matrice di affinità e uno facendo il gradiente della matrice mondo.
La matrice di flusso è la somma pesata di due forze: una che spinge la materia verso punti di maggior affinità ∇U e una che la spinge lontano dalle concentrazioni di materia $-\nabla A$.
Queste due forze sono regolate dal fattore α che a sua volta è il risultato di questa funzione:

$$\alpha^t = \left[\left(\frac{A^t}{\theta_A}\right)^n\right]_0^1$$

dove

A^t	intensità del pixel t.
θ_A	limite di massa.
n	gamma.

Da questa formulazione emergono i due iperparametri principali: θ_A e n. In un mondo di valori A^t da 0 a 1 θ_A indica la massima massa che può raggiungere un pixel prima che la forza repulsiva $-\nabla A$ vinca.
Infatti impostando l'iperparametro θ_A a 0,5 qualsiasi cella che supera questo valore avrà un $\alpha = 1$ e quindi un flusso corrispondente caratterizzato solo da $-\nabla A$.
Liperparametro n indica il gamma della curva che definisce α e quindi quanto ripido è il cambiamento di prevalenza delle due forze.

2.5.4 Reintegration Tracking (RT)

L' ultimo stadio nel calcolo dello step di simulazione è usare la matrice di flusso per ridistribuire la materia, per questo usiamo una tecnica chiamata reintegration tracking.

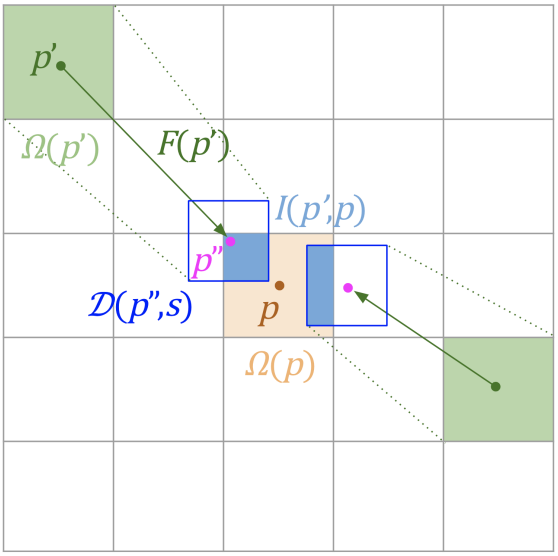


Figura 2.15: Reintegration Tracking.[2]

Questa tecnica è stata proposta da Moroz [4] come un modo di conservare la massa nelle simulazioni di fluidi, ideata originariamente per ovviare al problema della perdita di massa durante le sovrapposizioni di particelle tipici di simulazioni particellari usando dei metodi su griglia. Quando in Flow Lenia si sposta la materia da una cella a un'altra, si parte sempre da una cella con coordinate intere, ma il calcolo della destinazione avviene in uno spazio continuo, con coordinate reali.
La materia che viene spostata da un vettore che non cade in un indice intero non viene semplicemente aggiunta al pixel più vicino ma si usa una funzione di distribuzione, tipicamente un quadrato o una gaussiana, che sparge la materia in tutte i pixel intersecati.

Temperatura

Indica l'ampiezza della distribuzione o la casualità del flusso. Aumentandola non solo si ottiengono degli effetti più organici e "liquidi" ma simula anche il fenomeno fisico particellare del moto browniano.

2.5.5 Vantaggi Grafici

Le strutture che prima non avevano il tempo di emergere perchè la simulazione collassava ora riescono ad esprimersi. La conservazione della massa ha creato un ambiente simulativo più controllabile dove è possibile avere tutte le combinazioni della funzione di crescita.

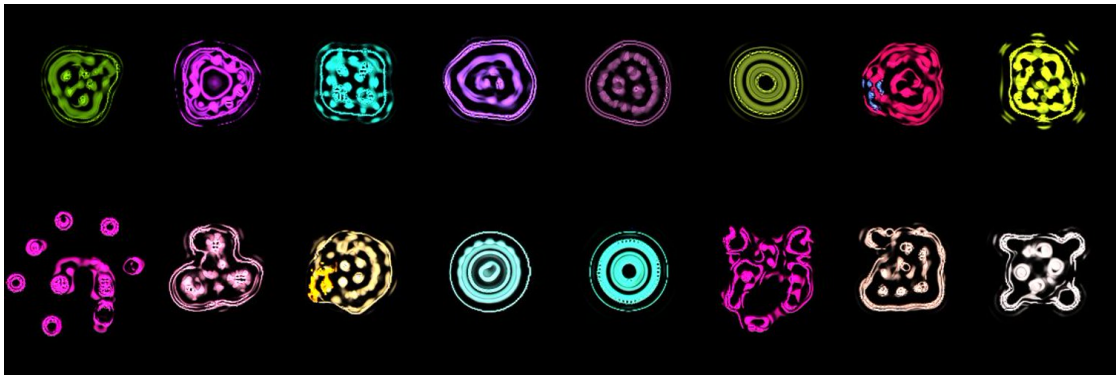
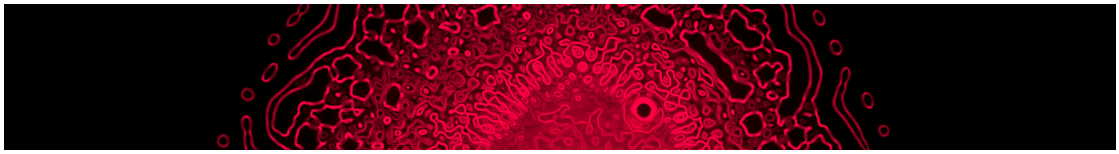


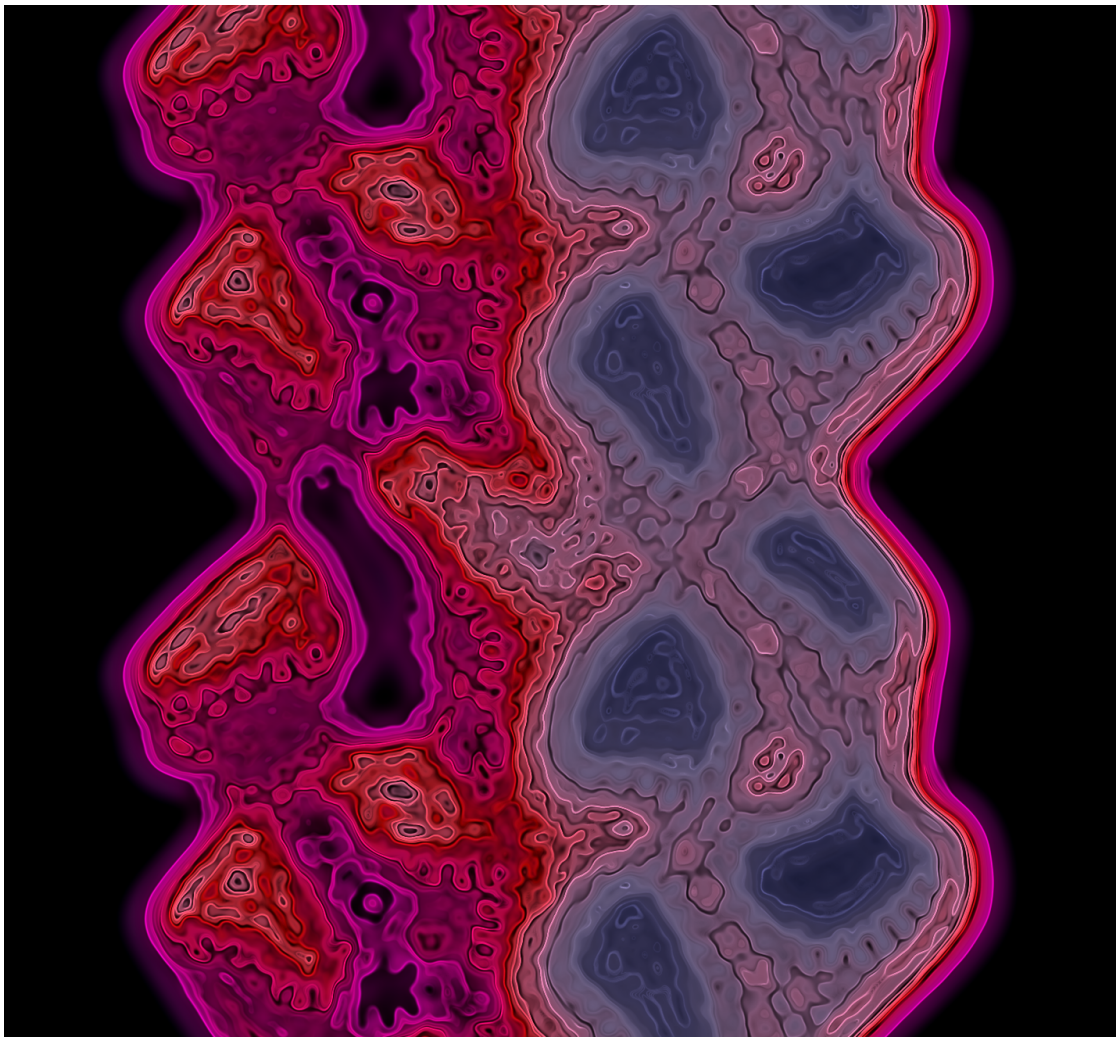
Figura 2.16: Specie in FlowLenia.[2]

Nonostante FlowLenia non consenta esplicitamente di avere più specie stabili all'interno della simulazione, questo è possibile inserendo la multi-dimensionalità come è stato fatto per Expanded Lenia[5], è possibile avere comportamenti diversi nella simulazione mantenendo una buona differenza di densità di materia.
A parità di parametri la simulazione ha comportamenti distinti quando il risultato della convoluzione cade in punti diversi della funzione di crescita, e quindi quando la concentrazione di materia circostante permette di avere un risultato nell'analisi dell' intorno diverso.
In Lenia la materia non doveva spostarsi fisicamente ma si materializzava o scompariva guidata dalla funzione di crescita, questo provoca simulazioni che si stabilizzano subito e che non permettono a zone di diversa densità di convivere stabilmente nel mondo simulato.
FlowLenia deve smistare la materia disponibile facendola scorrere tramite il gradiente della funzione di crescita e quindi se la materia inizialmente tenderà a popolare lo spazio seguendo un pattern specifico se non ne avrà abbastanza la massa rarefatta maturerà un nuovo comportamento stabile creando di fatto più comportamenti coesistenti in punti di densità diversa.



3. Tool

Lenia Tool



3.1 Introduzione

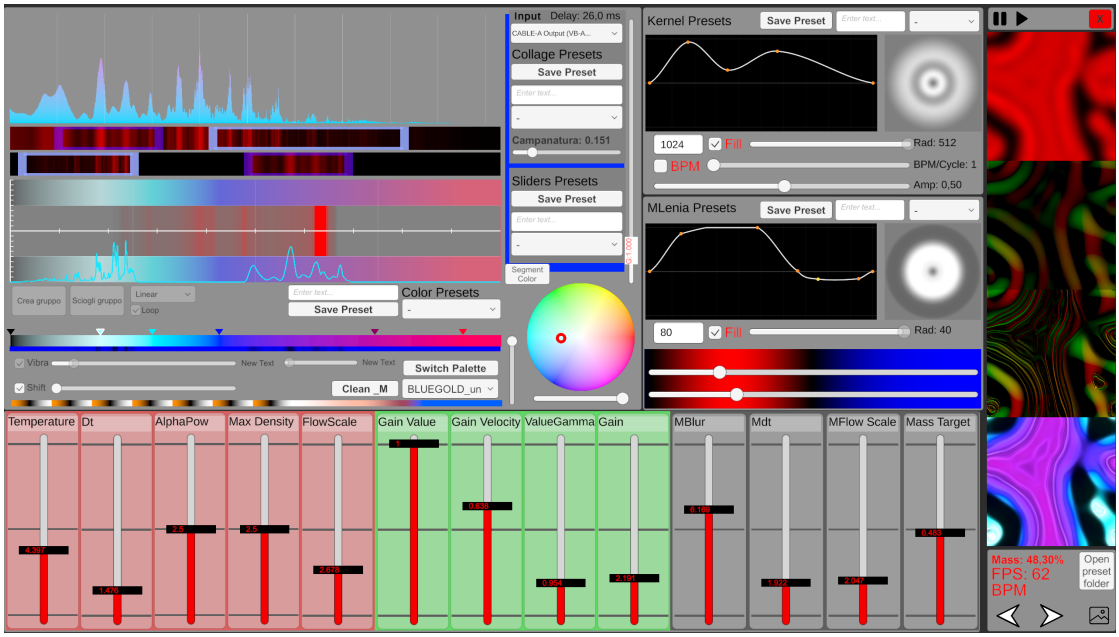
3.1.1 Menù

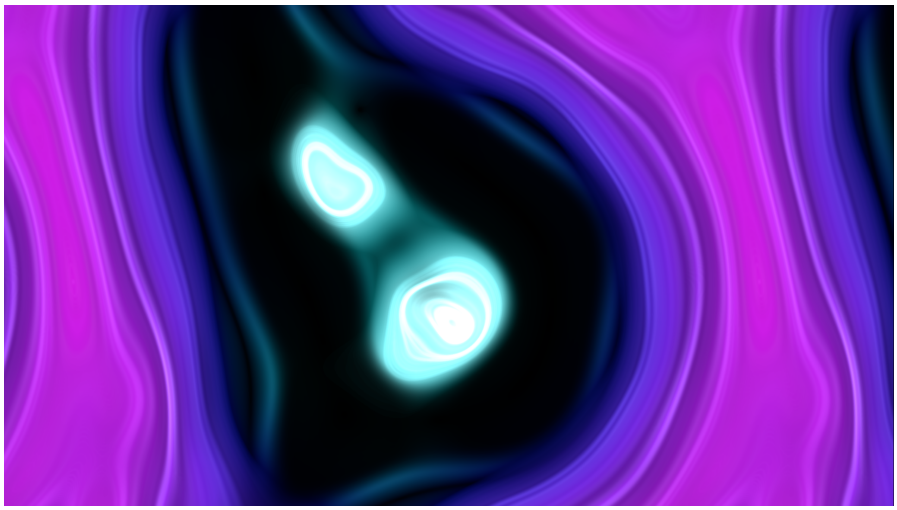
Il Tool da me proposto è una applicazione dual display e consiste in una simulazione su griglia di pixel basata sul cellular automata FlowLenia.
Al lancio dell'applicazione viene visualizzato il menù delle impostazioni in cui si devono specificare i settaggi preliminari prima di avviare la simulazione grafica e i suoi controlli.

Bisogna impostare le dimensioni della griglia del mondo di Lenia indicata in altezza e larghezza e il coefficiente di scalatura della "macro simulazione" che spiegheremo approfonditamente nel capitolo successivo dedicato alla pipeline di simulazione.
Sotto il pannello dedicato alle dimensioni del mondo di simulazione c'è un drop-down che permette di specificare il display adibito al rendering delle grafiche frutto della simulazione di Lenia Mixer mentre il display principale di windows è automaticamente assegnato al pannello di contollo.

3.1.2 Avvio e interfaccia complessiva

Una volta scelte le impostazioni corrette basta premere il pulsante "START" per iniziare il programma, a quel punto il display scelto verrà attivato e verranno assegnati i canvas agli schermi corretti riproducendo le grafiche nel display selezionato e i controlli che spiegheremo successivamente nel display principale.





3.1.3 Struttura generale

Questa applicazione permette di modulare il campo vettoriale di una simulazione FlowLenia andando a modificare nello specifico la matrice di affinità cambiando la funzione di crescita, il kernel di convoluzione e vari altri parametri usati nella combinazione e amplificazione dei campi vettoriali che compongono FL.
La modulazione delle regole della simulazione non avviene solo manualmente ma è anche assistita dall'uso dell'audio, infatti una parte fondamentale di questa applicazione è l'uso dello spettro sonoro nella personalizzazione delle regole di Lenia.
Ho deciso di trasformare lo spazio delle regole, descritto con la funzione di crescita U^t (2.4.4, una semplice funzione gaussiana con codomino [-1,1], in una look-at-table personalizzabile. D'ora in poi chiameremo questa versione campionata di questa funzione di crescita personalizzabile "GrowthLUT". Per aumentare la ricchezza degli effetti ho deciso di implementare un sistema basato su due pipeline parallele di FL con parametri diversi:

1. La **"micro-simulazione"** interagisce con il mondo in piena risoluzione e interagisce con lo spettro audio, crea gli effetti organici più definiti ma per questo non riesce a smuovere macroscopicamente la materia ma crea le imperfezioni locali.

2. La **"macro-simulazione"** interagisce con un mondo con una risoluzione minore della texture originale(di default 10 volte più piccola), questo permette di muovere grosse aree di materia insieme e quindi di avere un movimento macroscopico non raggiungibile solo con la prima simulazione.

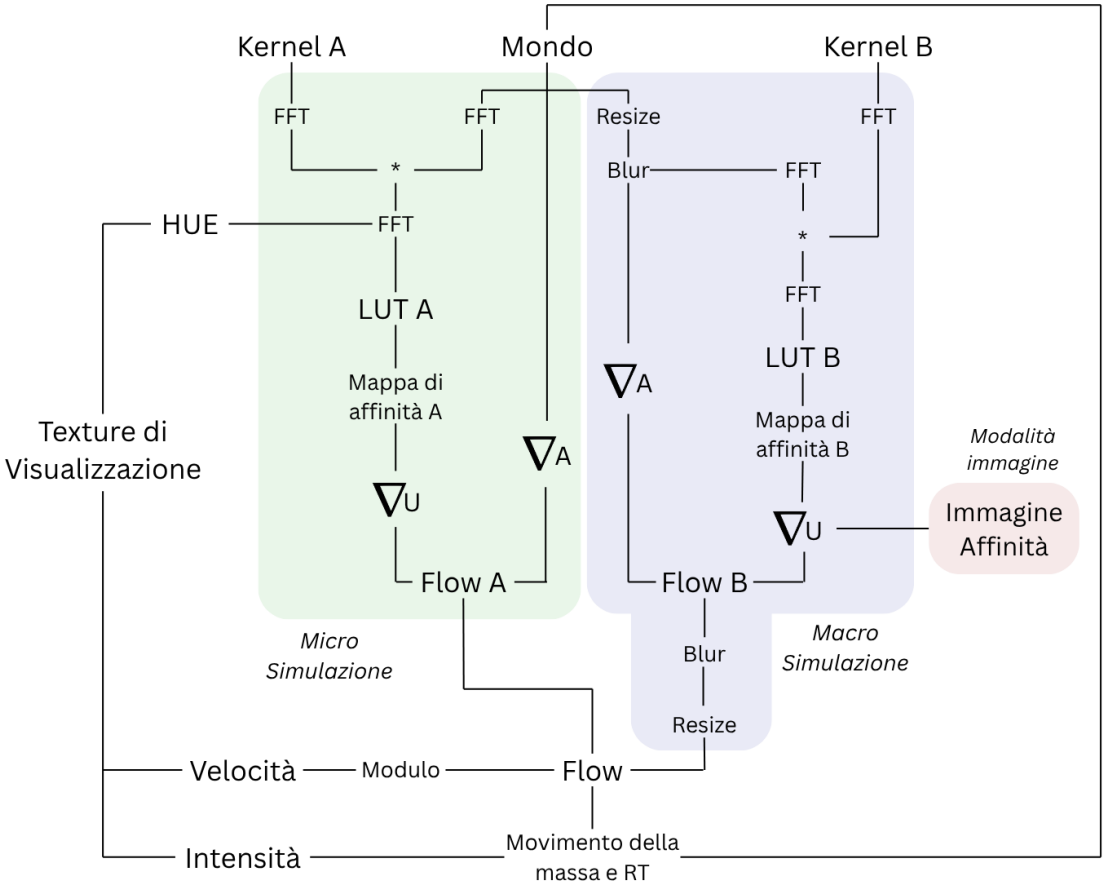
Entrambe le due simulazioni seguono la struttura tipica di FL (2.5.3) e quindi hanno dei propri kernel di convoluzione (K^t), LUT della funzione di affinità o crescita (U^t) e i parametri nella creazione della matrice di flusso (n, θ_A, $temp$).
La funzione di crescita della micro-simulazione è composta da una manipolazione dello spettro dell'audio mentre quella della macro-simulazione è creata usando la funzione di Lenia classica (2.4.4) con degli parametri per regolare la posizione e l'ampiezza della campana, quindi solo gli effetti locali sono influenzati dalla musica mentre quelli globali vanno cambiati manualmente per istigare un cambio macroscopico nel pattern.

La **macro-simulazione** può essere configurata in due modi:

1. La **modalità di default** è quella di Lenia classica e quindi segue la pipeline con convoluzione, calcolo dei gradienti e spostamento di materia ma con dei parametri diversi dalla micro-simulazione.

2. La **modalità immagine** invece permette di associare una qualsiasi immagine alla matrice di affinità della macro-simulazione, quindi, bypassando le operazioni di convoluzione e calcolo della matrice di affinità, il processo si riduce al calcolo del gradiente usando l'immagine come U^t. Questo permette di avere un flusso che fa convergere la materia nei punti più chiari dell'immagine.

3.2 Pipeline di Simulazione



3.2.1 Due Simulazioni

La simulazione consiste in due FL paralleli, il primo usa la griglia del mondo nella sua interezza a risoluzione originale, il secondo usa una sua versione ridimensionata. La prima simulazione, più definita della seconda, gestisce i comportamenti locali del flusso, crea i pattern effettivi che reagiscono alla musica.
La seconda viene usata per gestire il comportamento macroscopico della materia, è infatti una simulazione più smussata che sposta la materia gentilmente e serve per mantenere una dinamicità globale che non sarebbe raggiungibile con la prima per via della portata più ridotta del Kernel A.
I cambiamenti dei parametri della prima simulazione chiamata "micro-simulazione" modificano il comportamento dei pattern e delle grafiche ma lo fanno in maniera troppo locale, le concentrazioni macroscopiche della massa rimangono stabili e al massimo tendono a smussarsi e equilibrarsi.

La seconda simulazione chiamata "macro-simulazione" serve proprio ad avere una dinamicità macroscopica e a mantenere un alto livello di contrasto nella massa, una massa con una gamma di concentrazioni ampia permette infatti di avere più fenotipi nello stesso mondo simulato come abbiamo già spiegato nel paragrafo riguardante i vantaggi grafici di FlowLenia (2.5.5).
Oltre a seguire la pipeline classica di FL la macro-simulazione è abilitata alla configurazione "modalità immagine" che come abbiamo già accennato salta tutte le operazioni che dalla convoluzione arrivano al calcolo della mappa di affinità e usa direttamtente un immagine scelta per fare il gradiente ∇U.

3.2.2 FFT

La parte fondamentale di Lenia, l'analisi dell'intorno, la convoluzione tra il mondo e il kernel, viene risolta più efficacemente nel dominio delle frequenze con un semplice prodotto degli spettri.
La convoluzione, nonostante la sua semplicità di implementazione e la sua facile parallelizzazione, risulta poco ottimizzata con mondi o kernel di grande dimensione. Il procedimento usando la trasformata non è così facile da implementare ma fortunatamente esistono numerose soluzioni disponibili online per gestire le FFT su Unity in maniera efficente, io ho usato la ComputeFFT library di MatejLou [6].
Il risultato della convoluzione viene salvato in una texture che viene usata per ricavare la tonalità di ogni punto.

3.2.3 Affinità e GrowthLUT

Come abbiamo già visto, i modi di caratterizzare la simulazione e la sua fenomenologia sono due: la struttura del kernel (l'influenza dei vicini) e il modo in cui viene mappato il risultato della convoluzione.
La mappatura avviene tramite una texture float a una dimensione dai valori che vanno da -1 a 1, per la micro-simulazione viene usata la GrowthLUT che viene creata campionando lo spettro dell'audio manipolato a piacimento mentre per la macro-simulazione, se non è abilitata la modalità immagine, la LUT è creata campionando la funzione di crescita classica e parametrica di Lenia (2.4.4).
Il risultato della mappatura del risultato della convoluzione nella LUT è una matrice dei valori, chiamata matrice di affinità, che in Lenia classica viene direttamente usata per determinare la crescita per ogni pixel, nel nostro caso di FlowLenia viene usata per estrapolare il flusso di materia.
Nel caso la macro-simulazione sia impostata in modalità immagine, non avverrà nessun calcolo della convoluzione e della matrice affinità perchè quest'ultima verrà automaticamente sostituita con il canale value di un immagine scelta arbitrariamente.

3.2.4 Flusso di materia

FL usa il flusso di materia descrivere l'andamento della simulazione, in questo modo la massa è conservata e la simulazione non ha modo di collassare a zero o esplodere riempiendo il mondo.
Per trovare il campo vettoriale del flusso occorre calcolare due gradienti:

- ∇U : il gradiente della mappa di crescita.
- ∇A : il gradiente del mondo, molta convergenza in punti ad alta densità di massa.

Questi due gradienti vengono combinati seguendo la formula:

$$F = (1 - \alpha)\nabla U - \alpha \nabla A \tag{3.1}$$

Il flusso è quindi la somma pesata dal valore α del gradiente ∇U e l'opposto di ∇A.
Nella formula α è un valore arbitrale da 0 a 1 che regola il peso di entrambi i componenti.
Con α tendente a 0 predomina il gradiente ∇U e quindi la forza che spinge la massa verso i pattern tipici delle regole che abbiamo scelto con il nostro kernel e la nostra GrowthLUT, con α tendente a 1 predomina l'opposto del gradiente ∇A e quindi la forza repulsiva della materia che tenderà ad allontanarsi dalle sue alte concentrazioni.

3.2.5 Combinazione del Macro e Micro Flusso

La macro simulazione segue la stessa logica della micro-simulazione nella creazione del campo del flusso con l' eccezione che la mappa del mondo usata per la convoluzione è una versione riscalata e blurrata in modo da avere una analisi dell'intorno più generale e meno dettagliata.
La risultante mappa vettoriale Flow B è una mappa con una risoluzione diversa da quella di Flow A, quindi viene scalata e interpolata alla risoluzione originale non prima di essere nuovamente blurrata.
Infine per calcolare li flusso totale basta sommare i due flussi pesati per i rispettivi coefficienti arbitrari.

3.2.6 Movimento della Massa e Reintegration Tracking

La mappa del flusso viene infine usata per creare il nuovo stato della simulazione. In breve il contenuto del pixel (x, y) viene riposizionato nel pixel $(x + F(x, y).x, y +$

$F(x, y).y)$ e dopo viene ricalcolata la massa per ogni pixel interpolando i risultati. In questo modo è possibile calcolare il nuovo stato e anche la mappa di velocità andando ad associare al pixel di destinazione non più il valore del pixel di partenza ma il modulo del vettore spostamento.

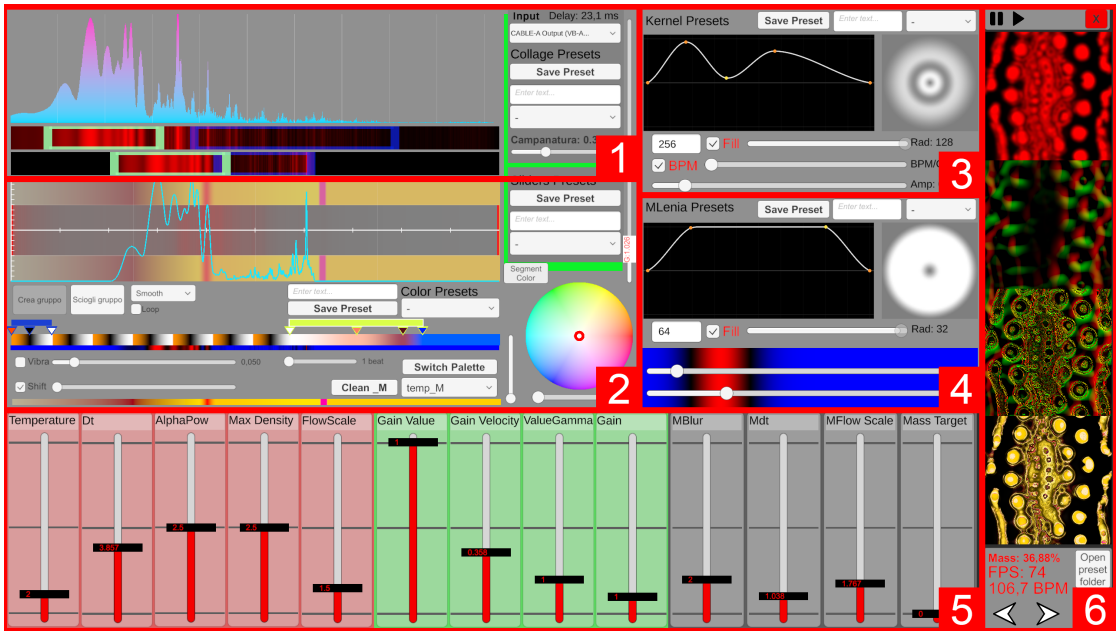
3.2.7 Shader di Simulazione

Lo shader di simulazione combina le texture aggiornate alla nuova iterazione in una texture HSV e poi la elabora in una sua copia RGBa pronta alla visualizzazione
Le texture usate sono tre:

- Intensity : massa della simulazione.
- HUE : risultato della convoluzione.
- Velocity : velocità del flusso.

La Intensity map è proprio la texture dello stato corrente della simulazione e insieme alla Velocity viene usata nello shader per modulare la luminosità per ogni pixel. L'HUE map è una matrice float di valori da 0 a 1 che serve per determinare la tonalità e la saturazione del colore di ogni pixel, viene usata per individuare un una texture custom 1D RGB il colore che ogni pixel deve assumere. La HUE map è presa direttamente dal risultato della convoluzione della micro-simulazione, in questo modo i pixel con colori simili avranno comportamenti simili perchè i risultati delle convoluzioni ricadranno in punti vicini della LUT di affinità.
La velocity map è ricavata dal modulo della matrice di flusso finale e indica la velocità di ogni particella di massa, anche quest'ultima mappa viene usata per modulare l'intensità luminosa dei pixel.

3.3 Pannello di Controllo



Il pannello di controllo si divide in sei aree di lavoro che servono a gestire la personalizzazione di parti diverse dell' applicazione:

1. Impostazioni audio e collage dello spettro.
2. Palette colori.
3. Editor del kernel micro-simulazione.
4. Editor per la macro-simulazione.
5. Sliders micro-simulazione, color, macro-simulazione.
6. Pannello laterale destro per i controlli generali, le preview, monitoring e ImageMode.

TapTempo

Una feature nascosta all'interfaccia è il calcolo dei BPM, i BPM sono usati in diverse sezioni per modulare a tempo parametri. Per calcolare i BPM il mio programma usa una soluzione presente in molte DAW, il "tap tempo" o il BPM counter, cliccando a tempo la barra spaziatrice il programma farà una media delle distanze tra un battito e l'altro e ricaverà dei BPM abbastanza fedeli.
Il mio tap tempo aggiusta anche la fase ad ogni tap, andando a sincronizzare i BPM con l'ultimo tap registrato.

3.3.1 Preset Manager

Ogni area di lavoro ha la sua sezione adibita al salvataggio e caricamento di settaggi delle sessioni di lavoro, tutti i parametri sono memorizzabili e riutilizzabili velocemente scegliendo nelle sezioni apposite i preset che più ci piacciono.
La struttura è la stessa per ogni area di lavoro, nella sezione chiamata "*nomeArea* Presets" ci sono:

- Un input di testo.
- Un menù a tendina.
- Un pulsante per il salvataggio.

Se si vogliono salvare i valori dei sattaggi usati al momento, occorrerà scrivere un nome nella casella di testo e premere il pulsante "Save Preset". Se si vuole caricare un preset precedentemente salvato, si può, ocercarlo e selezionarlo nel menù a tendina dei preset, oppure si possono filtrare gli elementi del munù a tendina scrivendo nell'input di testo.

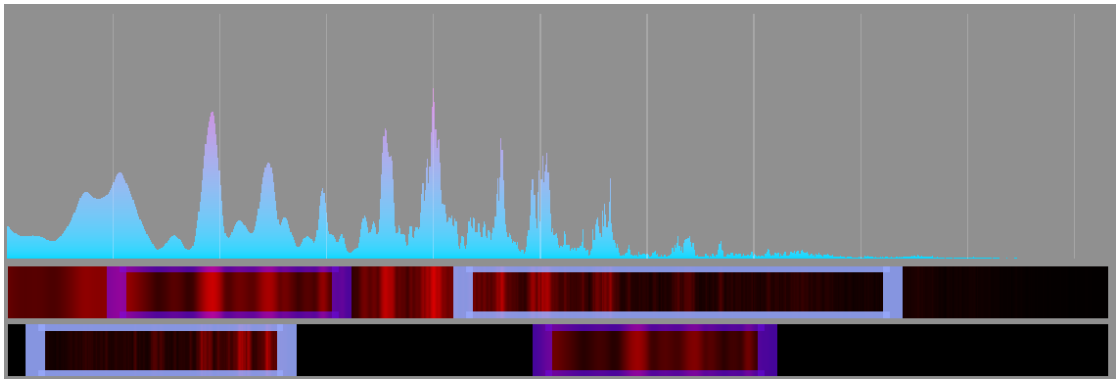
3.3.2 Manipolazione dello spettro

Il primo pannello che presenteremo è quello adibito alla trasformazione dello spettro sonoro in GrowthLUT per il calcolo dell'affinity map. L'approccio usato è quello del collage, la texture risultante è un insieme di ritagli dai volumi e ampiezze arbitrarie dello spettro.
Gli elementi che gestiscono questa feature sono posizionati nel pannello principale in alto a sinistra.

Equalizzatore e scelta dell'input

Questa applicazione è in grado di leggere qualsiasi ingresso audio del computer, il canale di input è selezionabile da un menù a tendina posto in cima alla colonna centrale.
Dall'audio viene calcolato il suo spettro e rappresentato in scala logaritmica da 20Hz a 20kHz in un grafico a barre.

Collage dello Spettro



Il meccanismo del collage consiste nell' isolare vari intervalli di frequenza e incollarli e stretcharli in qualsiasi punto della GrowthLUT si voglia.
Questa operazione si svolge su due barre, la prima rappresenta la mappa dell'intensità dello spettro dell'audio in entrata e da qui si specificano gli intervalli che si desiderano utilizzare, la seconda rappresenta la mappatura risultante e quindi dove si potrà decidere la posizione di output dell'intensità di queste frequenze.
Tenendo premuto ctrl e trascinando con il mouse sinistro sulla prima barra si può creare una finestra di ampiezza a piacere che poi potrà essere modificata spostando gli estremi o spostata per intero trascinando l'intero segmento a destra e sinistra.
Alla creazione di una finestra verrà creata la sua corrispettiva di output automaticamente nella stessa posizione sulla seconda barra, anch'essa potrà essere modificata analogamente alla prima con l'eccezione che, se si cambia l'ampiezza delle finestre, nella prima barra equivale al comprendere un intervallo diverso di frequenze, mentre, se modificheremo l'ampiezza delle finestre nella seconda esse saranno le stesse della rispettiva prima barra ma stirate o compresse.
Cliccando con ctrl + Mouse Destro si eliminano le finestre e, se alteriamo un segmento trascinando un suo estremo oltre l'altro, il segmento verrà invertito, stampando una versione invertita dello spettro.

Pannello di controllo centrale

A destra del pannello principale, al centro dell'interfaccia, c'è un insieme di comandi incolonnati, la prima metà è adibita proprio a questa fase del workflow.
In ordine, dopo il menù a tendina per la scelta del canale audio di ingresso, ci sono dei comandi che servono a salvare e caricare i preset di questa sezione. Sotto "Collage Presets" c'è il bottone "Save Preset", l'inserimento del testo e il menù a tendina, il loro funzionamento è stato già descritto nella sezione dedicata ai preset (3.3.1).
Gli altri due elementi usati in questa area di lavoro sono: lo slider di campanatura e lo slider del gain. Questi due slider gestiscono, per ogni segmento selezionato, di regolare la finestratura della sezione e la potenza del segnale nell'output.

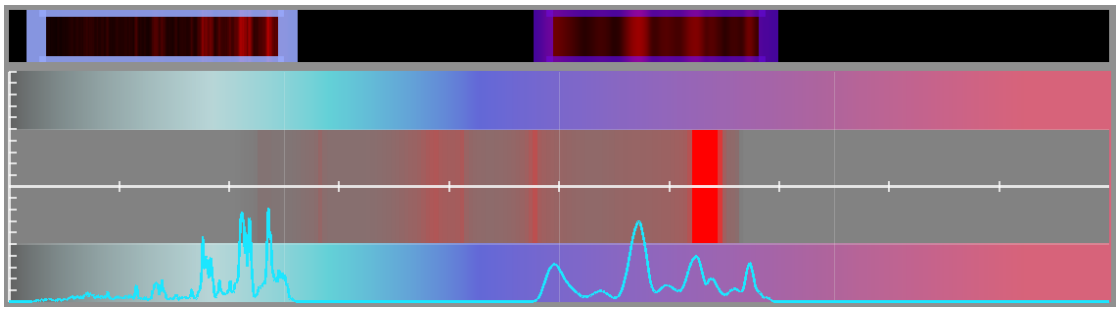
Esempio di utilizzo

Lavoro con un brano techno in cui c'è un basso che da il ritmo e un suono acuto che suona la melodia, creo due segmenti, uno che inizia da 20hz a all'incirca metà spettro e uno che da poco dopo arriva ai 20khz.
Ora che ho diviso lo spettro in queste due componenti indipendenti posso equalizzare in modo che anche se il basso è molto più energico dello strumento acuto entrambi influenzano la GrowthLUT della simulazione in maniera ricca e definita, per fare ciò abbasso il gain della finestra che copre i bassi e alzo il gain della finestra che copre gli alti, in questo modo entrambe le curve dello spettro saranno nell'intervallo della GrowthLUT senza che ci sia clipping o livelli impercettibili.

3.3.3 Creazione della palette colori

In questa sezione ci sono i controlli per la creazione della palette usata per colorare la simulazione.
Come abbiamo visto nell'analisi della pipeline di simulazione, il risultato della convoluzione viene salvato come HUE map che è una texture di valori float che vanno da 0 a 1.
La HUE map, che se mappata con la funzione di crescita o GrowthLUT crea la mappa di affinità, viene anche usata per determinare il colore dei pixel andando a trovare, usando il valore di ogni pixel, il colore corrispondente in una palette creata arbitrariamente.
Questa sezione è adibita alla creazione di questa palette e alla gestione dei suoi presets.

Visualizzazione della GrowthLUT risultato del collage



Subito sotto la parte dedicata al collage dello spettro c'è la rappresentazione in un grafico del suo output, quindi della GrowthLUT.
In trasparenza, una barra che taglia il grafico ci informa, tramite una texture semitrasparente, sulla distribuzione dei risultati della convoluzione dei pixel di simulazione (praticamente l'istogramma della texture HUE map) e quindi possiamo spostare i segmenti del nostro collage di spettri in modo che la GrowthLUT vada a sollecitare più o meno pixel.
Le bande del grafico sopra e sotto la banda dell'istogramma in trasparenza è possibile vedere la palette risultante, questa visualizzazione del grafico della LUT, dell'istogramma della HUE map e la color palette è utilissima per trovare le concentrazioni di materia reagente con l'istogramma e adattare la GrowthLUT e la palette di conseguenza.

Tool per la creazione della palette

Sotto questo grafico c'è il tool della gestione della palette. Questa sezione consiste in:

1. Sezione per le impostazioni dei gruppi.
2. Sezione per la gestione dei preset.
3. Barra della palette.
4. Impostazioni per l'animazione della palette.
5. Seconda palette con la sua selezione dei preset e meccanismo di switch.
6. HS(V) pie per la scelta del colore con slider per la luminanza.

Creare nodi colorati

Il componente fondamentale di questo tool è la barra della palette, subito sotto i pulsanti di gestione dei gruppi e dei preset c'è una barra colorata che all'avvio si presenta con due triangolini ribaltati ai suoi lati estremi.
Questi simboli indicano dei nodi, i nodi sono dei punti sulla texture della palette dove abbiamo settato dei colori, i punti tra due nodi colorati in maniera diversa avranno colori interpolati tra quelli dei nodi che li delimitano.
Cliccando su un qualsiasi punto con ctrl + Mouse Sinistro si crea un nodo che verrà automaticamente selezionato alla creazione ed è possibile deciderne il colore usando il grafico HS pie adiacente, i nodi possono essere spostati trascinandoli premendo il tasto sinistro sul loro indicatore triangolare.

Gruppi di nodi

Con shift + Mouse Sinistro si modifica la selezione aggiungendo o deselezionando nodi che poi possono essere raggruppati cliccando il pulsante "Crea gruppo".
I gruppi consentono di spostare e distanziare insiemi di nodi, alla creazione il gruppo si presenta come una barra superiore che copre tutti i nodi e quindi si

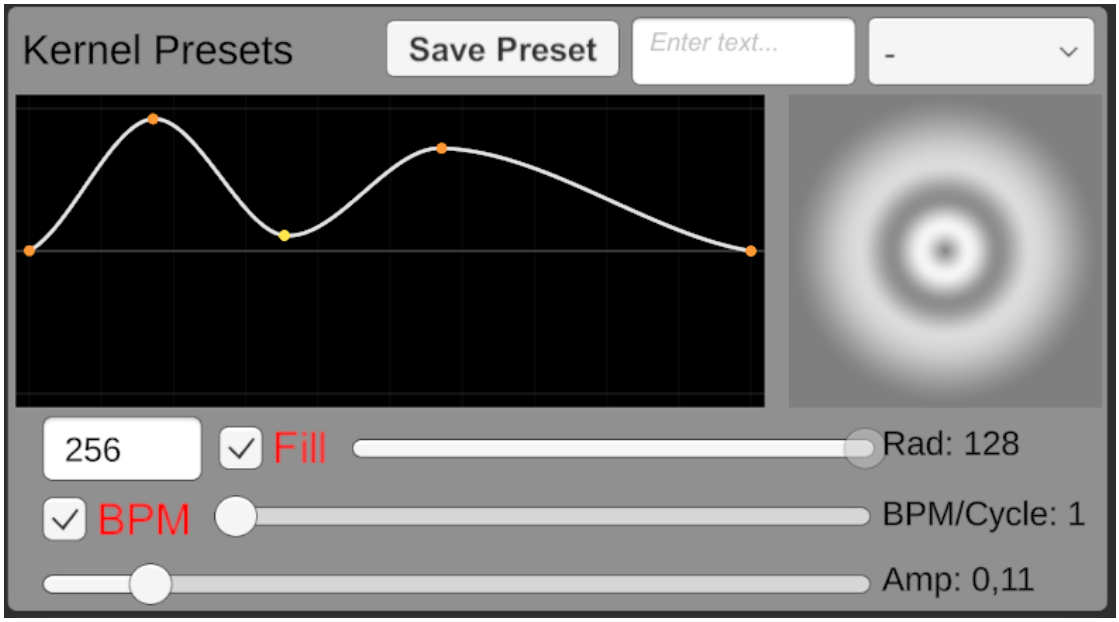
estende dal primo nodo all'ultimo, il gruppo può essere spostato spostando la sua barra e può essere ristretto o espanso andando a muovere i bordi della barra.
Le altre opzioni permettono di scegliere il modo di interpolazione tra i nodi del gruppo, si può scegliere tra "Costant", "Linear"e "Smooth" e si può scegliere se abilitare il "Loop". Se abilitato, il loop ripete il pattern dei nodi del gruppo di continuo da destra verso sinistra fino a quando incontra un altro nodo che non appartiene al gruppo.

Animazione Palette

Sotto la barra principale ci sono delle impostazioni che permettono di modificare la palette automaticamente.
Il primo effetto animato che si può introdurre è il vibrato, questo effetto assegnabile ad un gruppo fa vibrare i suoi nodi attorno al suo centro. L'ampiezza di vibrazione è regolata da uno slider mentre la frequenza e la fase dell'oscillazione sono gestite dal TapTempo che abbiamo già introdotto nel suo paragrafo(3.3). Un secondo slider gestisce i BPM/cicli, sceglibili tra 1, 2, 4, 8 si definiscono quanti BPM occorrono per compiere un ciclo di oscillazione.
Infine sotto c'è lo slider dello shift automatico, abilitando il toggle e impostando un valore sullo slider l'intera palette scorrerà a velocità costante da sinistra a destra o da destra a sinistra.

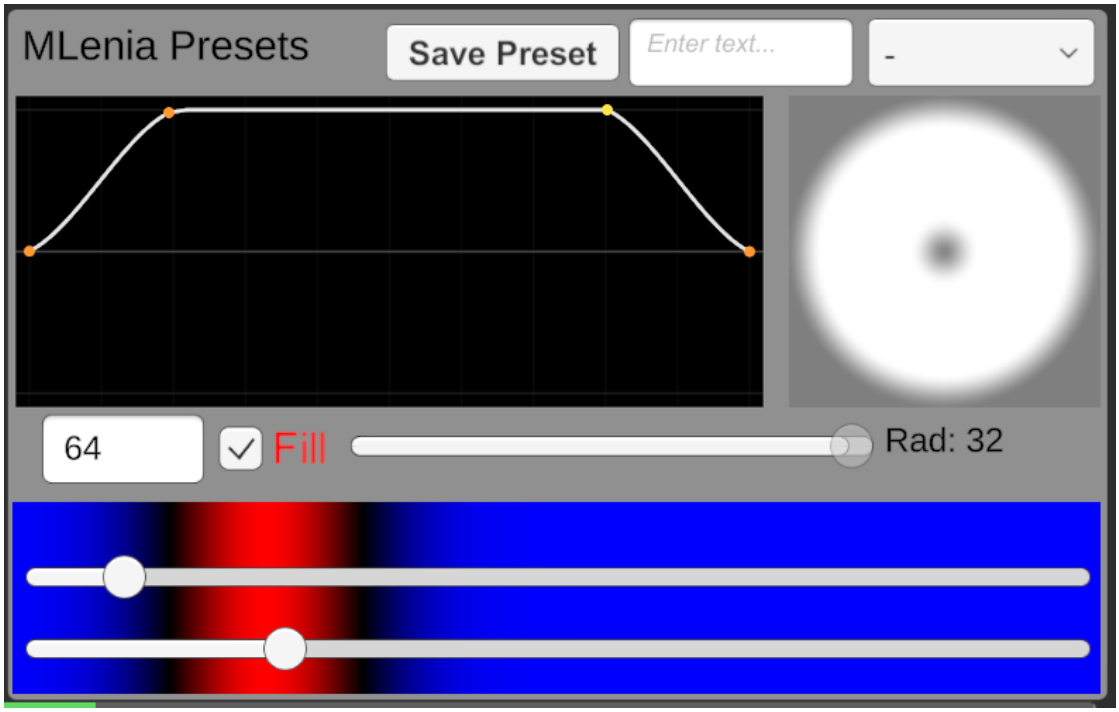
Preset palette e seconda palette

come ogni sezione anche questa ha i suoi preset, i comandi in alto a destra gestiscono proprio questo. Il funzionamento è analogo agli altri preset descritti in 3.3.1, i preset salvano completamente tutto quello che riguarda la palette principale: nodi, colori, gruppi, impostazioni e animazioni.
Sotto alle impostazioni delle animazioni c'è un'altra barra, la seconda palette. Questa palette non può essere modificata, può essere aggiornata da un preset per essere alternata con la palette principale.
La palette finale, quella che viene usata nella simulazione è la media pesata da uno slider delle due palette, lo slider che regola questa influenza è posto a destra delle due barre e un bottone "Switch Palette" permette di cambiare la palette principale con la secondaria e viceversa.
Se una palette non salvata come preset viene scambiata con la secondaria, essa viene salvata come preset momentaneo "$_M$" e caricata nella seconda palette, mentre la seconda palette che è per costruzione un preset già esistente viene semplicemente caricata nella primaria.
Lo slider dello switch segue la palette scelta anche dopo uno scambio di palette, se lo slider puntava alla palette 1 al cambio punterà alla palette 2.



3.3.4 Micro-Kernel Editor

Questo pannello è posizionato in alto a destra e serve per la creazione del kernel necessario per la micro-simulazione.
Il kernel è descrivibile definendo il suo semiprofilo con una spline in un editor a nodi, cliccando con il tasto sinistro inseriamo un nodo che avrà una interpolazione smooth in automatico, facendo doppio click su un nodo si può cambiare la smussatura tra linear e smooth.
Accanto all'editor c'è la visualizzazione della maschera in una texture quadrata.
Sotto l' editor di curve ci sono le impostazioni di dimensione e raggio con un ulteriore checker per riempire automaticamente il riquadro impostando il raggio come metà della dimensione.
Inoltre ci sono ulteriori impostazioni per far oscillare il raggio a ritmo di musica, Spuntando "BPM" il raggio del kernel fluttuerà di una sua frazione, scelta dallo slider dell'amplitude.
La frequenza e la fase di oscillazione sono quelle calcolate dal TapTempo(3.3), un ulteriore slider ci permette di specificare i BPM/Cycle, cioè quanti battiti far durare un ciclo di oscillazione.



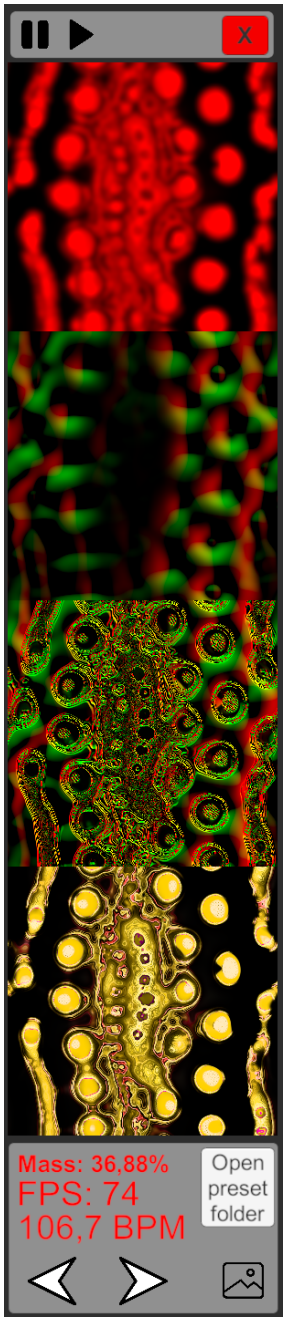
3.3.5 Macro-Sim Editor

Sotto l'editor del kernel della micro-simulazione c'è il Pannello per la macro-simulazione.
Il meccanismo di creazione del kernel è lo stesso editor di una spline ma mancano le impostazioni per l'oscillazione.
La GrowthLUT della macro simulazione è diversa da quella creata da collage e non dipende dallo spettro dell'audio, è creata da due slider che regolano μ e σ per creare la celebre funzione di crescita di Lenia classica(2.4.4 che viene visualizzata in una texture sottostante.

$$G(u, \mu, \sigma) = 2 \exp\left(-\frac{(u-\mu)^2}{2\sigma^2}\right) - 1 \tag{3.2}$$

μ descrive lo shift della campana mentre σ è il parametro di campanatura.

3.3.6 Pannello laterale: controlli e preview



Alla estrema destra è collocato un pannello che occupa tutta la altezza dell'interfaccia.
In questo pannello sono presenti:

- Controlli Stop e Play.
- Exit button.
- Preview di monitoring.
- statistiche.
- Link alla cartella dei preset.
- controlli dell'ImageMode.

Partendo dall'alto troviamo i tipici controlli per fermare o riprendere la simulazione e il pulsante per uscire dall'applicazione.
Sotto ci sono delle preview utili per monitorare la simulazione e in ordine incontriamo:

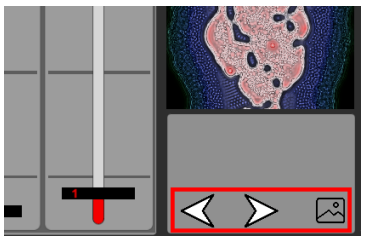
1. Mappa di affinità della macro-simulazione: sia quella creata da Lenia, sia quella usata per l'ImageMode.
2. La matrice del flusso della macro-simulazione.
3. Il flusso totale.
4. Una preview della simulazione finale.

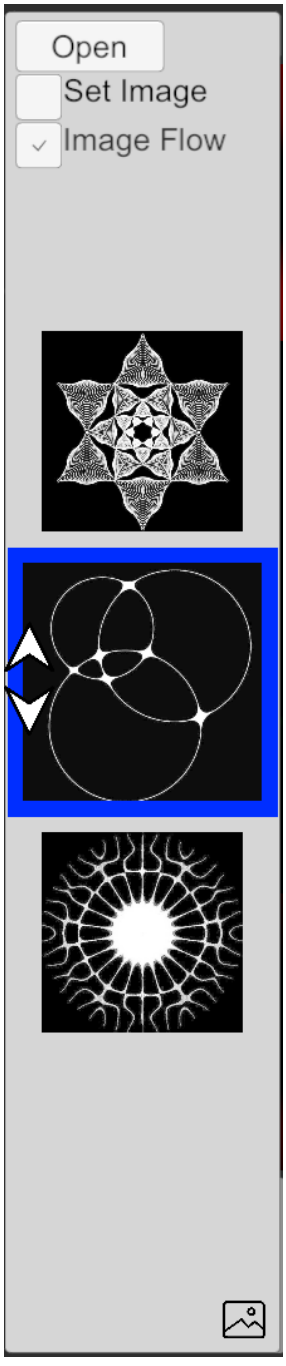
L'ultimo piccolo pannello ha delle informazioni generali della simulazione come la massa in percentuale, il numero di frame al secondo e i BPM calcolati dal TapTempo, accanto un bottone, se cliccato, rimanda all'indirizzo di memoria dove sono salvati i preset.
Infine ci sono i comandi per l'ImageMode che introdurremo meglio nella prossima sezione.

3.3.7 Image Mode

La modalità immagine della macro-simulazione è incredibilmente efficace nel controllo del flusso, la modalità standard che crea la matrice di affinità con le operazioni tipiche di lenia riesce a smuovere il mondo simulato in grossi pattern di FL ma non consente di controllarne gli spostamenti, image mode è ideato proprio per avere un maggior controllo della struttura dei pattern macroscopici.

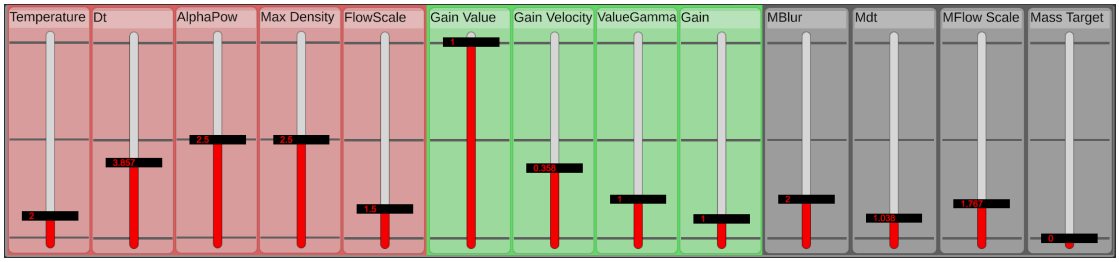
In basso a sinistra del pannello di controllo ci sono i controlli della modalità immagine, cliccando sull'icona dell'immagine stilizzata si aprirà un pannello a scomparsa da destra.





In questo pannello si regola il funzionamento di Image Mode, con il pulsante "Open" in alto si possono selezionare, tramite file picker di windowsm, una o più immagini da salvare localmente nell'applicazione, queste immagini convertite in grayscale sono inserite in un vettore di immagini nell'ordine in cui le abbiamo selezionate.
Sotto c'è la preview della prima immagine, evidenziata da un contorno blue, subito sopra di essa c'è l'immagine precedente nell'ordine dell' array (l'indice avvolge l'array, quindi arrivati all'ultima immagine si torna alla prima e viceversa), subito sotto l'immagine successiva.
Cliccando sulle frecciette verso l'alto o verso il basso si può scorrere il nostro array e selezionare l'immagine che vogliamo usare per deformare il flusso. L'influenza all'interno della simulazione non è ancora abilitata, per usare la nostra immagine come matrice di affinità della macro-simulazione bisogna spuntare "Image Flow", se imvece vogliamo anche sostituire interamente i valori dello stato di simulazione occorrerà spuntare "Set Image", in questo modo l'intensità luminosa dell'immagine sostituirà lo stato corrente e la simulazione di FlowLenia partirà dalla nostra immagine.

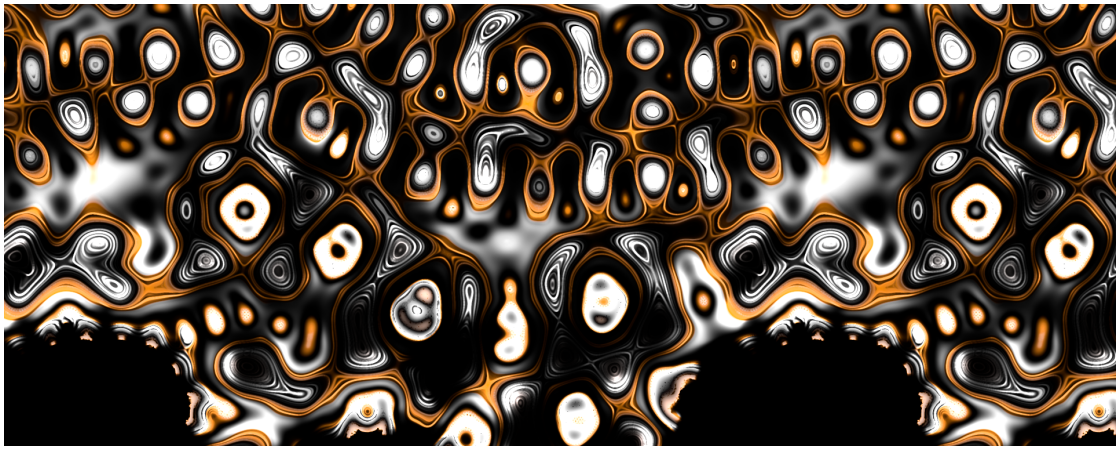
3.3.8 Sliders



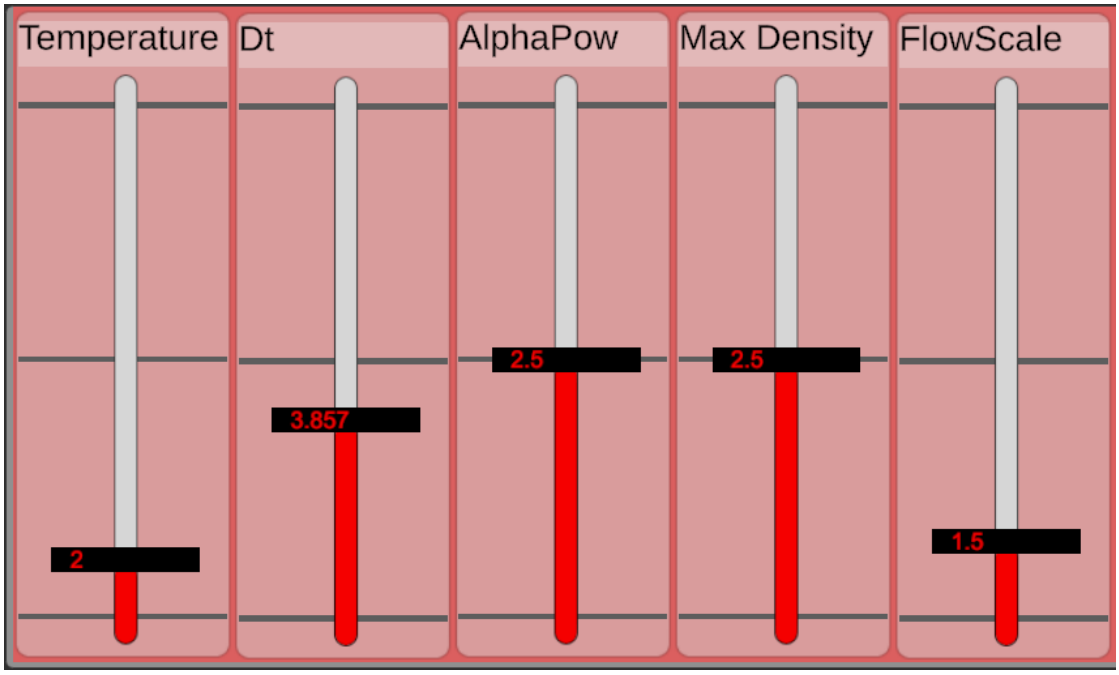
Il pannello inferiore riempie tutta la larghezza della scheda di controllo ed è adibito agli slider. Diversi colori dividono gli slider in tre categorie:

1. Slider per la micro-simulazione. (rossi)
2. Slider per la visualizzazione. (verdi)
3. Slider per la macro-simulazione. (grigi)

La conformazione e la posizione degli slider ricordano la conformazione di un mixer audio reale, questo mixer virtuale infatti è ideato per emulare questo strumento fisico.
Come i mixer reali permettono la modifica di più slider contemporaneamente, usando più dita per spostare più manopole, anche a questa applicazione gioverebbe poter cambiare più parametri in contemporanea. Per questo in futuro renderò ogni controllo di questo pannello associabile a un canale MIDI, così qualsiasi mixer MIDI può essere connesso al dispositivo e usato per mixare più comodamente ed efficacemente Lenia.
Ora l'applicazione è abilitata all'uso di dispositivi MIDI, uso un plug-in per unity chiamato MidiJack[7] che mi permette proprio di leggere i messaggi MIDI di un dispositivo collegato con USB al computer, ma non è ancora implementato in GUI e quindi non sarà disponibile nella prima versione dell'applicazione.



Sliders Rossi



Gli slider rossi gestiscono le variabili della micro-simulazione di FlowLenia che abbiamo spiegato più approfonditamente introducendo le tecnologie utilizzate. (2.5.2)

1. Temperature: è il parametro che descrive l'ampiezza della distribuzione randomica nella redistribuzione della massa. (2.5.4)

2. Dt: il moltiplicatore che regola la velocità del cambiamento della matrice di affinità. Un valore più alto risulterà in una simulazione più reattiva ai cambiamenti dei parametri, del kernel e della GrowthLUT. (2.5.2)

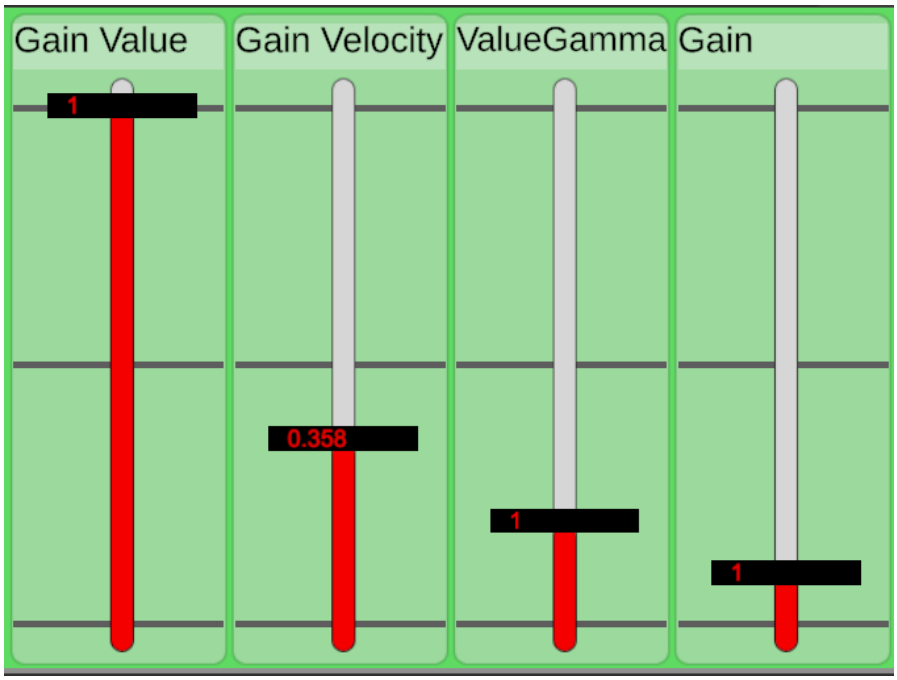
3. AlphaPow: il parametro n o gamma della funzione che regola le influenze del flusso di affinità e quello di repulsione. (2.5.3)

4. Max Density: il parametro θ della funzione nel calcolo di α e regola il massimo valore della massa che interagisce con la matrice di affinità. Tutti i pixel dal valore superiore verranno influenzati solo dal flusso repulsivo. (2.5.3)

5. FlowScale: il moltiplicatore del flusso finale della micro-simulazione, il flusso della simulazione intera è la combinazione lineare dei flussi della micro e macro-simulazione:

$$F_{tot} = FlowScale_m * MicroFlow + FlowScale_M * MacroFlow \qquad (3.3)$$

Slider Verdi



Gli slider verdi servono a gestire lo shader di viusalizzazione, in particolare l'intesità luminosa. La tonalità invece è descritta dalla ColorGrowthLUT. (3.3.3)

1. Gain Value: gain per la texture raffigurante la massa, la stessa texture usata per la simulazione di FlowLenia.

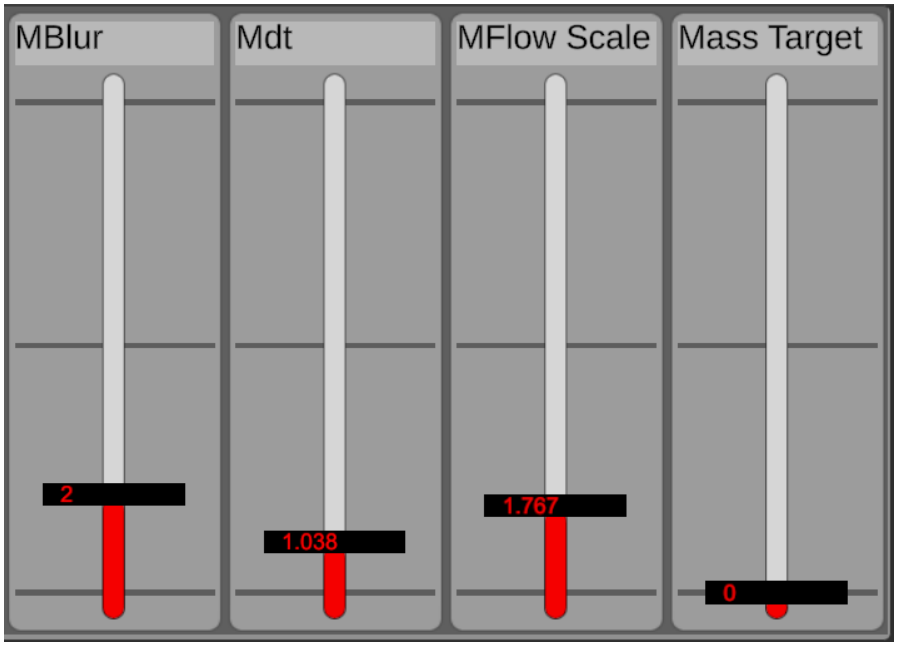
2. Gain Velocity: gain per la texture della velocità, estrapolata dal campo vettoriale del flusso.

3. Gain: gain totale della combinazione lineare della texture Value e della texture Velocity.

4. ValueGamma: gamma finale per incrementare il contrasto.

$$I_{tot} = (I_{value} * GainValue + I_{velocity} * GainVelocity)^{gamma} * Gain \tag{3.4}$$

Slider Grigi



Gli slider grigi sono gli slider che controllano la macro-simulazione e la percentuale di materia totale.

1. MBlur: Quantità di blur applicato sia al mondo della macro-simulazione prima del calcolo della convoluzione sia al flusso della macro-simulazione, in pratica addolcisce le macro-forze ed è utile se si vuole smuovere aree estese delicatamente.
2. Mdt: il moltiplicatore che regola la velocità del cambiamento della matrice di affinità della macro-simulazione.
3. MFlow Scale: il moltiplicatore del flusso finale della macro-simulazione.
4. Mass Target: serve a regolare la massa media del mondo simulato, valore che va da 0 a 1 con 0 mondo vuoto, 1 mondo completamente saturo.

Mass Target

Lo slider Mass Target ci permette di specificare un valore di massa media che il mondo deve assumere.
Per incrementare o diminuire la massa totale viene aggiunto a ciascun pixel un incremento calcolato come:

$$increment = cons \cdot (massTarget - mass)$$

Mass è calcolato come l'intensità media dei pixel, in questo modo se massTarget è maggiore di mass l'increamento sarà positivo e la massa media aumenterà fino a raggiungere massTarget e allora l'incremento tenderà a zero, se massTarget è minore di mass allora l'incremento sarà negativo.

3.4 Visualizzazione

nell'altro display, quello che abbiamo scelto all'avvio di Lenia Mixer come "Simulation display", verrà renderizzata la simulazione.
Qui non sono presenti elementi di interfaccia ma sono comunque implementate delle funzionalità.

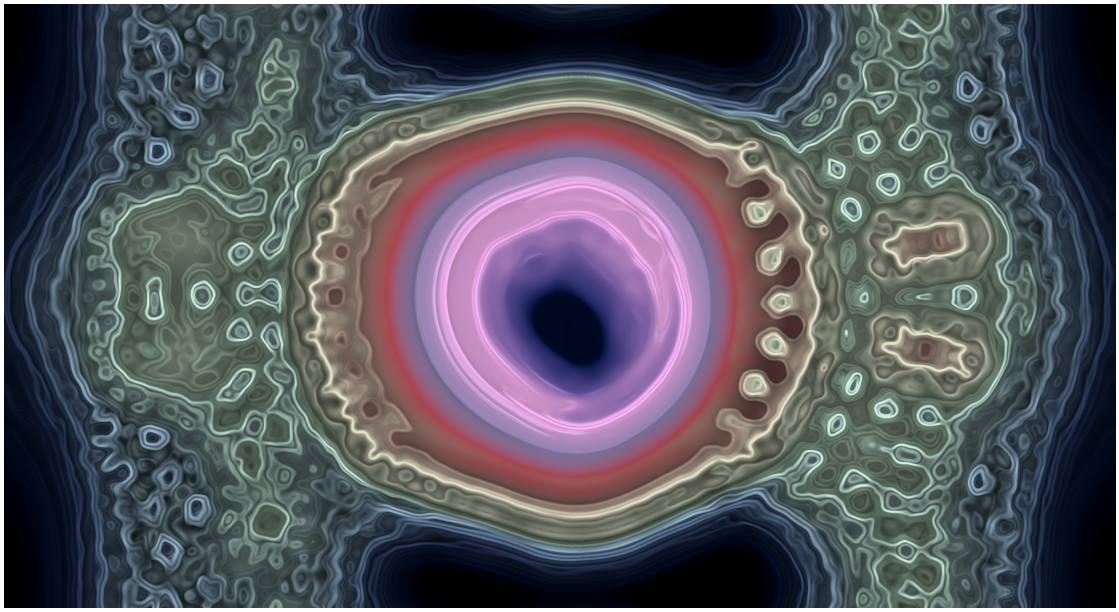
3.4.1 Spostamento, zoom, e disegno

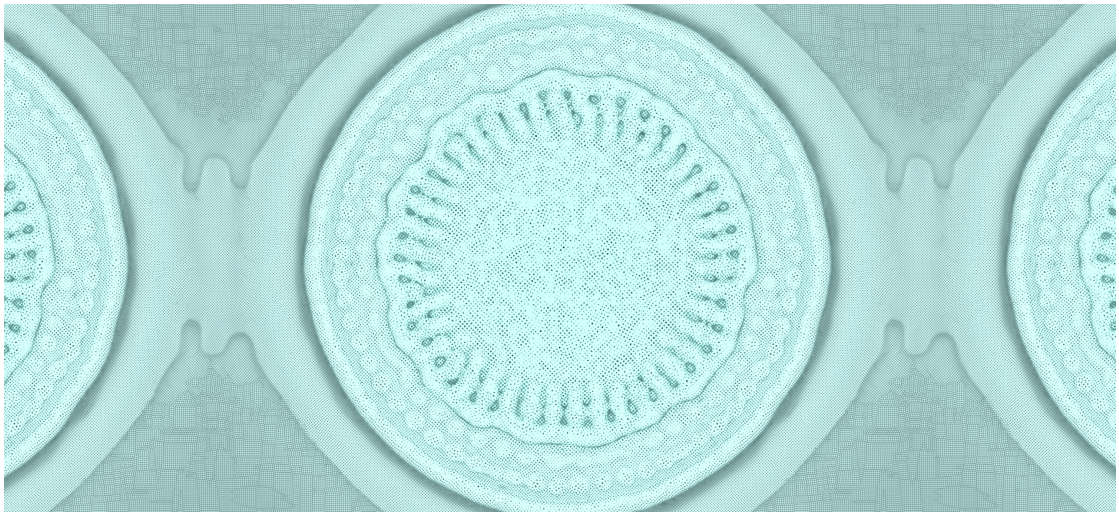
Muoversi nella simulazione

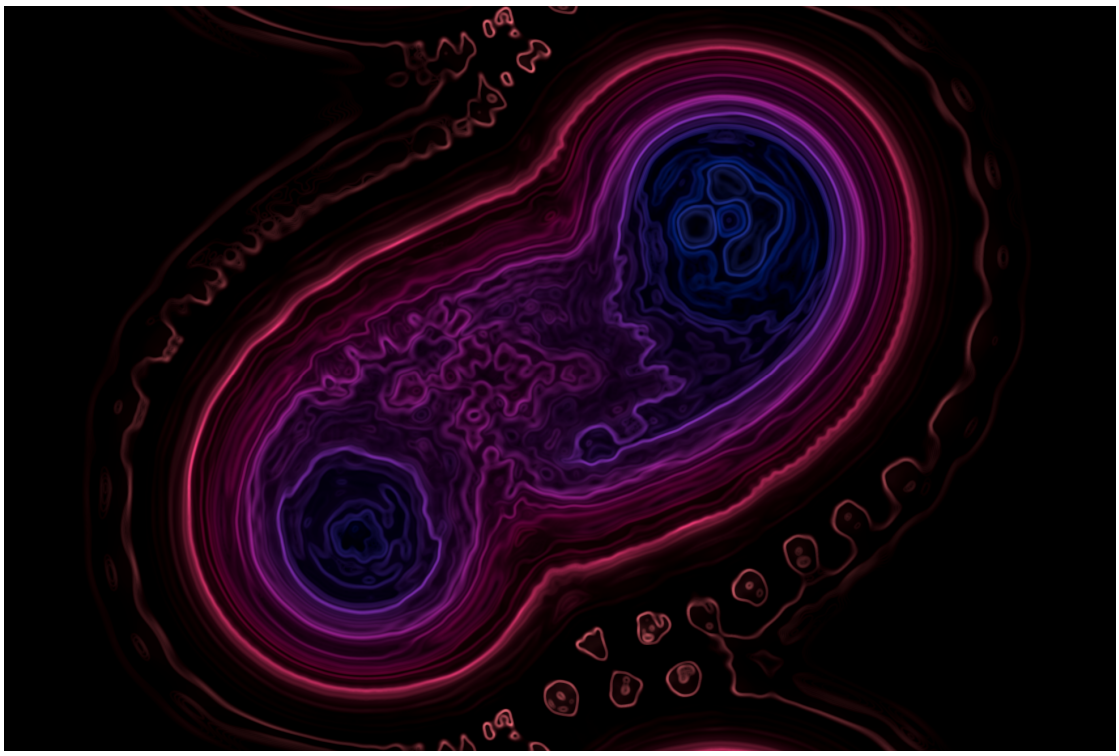
Trascinando il mouse con il tasto sinistro è possibile spostarci nella simulazione, rilasciando il tasto sinistro prima di aver concluso lo spostamento del mouse provoca una continuazione dello spostamento a velocità costante nella direzione di trascinamento, questo spostamento inerziale può essere fermato gradualmente cliccando con il tasto destro.
La dimensione della simulazione che abbiamo inserito nel menù iniziale non è limitata dalla risoluzione dello schermo ma serve per gestire il livello di dettaglio della texture mondo. Inoltre il mondo visualizzato è ripetuto all'infinito, la simulazione avviene su di un mondo toroidale e tutta la materia che esce dai bordi della texture del mondo si ritrova ai bordi opposti.
Con la rotella del mouse si effettua lo zoom.

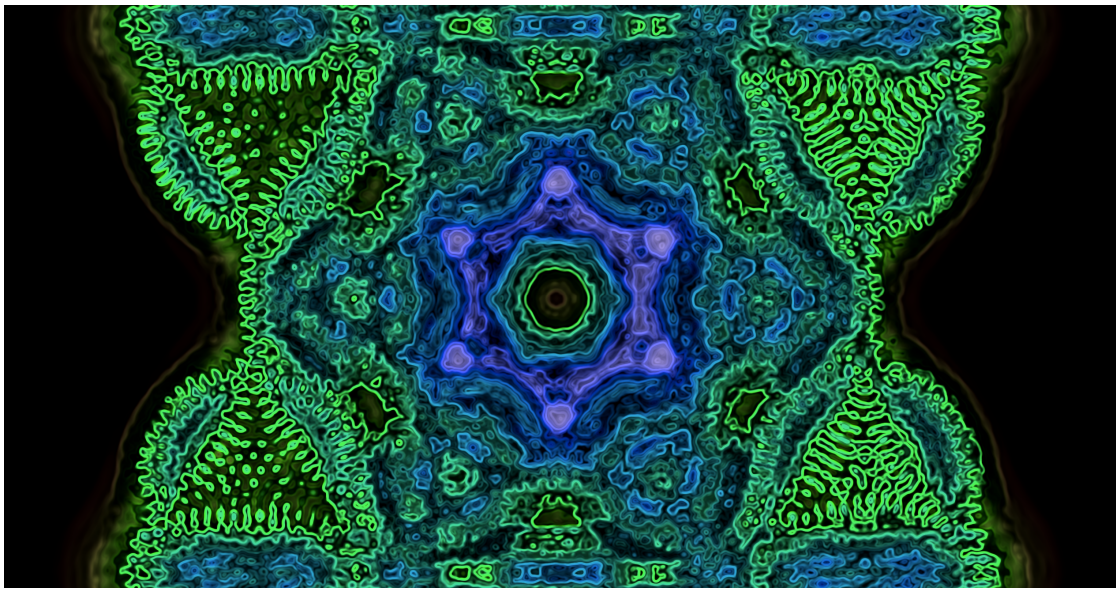
3.4.2 Paint mode

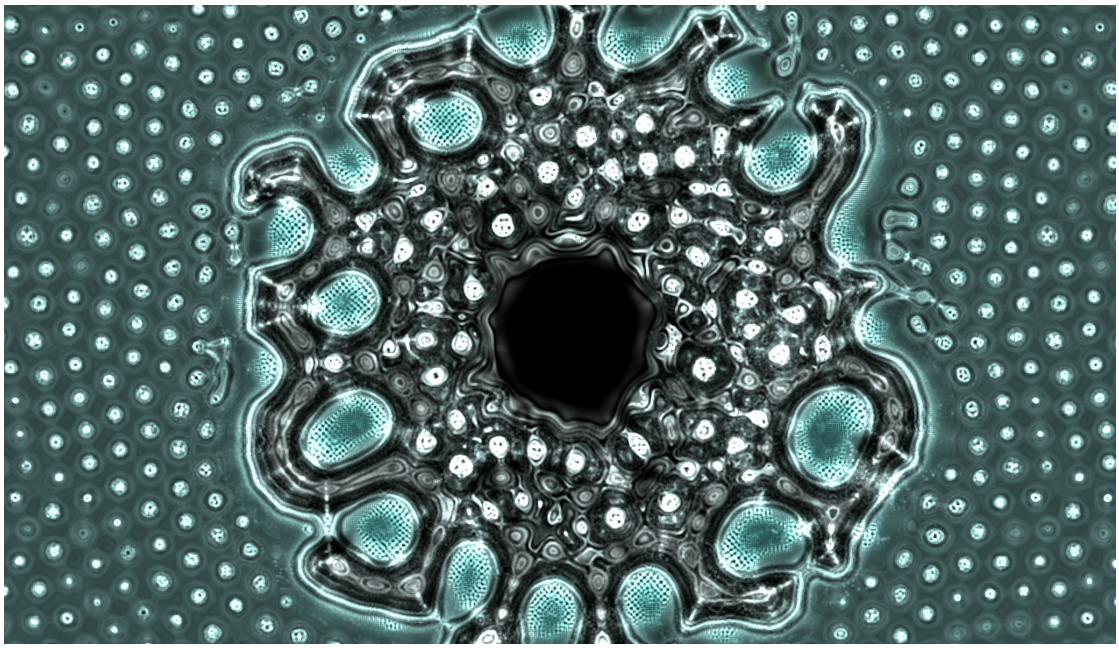
Una funzionalità creativa che permette di "dipingere massa", tenendo premuto ctrl si attivano i comandi della modalità disegno, in questa modalità è possibile regolare la dimensione del pennello, ruotando la rotellina del mouse, e aggiungere massa premendo il pulsante sinistro del mouse o rimuoverla con il destro.

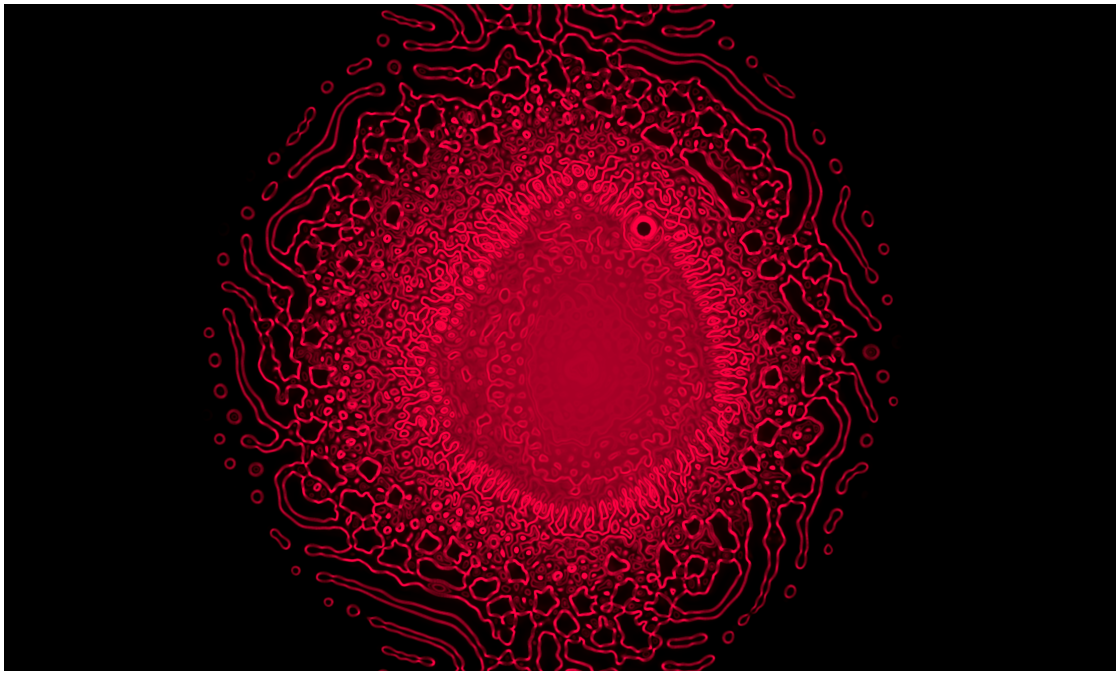


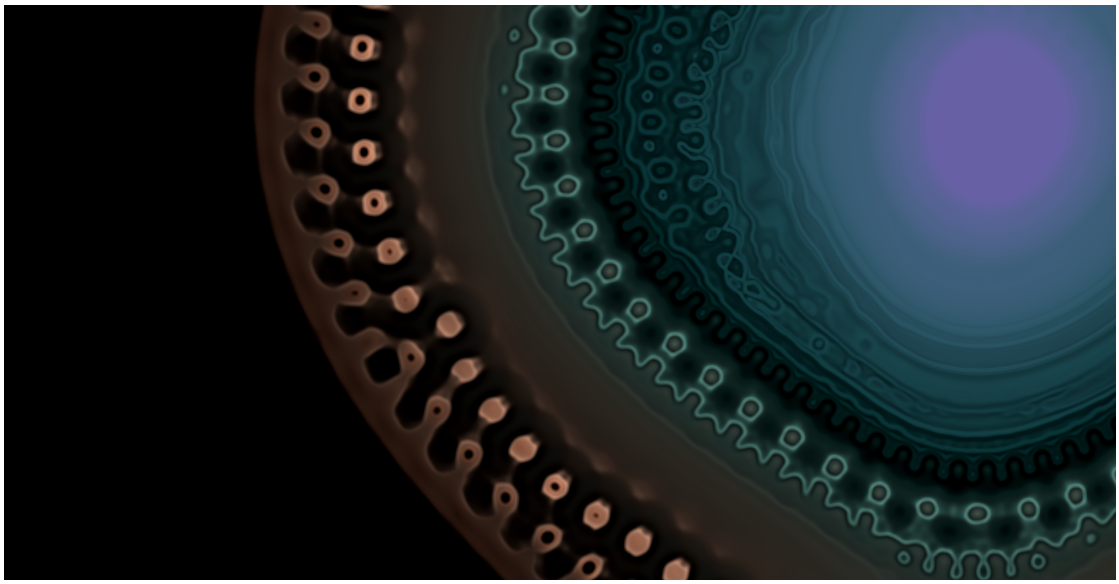


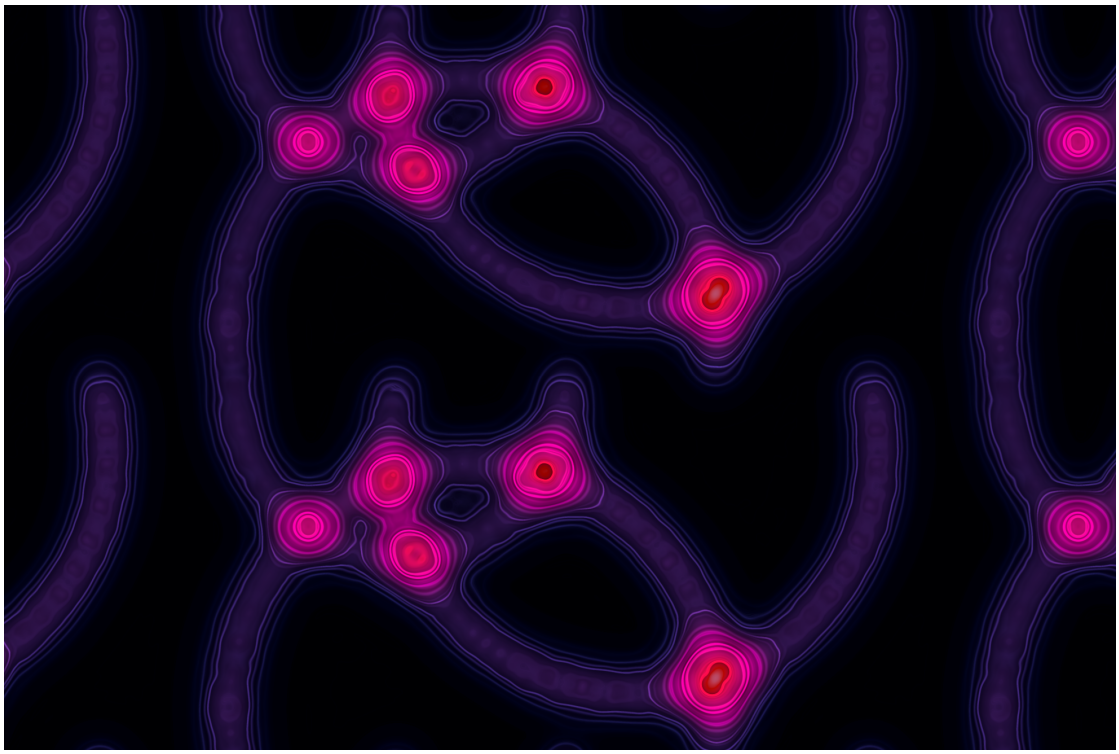


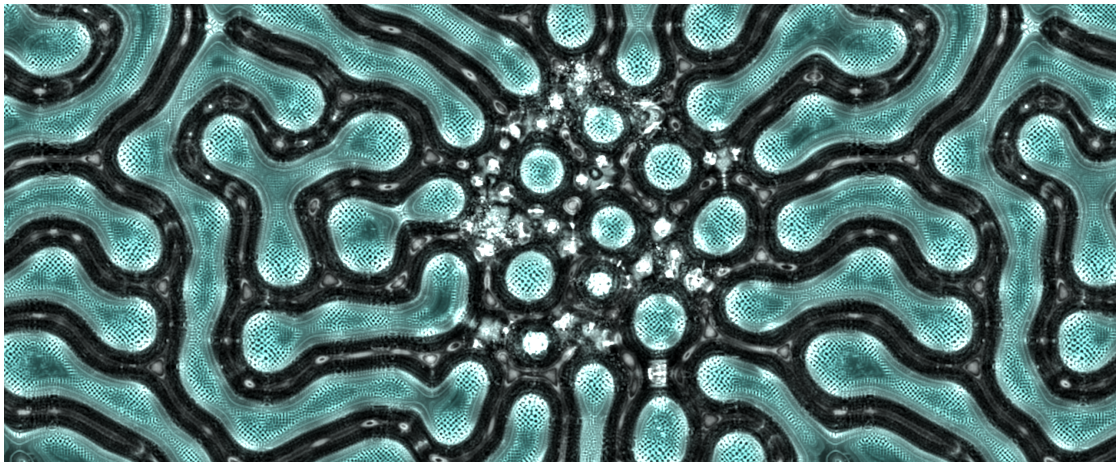


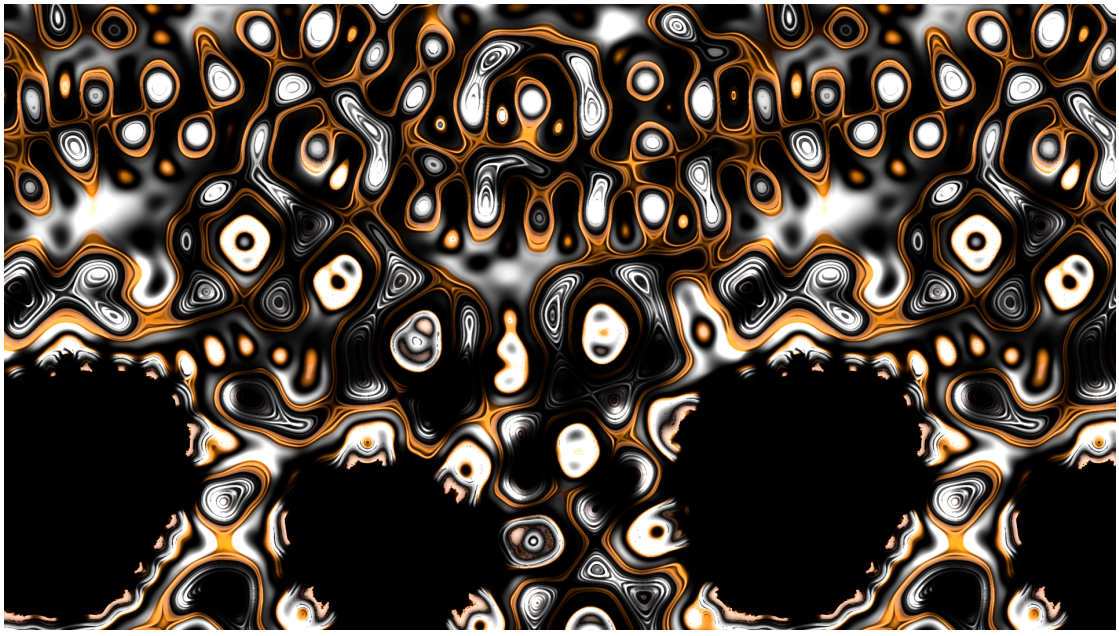












Conclusioni

4.1 Usabilità

Allo stato attuale il tool ha una interfaccia molto prototipale, frutto del mio "trial and error". La curva di apprendimento è abbastanza ripida se non si ha contezza del funzionamento matematico della simulazione, inoltre solo dopo aver molto navigato le combinazioni possibili dei parametri si può raggiungere la dimestichezza necessaria per controllare coscientemente il risultato della simulazione senza cambiare randomicamente i parametri.
Senza la dimestichezza necessaria è difficile anche restare al passo con la dinamica variabile di una canzone e cambiare la simulazione velocemente, accuratamente e creativamente.
L'approccio al Tool è facilitato radicalmente dall'uso dei controlli MIDI, avere tutto a portata di mano e poter cambiare più parametri contemporaneamente è incredibilmente utile per l'efficacia dell'esecuzione.
Nonostante sia consapevole che L'UI e l'architettura dell'interazione siano lontane dall'essere perfette, credo che la difficoltà intrinseca di dover modulare molti parametri contemporaneamente accomuni il mio strumento a un vero e proprio strumento musicale in cui l'esperienza e la dimestichezza con lo strumento sono alla base di un utilizzo virtuoso.

4.2 Prestazioni

4.2.1 Convoluzione e FFT

L'operazione più costosa dell'intera pipeline è la convoluzione tra la griglia del mondo e il kernel. Nella sua forma più semplice, ogni pixel deve sommare il contributo di tutti i pixel vicini pesati dal kernel, e questo calcolo va ripetuto per ogni pixel della griglia: al crescere della dimensione del kernel il costo diventa rapidamente proibitivo.
La soluzione adottata è calcolare la convoluzione nel dominio delle frequenze tramite FFT, in questo modo si trasforma questo problema in un semplice prodotto tra due matrici, con un guadagno computazionale che per kernel e griglie grandi può raggiungere diversi ordini di grandezza. La trasformata è implementata usando una libreria esterna: ComputeFFT di MatejLou[6]. Un'ulteriore ottimizzazione consiste nel precalcolare e cachare la trasformata del kernel, che cambia solo quando l'utente interviene nell'editor per cambiarne la forma.

4.2.2 Reintegration Tracking

Una delle operazioni più pesanti a runtime è il reintegration tracking, che si occupa di ridistribuire gli spostamenti della massa da ogni pixel verso i pixel vicini seguendo il campo vettoriale del flusso. Poiché più thread tentano di scrivere contemporaneamente sulle stesse celle, è necessario usare operazioni atomiche che serializzano gli accessi in scrittura, introducendo attese tra i thread. Il parametro temperature controlla quanto la massa viene dispersa durante questo processo: valori elevati aumentano il numero di pixel coinvolti nella redistribuzione e quindi il numero di queste operazioni seriali lente, con un impatto significativo sulle prestazioni.

4.2.3 Doppia pipeline e misura della massa

La presenza di due simulazioni parallele micro e macro raddoppia carico computazionale per frame, ma la macrosimulazione opera su una texture ridimensionata, di default dieci volte più piccola, contenendo il costo aggiuntivo. Il calcolo della massa globale, necessario per il controllo automatico della densità del mondo simulato, avviene anch'esso interamente sulla GPU tramite una riduzione parallela, evitando di dover trasferire l'intera texture alla CPU per farne la media.

4.3 Sviluppi futuri

4.3.1 Lenia e l'universo espanso

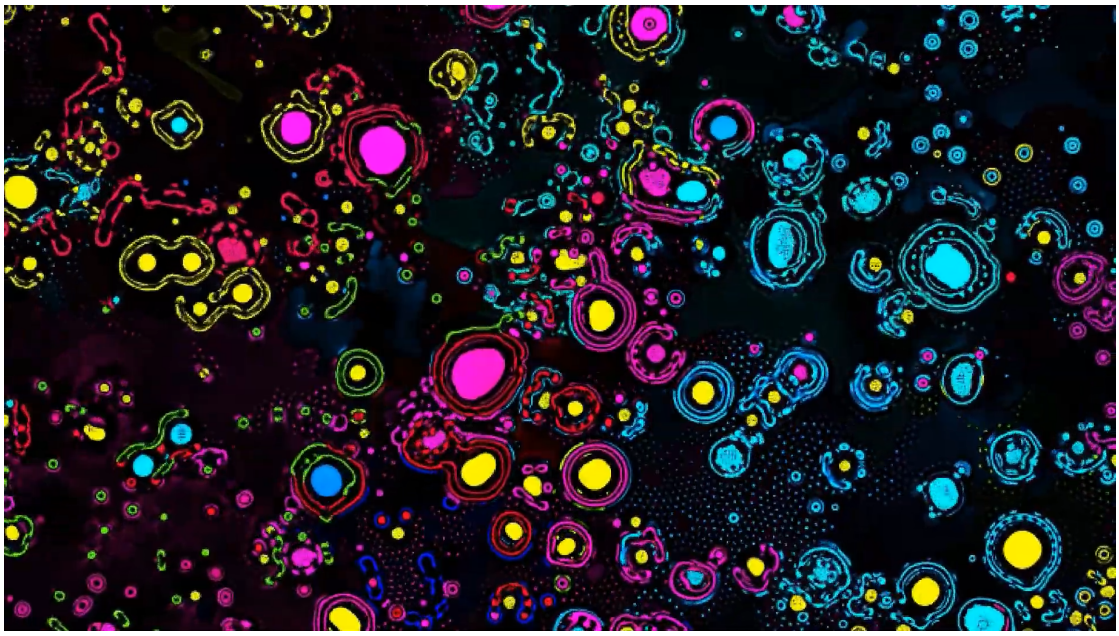


Figura 4.1: Expanded universe Lenia
`https://www.youtube.com/watch?v=bAJIETmC-6o&list=PLtIufgPHQa4RMXO9p-s9wxPDVnhPlULMO&index=4`

Una aggiunta possibile è usare un nuovo tipo di automa cellulare, la successiva evoluzione di Flow Lenia: Expanded Universe Flow Lenia.[5]
Esso consiste in più simulazioni Lenia parallele che però interagiscono tra loro, ogni simulazione ha un proprio calcolo della affinità che combina analisi dell'intorno incrociate tra le altre simulazioni, questo crea materia con comportamenti ancora più complessi, crea dei solitoni (pattern stabili) che sono simili a organelli cellulari che svolgono diversi compiti per mantenere la propria struttura.
Questa nuova simulazione crea dei pattern più caotici, organici e complessi di Lenia normale ma renderebbe Lenia Mixer molto meno controllabile, i parametri aumenterebbero esponenzialmente al numero di universi e sarebbe necessario abbassare la risoluzione per alleggerire la simulazione di più matrici in contemporanea.

4.3.2 Particle Life

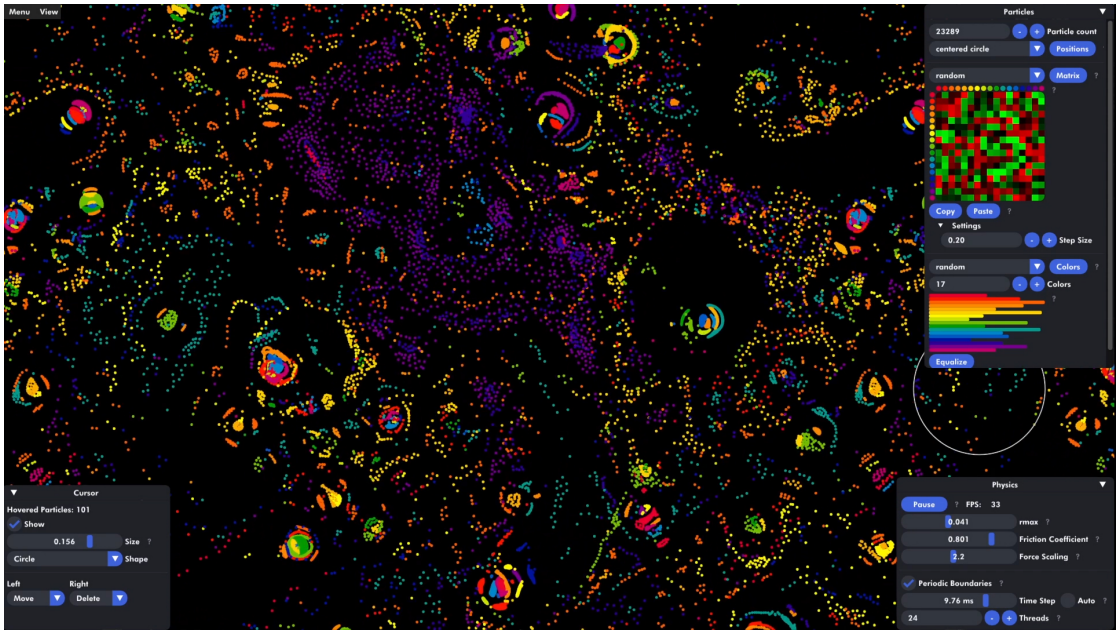


Figura 4.2: Particle Life Simulator
`https://particle-life.com/`

In futuro sarebbe interessante provare anche simulazioni diverse come Particle Life di Tom Mohr[8] a sua volta basato sul lavoro di Jeffrey Ventrella con il suo Clusters[9].
Particle Life è un sistema particellare che si basa sulla interazione di particelle di tipi diversi, ogni tipo è attratto o respinto dagli altri tipi in base a delle regole che descrivono le loro relazioni visualizzabili in una matrice di correlazione.
In pratica è una versione particellare di Lenia Expanded, più sempice e leggera, non è però stata ancora implementata in modo parallelo per scheda video.
Il passaggio da una simulazione basata su griglia a una basata su particelle cambierebbe completamente la struttura del mio programma e bisognerebbe ideare un modo per tradurre l'informazione dello spettro sonoro in una griglia di valori da usare per la matrice di correlazione.
Inoltre andrebbe pensato uno shader per la visualizzazione in quanto le particelle visualizzate con dei semplici puntini potrebbero annoiare velocemente.
Nonostante questo varrebbe la pena di provarla in quanto è un modo più semplificato per raggiungere un livello di complessità nell' emergenza simile a Lenia Expanded senza i pesanti calcoli delle molte convoluzioni incrociate e quindi un modo per mantenere una simulazione che consenta le interazioni in real time.

Bibliografia

[1] Bert Wang-Chak Chang. «Lenia: Biology of Artificial Life». In: *Complex Systems* (2019). URL: http://dx.doi.org/10.25088/ComplexSystems.28.3.251 (cit. alle pp. 7, 22, 24, 26).

[2] «Flow-Lenia: Towards open-ended evolution in cellular automata through mass conservation and parameter localization». In: (2023). URL: https://arxiv.org/abs/2212.07906 (cit. alle pp. 7, 26, 27, 29, 30).

[3] Conway John H. «The Game of Life». In: *Scientific American* (1970) (cit. a p. 16).

[4] M. Moroz. «Reintegration tracking». In: (2020). URL: https://michaelmoroz.github.io/Reintegration-Tracking/ (cit. a p. 29).

[5] «Lenia and Expanded Universe». In: (2020). URL: https://arxiv.org/abs/2005.03742 (cit. alle pp. 30, 66).

[6] MatejLou. *ComputeFFT*. URL: https://assetstore.unity.com/packages/tools/utilities/computefft-287079 (cit. alle pp. 37, 65).

[7] Keijiro Takahashi. *MidiJack*. URL: https://github.com/keijiro/MidiJack (cit. a p. 52).

[8] Tom Mohr. *Particle Life*. URL: https://github.com/tom-mohr/particle-life-app (cit. a p. 67).

[9] Jeffrey Ventrella. *Clusters*. URL: https://www.ventrella.com/Clusters/ (cit. a p. 67).